INSTITUTE FOR ADVANCED COMPUTINGAND

SOFTWARE DEVELOPMENT
AKURDI, PUNE

DOCUMENTATION ON

"WebGuard: Strengthening Website by Multi-Feature Security Model"

PG-DITISS September-2023

SUBMITTED BY:

GROUP NO: 11

MAYURI ASHOK SONONE (239422)

SHRADDHA NARESH RAUT (239443)

MR. ROHIT PURANIK
CENTRE CO-ORDINATOR

MISS. SUSHMA HATTARKI
PROJECT GUIDE

INDEX

Topics **Page No.**

1.INTRODUCTION

In an era where digital transformation continues to reshape the landscape of business and communication, the security of web servers and databases stands as an unyielding concern. As our reliance on the internet deepens, so too does the sophistication of cyber threats that seek to compromise the confidentiality, integrity, and availability of online resources. To address this ever-evolving challenge, this report documents a meticulous project undertaken to enhance the security of a web server and its associated database.

The core objective of this project is to design and implement a comprehensive security framework that encompasses multiple layers of defense. This framework draws upon a suite of industry-standard security tools and techniques, carefully configured to create a robust shield against a wide array of cyberthreats.

The project unfolds in a structured manner, beginning with the integration of ModSecurity and ModEvasive firewalls, strategically positioned to intercept and neutralize web-based attacks. These firewalls serve as the first line of defense, actively analyzing incoming traffic for signs of malicious intent and responding swiftly to thwart potential exploits.

Building upon this foundation, the project introduces Snort, a renowned intrusion detection system, coupled with Barnyard to provide real-time monitoring and alerting capabilities. This combination enhances the server's ability to detect and respond to suspicious activities, bolstering the overall securityposture.

The report also delves into the implementation of xtables-addons for GeoIP blocking, a feature that enables administrators to restrict access based on geographic origin, thus adding an additional layer of defense against region-specific threats. Furthermore, Fail2ban is configured to automate the identification and mitigation of malevolent behavior, enhancing the server's resilience against repeated login attempts and other malicious activities.

This project represents a comprehensive and practical approach to enhancing web server and database security. By documenting the meticulous configuration and integration of these security measures, this report aims to provide valuable insights and practical guidance for organizations and individuals seeking to fortify their online defenses in an increasingly perilous digital landscape.

2.EXISTING SYSTEM

Before implementing the project involving ModSecurity, the existing system likely lacked a dedicated layer ofsecurity specifically designed to protect the web applications. Without a web application firewall like ModSecurity, the system relied on the security measures provided by the web server and the application itself.

Here are some key points about the existing system:

Limited Protection: The existing system might have relied on basic security mechanisms provided by the web server, such as IP filtering, authentication, and basic access controls. However, these measures may not be sufficient to defend against more advanced and evolving threats.

Vulnerabilities: Without a dedicated web application firewall, the system was likely more susceptible to common attacks like SQL injection, cross-site scripting (XSS), and others. Attackers could exploit vulnerabilities in the application code without as much resistance.

Manual Security Patches: Security updates and patches for vulnerabilities in the application code might have been applied manually. This could lead to delays in addressing emerging threats and could potentially expose the system to attacks.

Higher Risk of Data Breaches: The absence of specialized security measures like ModSecurity might have put sensitive user data at a higher risk of being compromised in the event of a successful attack.

Limited Monitoring and Logging: The existing system might have lacked detailed monitoring and logging capabilities for detecting and investigating potential security incidents. This could make it difficult to identify and respond to threats in a timely manner.

No Rule-Based Protection: Unlike ModSecurity, the existing system might not have had the ability to enforce aset of predefined rules to detect and block suspicious or malicious traffic.

Scalability Challenges: As the system scaled and more users accessed the web applications, the risk of attacks could have increased, and the existing security measures might not have scaled effectively to meet the growingdemands.

3.PROPOSED SYSTEM

The proposed system aims to enhance the security and resilience of the web applications by implementing ModSecurity, a web application firewall (WAF). This addition will significantly improve the overall security posture of the system, providing a comprehensive defense against a variety of cyber threats and vulnerabilities.

Here are the key components and benefits of the proposed system:

ModSecurity Integration:
The heart of the proposed system is the integration of ModSecurity as a dedicated web application firewall. ModSecurity will act as an intermediary between the web server and incoming traffic, inspecting each request and response for potential security issues.

Rule-Based Protection:
ModSecurity operates using a rule-based approach, where a set of predefined rules will be configured to detectand block common attack patterns. These rules cover a wide range of threats, including SQL injection, cross- site scripting (XSS), command injection, and more.

Customizable Rule Sets:
The proposed system allows for the customization of rule sets. This means that rules can be tailored to the specific needs and characteristics of the web applications being protected. Custom rule sets can address application-specific vulnerabilities and requirements.

Positive and Negative Security Models:
ModSecurity supports both positive and negative security models. It can block requests that match known attack patterns (negative security), as well as validate input against allowed patterns (positive security), providing a versatile approach to threat mitigation.

Learning Mode and Tuning:
ModSecurity includes a learning mode feature that observes traffic patterns and generates rules based on legitimate traffic. This reduces the risk of false positives and streamlines the process of fine-tuning the rule sets.

Real-Time Monitoring and Logging:
The system will provide real-time monitoring and logging capabilities, allowing administrators to track and analyze incoming traffic, rule matches, and blocked requests. This data can be instrumental in identifying emerging threats and security incidents.

Enhanced Incident Response:
In the event of a security incident, the proposed system equipped with ModSecurity will provide valuable data for incident response and forensic analysis. This can aid in understanding the nature of the attack and the affected resources.

Reduced Attack Surface:
By proactively identifying and blocking malicious traffic, the proposed system reduces the attack surface of the web applications. This makes it significantly harder for attackers to exploit vulnerabilities and compromise sensitive data.

Scalability and Performance:
ModSecurity is designed to handle high traffic volumes efficiently. The proposed system can scale to accommodate growing user demands while maintaining robust security measures.

4.TECHNOLOGY USED

This project is built upon a carefully selected set of technologies and tools, each contributing to the creation of a robust and multi-layered security infrastructure. The following technical requirements form the foundation ofthe implementation:

Operating System: Debian 10
The project is hosted on the Debian 10 operating system, chosen for its stability, security features, and wide community support.

Web Server: Apache2
Apache2 is the chosen web server for its proven track record, flexibility, and compatibility with various web technologies.

Database Server: MariaDB
MariaDB serves as the backend database management system, known for its reliability and performance, ensuring secure storage and retrieval of data.

ModSecurity Firewall:
ModSecurity is integrated as a web application firewall, providing real-time monitoring and defense against web-based attacks and vulnerabilities.

ModEvasive Firewall:
ModEvasive complements the security setup with its ability to detect and mitigate DDoS and brute force attacks by analyzing traffic patterns.

Intrusion Detection System: Snort
Snort is employed as an intrusion detection system, designed to detect and respond to suspicious network activities, offering an additional layer of defense.

Intrusion Alert Management: Barnyard
Barnyard is utilized alongside Snort to facilitate the collection, management, and correlation of intrusion alertsgenerated by the Snort system.

Firewall Management: IPTables
IPTables is employed to manage and configure the network-level firewall rules, allowing for fine-grained control over incoming and outgoing traffic.

Geographical IP Blocking: xtables-addons
The xtables-addons extension is utilized to enforce GeoIP blocking, enabling the system to restrict access fromspecific geographic locations based on IP addresses.

Automated Ban System: Fail2ban
Fail2ban is configured to automatically ban IP addresses that exhibit malicious behavior, such as repeated failed login attempts, enhancing the system's self-defense capabilities.

By carefully aligning these technical requirements, the project aims to create a secure and resilient environmentfor the web server and database. The integration of these tools results in a multi-faceted defense strategy, safeguarding against a spectrum of cyber threats and vulnerabilities that could compromise the integrity and availability of the hosted resources.

4.1 APACHE HTTP SERVER (Apache2)

Apache HTTP Server, often referred to as Apache2, is a widely-used open-source web server software. It's developed and maintained by the Apache Software Foundation. Apache2 is known for its stability, performance, and flexibility, making it one of the most popular choices for hosting websites and web applications.

Key Features and Functionality:

HTTP Server: Apache2 primarily functions as a web server, serving web content to users' browsers in responseto HTTP requests.

Virtual Hosting: Apache2 supports virtual hosting, allowing a single physical server to host multiple websites with different domain names. This is particularly useful for shared hosting environments.

Modules: Apache2 is modular in design, allowing administrators to enable or disable various modules to add specific functionality. This modularity makes it adaptable to different use cases.

Security: Apache2 provides security features like SSL/TLS support for encrypted connections (HTTPS), authentication and access control mechanisms, and integration with web application firewalls like ModSecurity.

URL Rewriting: Apache2 supports URL rewriting, which enables administrators to define custom rules for manipulating URLs. This is useful for creating user-friendly URLs or redirecting requests.

Logging and Monitoring: Apache2 generates detailed logs that record incoming requests, server responses, andvarious events. These logs are valuable for troubleshooting, performance analysis, and security auditing.

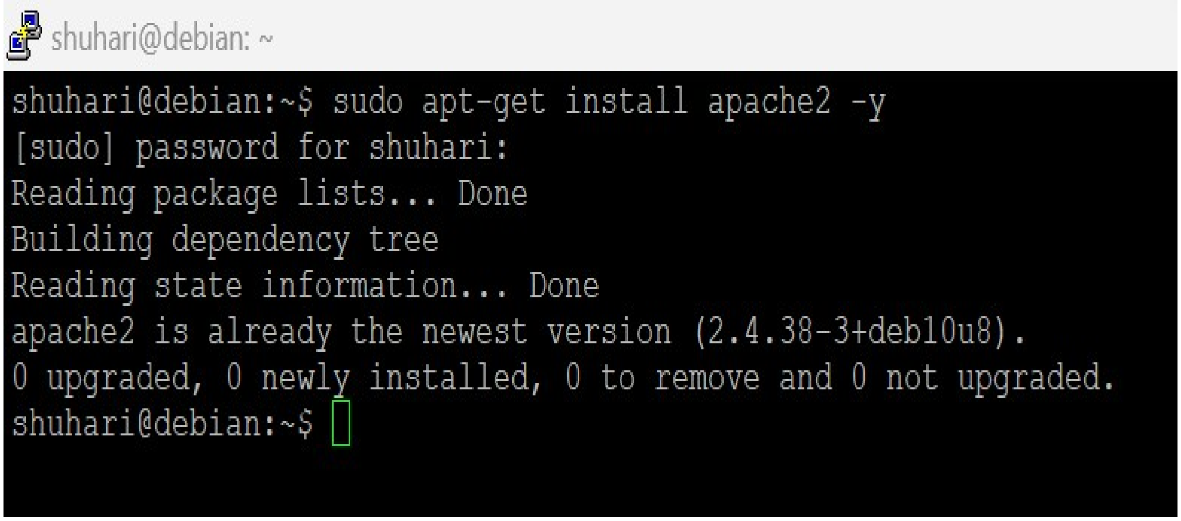
Performance: Apache2 is known for its performance optimizations, including support for multi-processing modules (MPMs) that allow efficient handling of concurrent requests.

Installation and Deployment:

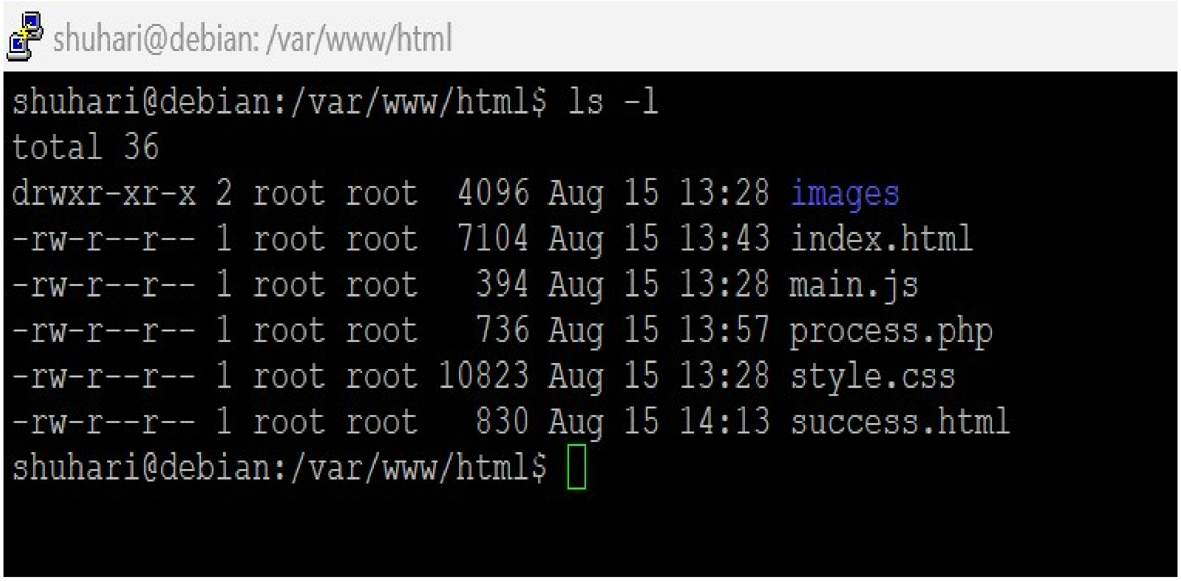
Operating Systems: Apache2 can be installed on various operating systems, including Linux, Windows, macOS, and more.

Package Managers: On Linux distributions, package managers like apt (Debian/Ubuntu) and yum (CentOS) arecommonly used to install Apache2.

Web Hosting Control Panels: Apache2 is often integrated with web hosting control panels like cPanel, Plesk, and Webmin for simplified management.



```
shuhari@debian:~$ sudo apt-get install apache2 -y
[sudo] password for shuhari:
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.38-3+deb10u8).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
shuhari@debian:~$
```



```
shuhari@debian:/var/www/html$ ls -l
total 36
drwxr-xr-x 2 root root  4096 Aug 15 13:28 images
-rw-r--r-- 1 root root  7104 Aug 15 13:43 index.html
-rw-r--r-- 1 root root   394 Aug 15 13:28 main.js
-rw-r--r-- 1 root root   736 Aug 15 13:57 process.php
-rw-r--r-- 1 root root 10823 Aug 15 13:28 style.css
-rw-r--r-- 1 root root   830 Aug 15 14:13 success.html
shuhari@debian:/var/www/html$
```

4.2 MARIADB (Database)

MariaDB is a widely-used, open-source relational database management system (RDBMS) that is built as a fork of MySQL. It is developed by the MariaDB Foundation and maintained by a community of contributors. MariaDB is known for its compatibility with MySQL, performance improvements, and advanced features.

Key Features and Functionality:

Relational Database Management: MariaDB provides a robust and scalable platform for storing, managing, andretrieving structured data in a relational format.

SQL Support: MariaDB supports the Structured Query Language (SQL), which is used for querying and manipulating data within the database.

ACID Compliance: MariaDB ensures ACID (Atomicity, Consistency, Isolation, Durability) compliance, ensuring data integrity and reliability.

Storage Engines: MariaDB supports multiple storage engines, allowing you to choose the most suitable one foryour use case. Common engines include InnoDB (the default), Aria, and MyISAM.

Replication: MariaDB offers robust replication features, enabling data to be duplicated across multiple servers for high availability, load balancing, and backup purposes.

Clustering: MariaDB provides clustering solutions like Galera Cluster, allowing for synchronous multi-master replication and automatic failover.

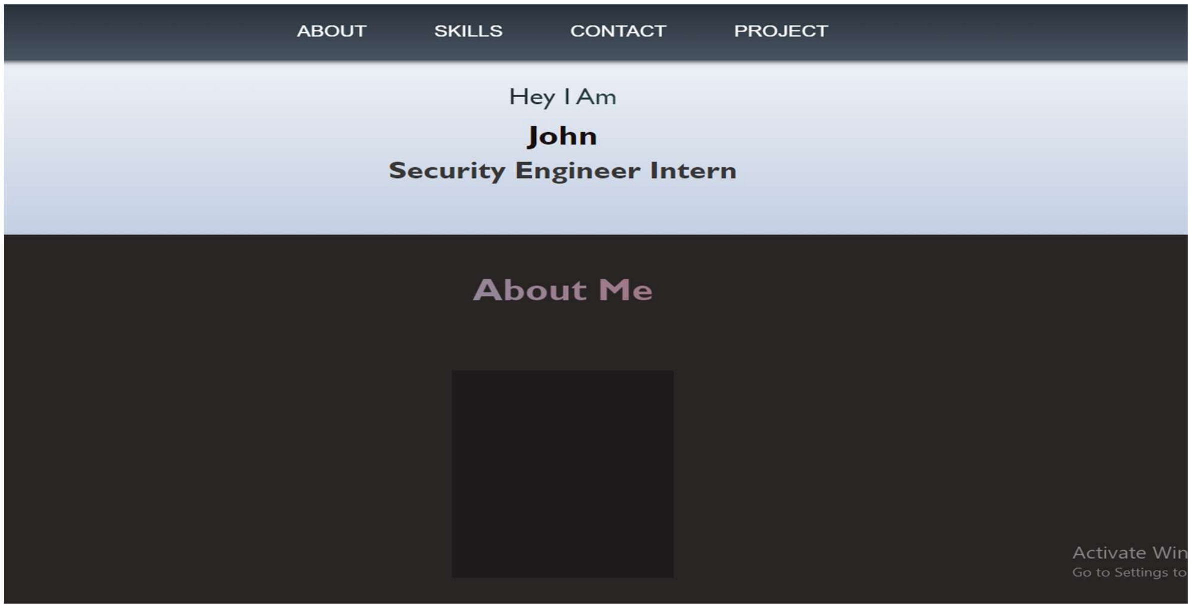
Partitioning: MariaDB supports table partitioning, which improves query performance for large datasets by distributing them across multiple file systems.

JSON Support: MariaDB has native support for JSON data types, allowing you to store, query, and manipulateJSON data within the database.

Full-Text Search: MariaDB includes full-text search capabilities, enabling efficient and accurate searches within textual data.

Security Features: MariaDB offers various security features, including user authentication, encryption, and access controls to safeguard data.

Lets check how website looks like :

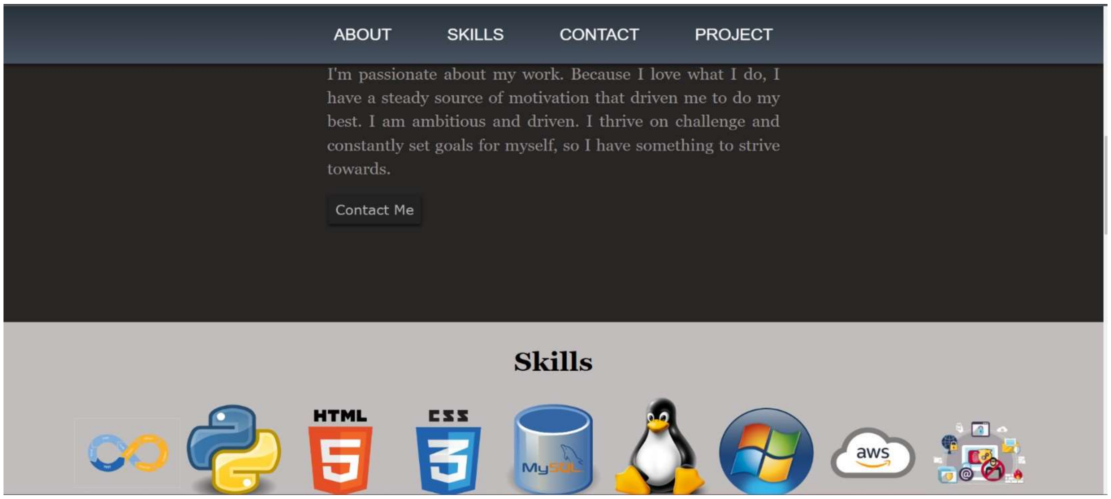


Hey I Am

John

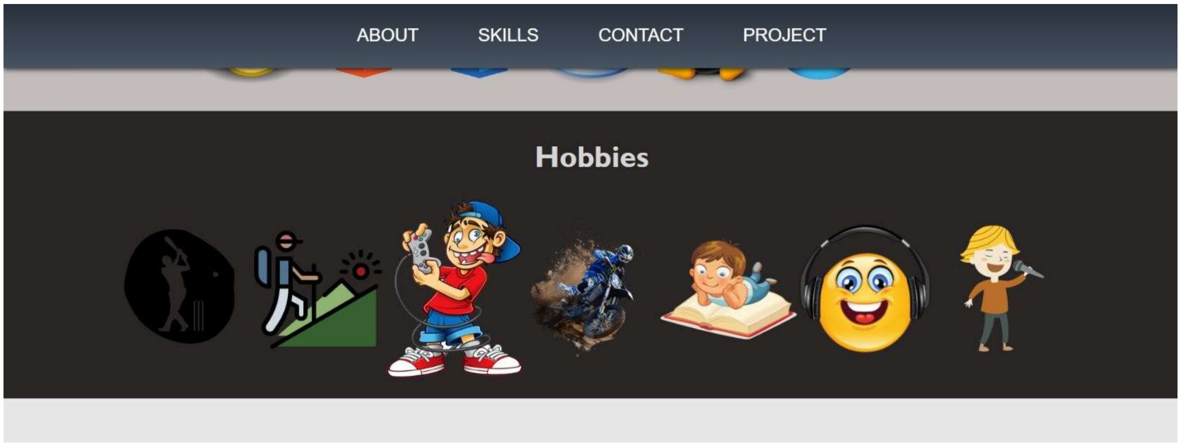
Security Engineer Intern

About Me

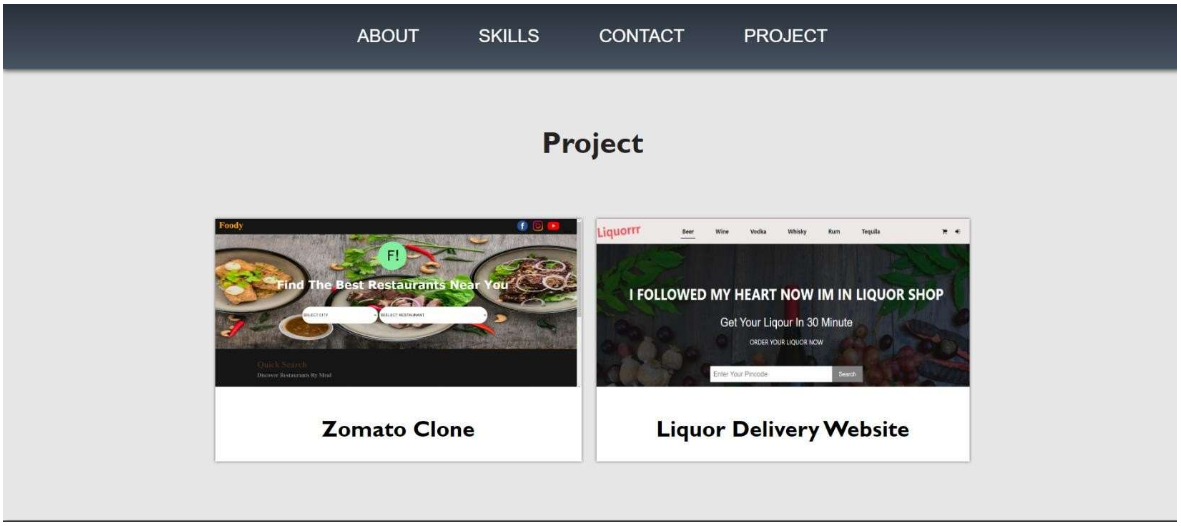


I'm passionate about my work. Because I love what I do, I have a steady source of motivation that driven me to do my best. I am ambitious and driven. I thrive on challenge and constantly set goals for myself, so I have something to strive towards.

Skills



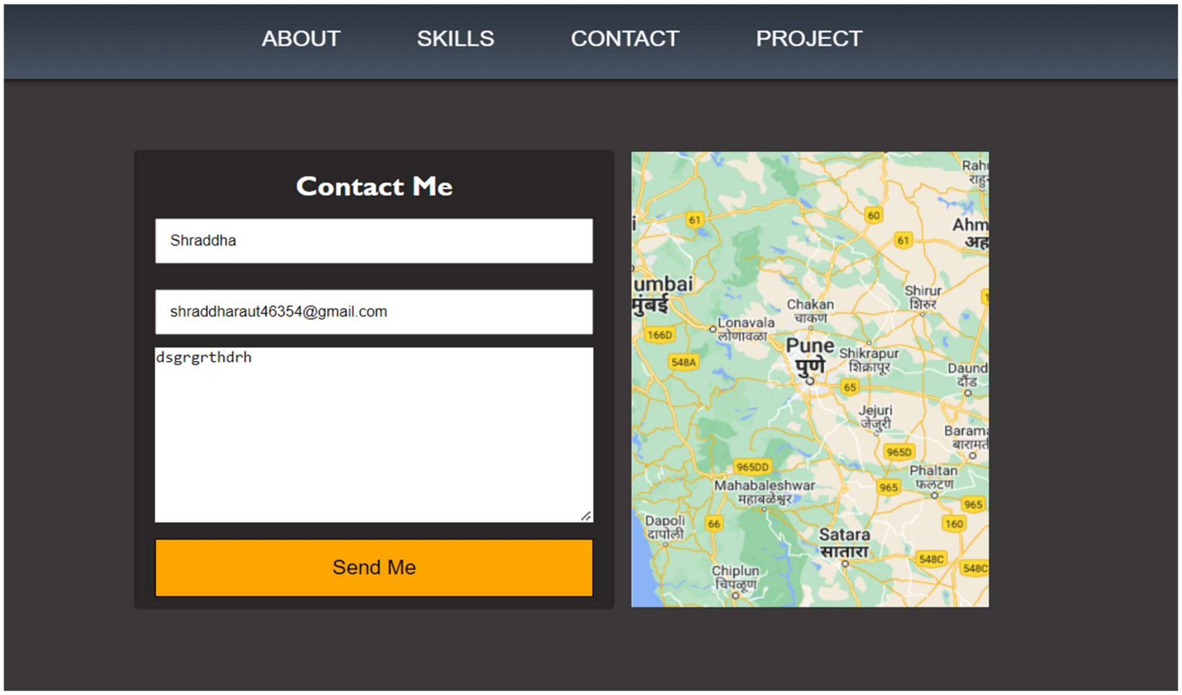
Hobbies



Project

Zomato Clone

Liquor Delivery Website



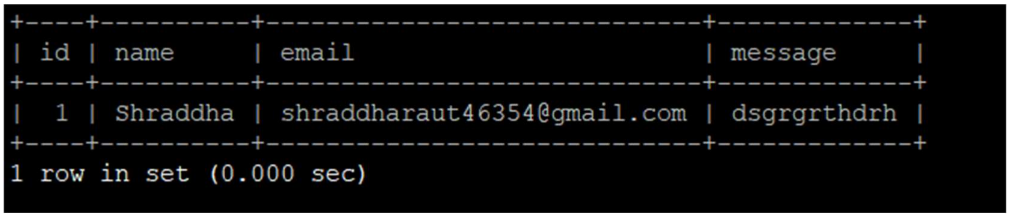
Contact Me

Shraddha

shraddharaut46354@gmail.com

dsgrgrthdrh

If we fill the information in the above form then it will go to the database as follows:



```
+----+----------+-----------------------------+-------------+
| id | name     | email                       | message     |
+----+----------+-----------------------------+-------------+
|  1 | Shraddha | shraddharaut46354@gmail.com | dsgrgrthdrh |
+----+----------+-----------------------------+-------------+
1 row in set (0.000 sec)
```

4.3 WEB APPLICATION FIREWALL

A Web Application Firewall (WAF) is a security system designed to protect web applications from various online threats and attacks. It sits between a user's browser and the web application server, monitoring and filtering incoming and outgoing HTTP/HTTPS traffic. The primary purpose of a WAF is to identify and block malicious traffic while allowing legitimate traffic to pass through.

Here are some key features and functions of a Web Application Firewall:

Traffic Filtering: WAFs inspect incoming web traffic for suspicious or malicious patterns, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and other common web application vulnerabilities.

Attack Detection: They employ a variety of techniques, including signature-based detection and behavior analysis, to identify known and zero-day attacks in real-time.

Protection Against OWASP Top Ten: Many WAFs are designed to protect against the OWASP (Open Web Application Security Project) Top Ten, which is a list of the most critical web application security risks.

Logging and Reporting: WAFs often provide logging capabilities, allowing administrators to review and analyze traffic patterns and potential security threats. They can generate reports for compliance purposes.

Access Control: Some WAFs provide access control features, allowing administrators to define and enforce access policies based on factors like IP addresses, user agents, or specific URL patterns.

Rate Limiting: WAFs can impose rate limits on incoming requests to prevent DDoS (Distributed Denial of Service) attacks or excessive resource consumption by bots.

SSL/TLS Termination: Many WAFs can handle SSL/TLS encryption and decryption, which is useful for inspecting HTTPS traffic for security threats.

We have implemented two open-source web application firewall as follows :

Modsecurity Firewall :

ModSecurity is not a firewall in the traditional sense, but rather a web application firewall (WAF) module that can be integrated with web servers like Apache and Nginx to provide an additional layer of security for web applications. It helps protect web applications from various attacks, such as SQL injection, cross-site scripting (XSS), and other malicious activities.

ModSecurity operates by intercepting and analyzing incoming HTTP requests before they reach the web application. It evaluates the requests based on a set of predefined rules and policies to identify and block potentially malicious traffic. If a request is flagged as suspicious or violating a rule, ModSecurity can take actions like blocking the request, logging the event, or customizing the response.

Key features of ModSecurity include:

Rule-Based Security: ModSecurity uses a set of rules that define the criteria for identifying malicious or suspicious activity. These rules can be customized to suit the specific needs of an application.

Protection Against Known Attacks: ModSecurity can protect against a wide range of common web applicationvulnerabilities and attacks, such as SQL injection, XSS, remote file inclusion, and more.

Real-Time Monitoring: It can log and monitor incoming requests in real-time, allowing administrators to analyze traffic patterns and potential threats.

Customization: ModSecurity allows users to create custom rules to address specific security requirements or application behaviors.

Positive and Negative Security Models: ModSecurity supports both positive and negative security models. In the negative model, it blocks known attack patterns, while in the positive model, it only allows traffic that matches a predefined set of safe patterns.

Integration: ModSecurity can be integrated into various web servers, including Apache and Nginx, making it compatible with a wide range of web applications.

Learning Mode: ModSecurity can be configured in a learning mode where it monitors and logs traffic without actively blocking anything. This mode helps administrators fine-tune rules without disrupting the application.

Mod-Evasive Firewall :

Mod_evasive is an Apache module that functions as a network and application layer firewall to help protect web servers from Distributed Denial of Service (DDoS) and brute force attacks. It's not a traditional firewall like those found at the network level, but rather a module that enhances the security of your web server by mitigating certain types of attacks.

Key features of Mod_evasive include:

DDoS Protection: Mod_evasive helps prevent DDoS attacks by detecting excessive requests from a single IP address or user and taking actions to block or throttle that traffic.

Brute Force Protection: It can also protect against brute force attacks, where attackers try to gain unauthorized access by repeatedly attempting different username and password combinations.

IP Whitelisting: You can specify certain IP addresses or ranges to be whitelisted, allowing them to bypass the module's protection mechanisms.

Detection and Blocking: Mod_evasive uses configurable thresholds to determine what constitutes excessive traffic from a single source. When these thresholds are exceeded, the module can take actions like blocking theoffending IP address or serving a 403 Forbidden response.

Customizable Configuration: The module's behavior can be customized based on your application's requirements and traffic patterns. You can adjust settings like detection thresholds, blocking mechanisms, and logging options.

Logging: Mod_evasive can log suspicious activities, allowing you to review and analyze potential attacks.

Configurable Whitelists: You can define URL patterns or regular expressions to exclude specific URLs or types of requests from being subject to the module's protections.

4.4 SNORT

Snort is an open-source network intrusion detection and prevention system (NIDS/NIPS) that is widely used to monitor network traffic and detect signs of malicious or unauthorized activities. It operates by analyzing packets of data as they move through a network and comparing them against a set of predefined rules to identify potential security threats. Snort is highly customizable and can be configured to suit the specific security needs of a web server environment.

Here's how Snort works and how it can help protect a web server:

Packet Analysis:
Snort captures network packets that flow through the network interface of the system where it's installed. These packets contain information about the data being transmitted across the network, including source and destination IP addresses, port numbers, protocols used, and the payload of the packets.

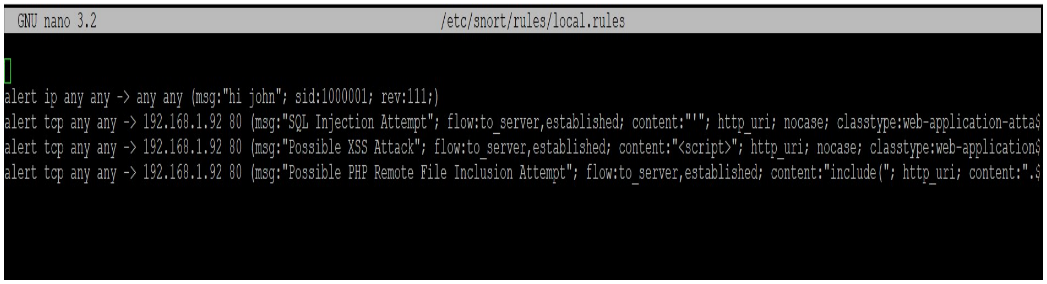
Rule-Based Detection:
Snort uses a set of rules to analyze the captured packets and detect signs of malicious or suspicious activity. These rules define specific patterns, signatures, and behaviors associated with known attacks or vulnerabilities. For example, a rule might specify that if a packet contains certain keywords or matches a known attack pattern, an alert should be generated.

Alert Generation:
When Snort identifies packets that match the patterns specified in its rules, it generates alerts to notify administrators about the potential security threat. These alerts provide details about the detected activity, including the source and destination IP addresses, the type of attack or behavior observed, and other relevant information.

Logging and Reporting:
Snort logs the alerts it generates, along with information about the packets that triggered those alerts. This logging can help administrators review and investigate the detected incidents. The logged data can also be used for reporting and analysis, helping to identify trends and patterns in attack attempts.

Customization:
One of Snort's strengths is its flexibility and customization. Administrators can create and modify rules to match the specific security requirements of their web server environment. This means that Snort can be tailoredto detect threats that are relevant to the specific web applications and services running on the server.

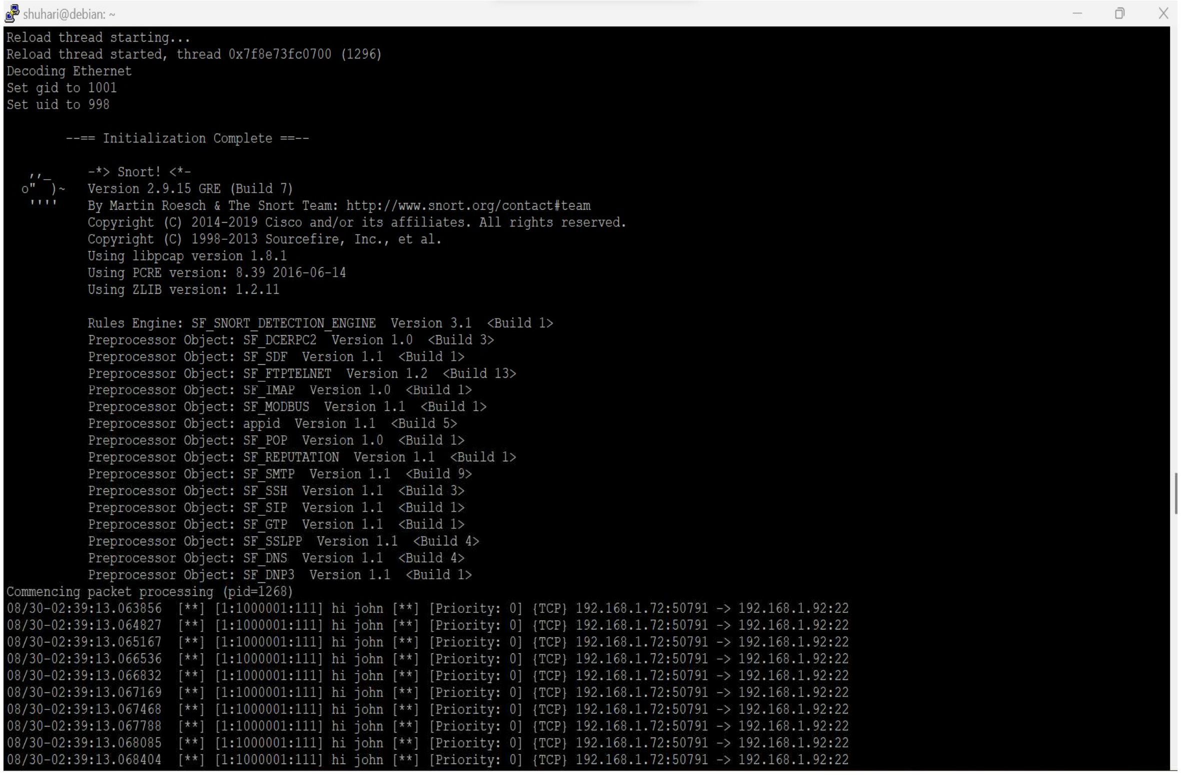
We have imposed some rule in /etc/snort/rules/local.rules to detect malicious activities :



```
 GNU nano 3.2                                   /etc/snort/rules/local.rules


alert ip any any -> any any (msg:"hi john"; sid:1000001; rev:111;)
alert tcp any any -> 192.168.1.92 80 (msg:"SQL Injection Attempt"; flow:to_server,established; content:"'"; http_uri; nocase; classtype:web-application-atta$
alert tcp any any -> 192.168.1.92 80 (msg:"Possible XSS Attack"; flow:to_server,established; content:"<script>"; http_uri; nocase; classtype:web-application$
alert tcp any any -> 192.168.1.92 80 (msg:"Possible PHP Remote File Inclusion Attempt"; flow:to_server,established; content:"include("; http_uri; content:".$
```

It will detect some activities and alert to the administrator based on rules who is monitoring :



```
Reload thread starting...
Reload thread started, thread 0x7f8e73fc0700 (1296)
Decoding Ethernet
Set gid to 1001
Set uid to 998

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15 GRE (Build 7)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           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: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
Commencing packet processing (pid=1268)
08/30-02:39:13.063856  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.064827  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.065167  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.066536  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.066832  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.067169  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.067468  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.067788  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.068085  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
08/30-02:39:13.068404  [**] [1:1000001:111] hi john [**] [Priority: 0] {TCP} 192.168.1.72:50791 -> 192.168.1.92:22
```

4.5 BARNYARD

Barnyard is a companion application that works in conjunction with Snort, an open-source intrusion detection and prevention system. Barnyard helps manage and process the alerts generated by Snort, allowing for more efficient storage, analysis, and management of intrusion detection data. It acts as a "post-processor" for Snort alerts.

Here's how Barnyard typically works with Snort:

Alert Generation: Snort analyzes network traffic and generates alerts when it detects suspicious or malicious activity based on the configured rules.

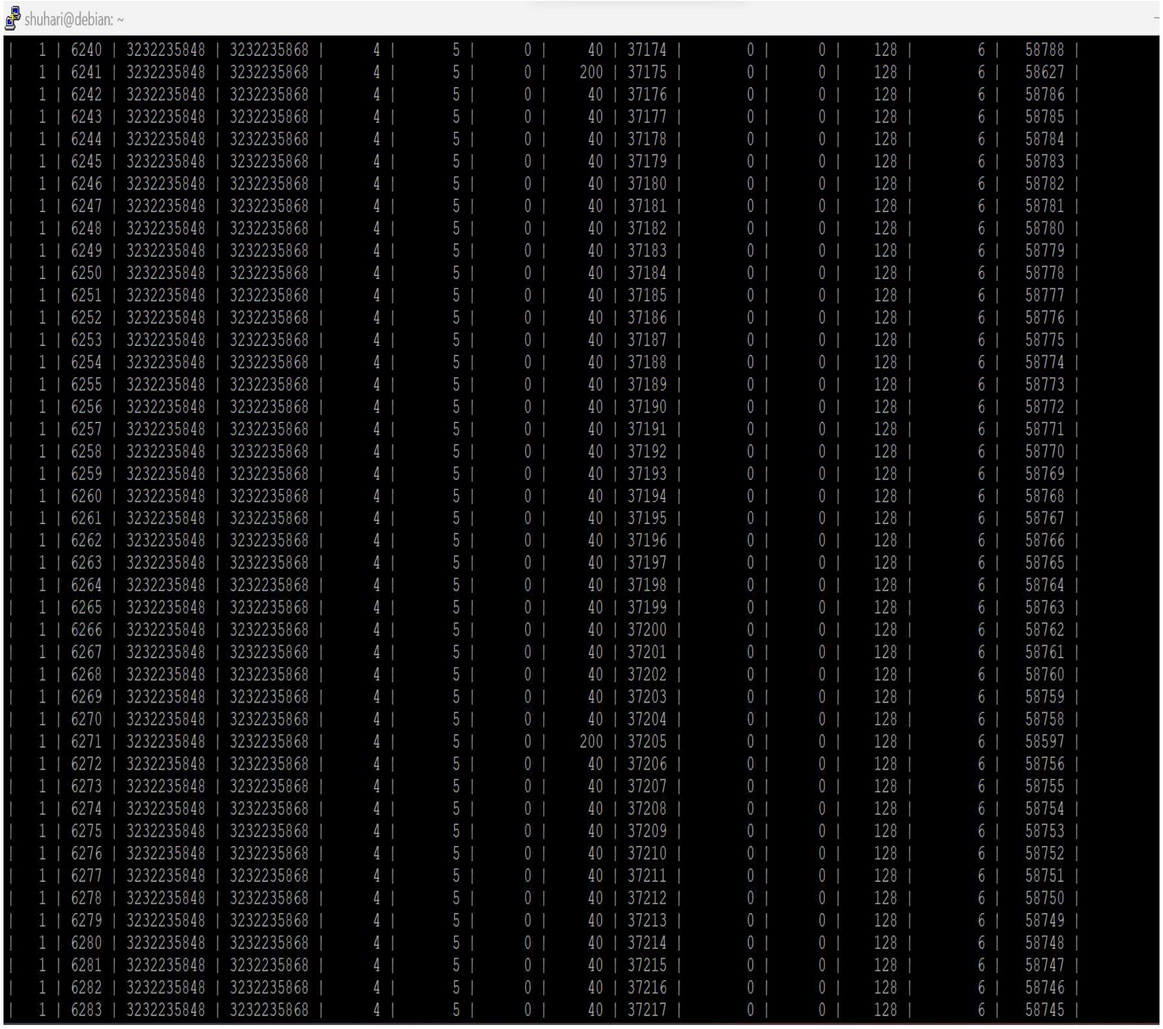
Alert Output: Instead of managing and processing the alerts directly, Snort can output the generated alerts in binary Unified2 format. This format provides a more streamlined and efficient way to store alert data compared to the traditional ASCII text format.

Barnyard Processing: Barnyard reads the Unified2 output generated by Snort and processes it. It extracts the alert information, transforms it into a human-readable format, and then forwards it to various destinations, such as a database, syslog, or another log management system.

Storage and Analysis: By offloading the alert processing to Barnyard, Snort can continue analyzing network traffic without being bogged down by alert management tasks. Barnyard facilitates the storage and analysis of alert data, making it easier for security analysts to review and respond to potential threats.

Barnyard was developed to improve the performance and efficiency of Snort-based intrusion detection systems by separating the alert generation from the alert management process. It's worth noting that the development and support for Barnyard have slowed down in recent years, and some alternative approaches and tools have emerged for managing Snort alerts, including the use of tools like PulledPork, Sagan, and custom scripts to process and analyze alert data.

The data generated by snort will save to the database by using barnyard as follows :



```
|    1 | 6240 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37174 |         0 |       0 |     128 |         6 |    58788 |
|    1 | 6241 | 3232235848 | 3232235868 |       4 |       5 |       0 |     200 | 37175 |         0 |       0 |     128 |         6 |    58627 |
|    1 | 6242 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37176 |         0 |       0 |     128 |         6 |    58786 |
|    1 | 6243 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37177 |         0 |       0 |     128 |         6 |    58785 |
|    1 | 6244 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37178 |         0 |       0 |     128 |         6 |    58784 |
|    1 | 6245 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37179 |         0 |       0 |     128 |         6 |    58783 |
|    1 | 6246 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37180 |         0 |       0 |     128 |         6 |    58782 |
|    1 | 6247 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37181 |         0 |       0 |     128 |         6 |    58781 |
|    1 | 6248 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37182 |         0 |       0 |     128 |         6 |    58780 |
|    1 | 6249 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37183 |         0 |       0 |     128 |         6 |    58779 |
|    1 | 6250 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37184 |         0 |       0 |     128 |         6 |    58778 |
|    1 | 6251 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37185 |         0 |       0 |     128 |         6 |    58777 |
|    1 | 6252 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37186 |         0 |       0 |     128 |         6 |    58776 |
|    1 | 6253 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37187 |         0 |       0 |     128 |         6 |    58775 |
|    1 | 6254 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37188 |         0 |       0 |     128 |         6 |    58774 |
|    1 | 6255 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37189 |         0 |       0 |     128 |         6 |    58773 |
|    1 | 6256 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37190 |         0 |       0 |     128 |         6 |    58772 |
|    1 | 6257 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37191 |         0 |       0 |     128 |         6 |    58771 |
|    1 | 6258 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37192 |         0 |       0 |     128 |         6 |    58770 |
|    1 | 6259 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37193 |         0 |       0 |     128 |         6 |    58769 |
|    1 | 6260 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37194 |         0 |       0 |     128 |         6 |    58768 |
|    1 | 6261 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37195 |         0 |       0 |     128 |         6 |    58767 |
|    1 | 6262 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37196 |         0 |       0 |     128 |         6 |    58766 |
|    1 | 6263 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37197 |         0 |       0 |     128 |         6 |    58765 |
|    1 | 6264 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37198 |         0 |       0 |     128 |         6 |    58764 |
|    1 | 6265 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37199 |         0 |       0 |     128 |         6 |    58763 |
|    1 | 6266 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37200 |         0 |       0 |     128 |         6 |    58762 |
|    1 | 6267 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37201 |         0 |       0 |     128 |         6 |    58761 |
|    1 | 6268 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37202 |         0 |       0 |     128 |         6 |    58760 |
|    1 | 6269 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37203 |         0 |       0 |     128 |         6 |    58759 |
|    1 | 6270 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37204 |         0 |       0 |     128 |         6 |    58758 |
|    1 | 6271 | 3232235848 | 3232235868 |       4 |       5 |       0 |     200 | 37205 |         0 |       0 |     128 |         6 |    58597 |
|    1 | 6272 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37206 |         0 |       0 |     128 |         6 |    58756 |
|    1 | 6273 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37207 |         0 |       0 |     128 |         6 |    58755 |
|    1 | 6274 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37208 |         0 |       0 |     128 |         6 |    58754 |
|    1 | 6275 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37209 |         0 |       0 |     128 |         6 |    58753 |
|    1 | 6276 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37210 |         0 |       0 |     128 |         6 |    58752 |
|    1 | 6277 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37211 |         0 |       0 |     128 |         6 |    58751 |
|    1 | 6278 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37212 |         0 |       0 |     128 |         6 |    58750 |
|    1 | 6279 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37213 |         0 |       0 |     128 |         6 |    58749 |
|    1 | 6280 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37214 |         0 |       0 |     128 |         6 |    58748 |
|    1 | 6281 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37215 |         0 |       0 |     128 |         6 |    58747 |
|    1 | 6282 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37216 |         0 |       0 |     128 |         6 |    58746 |
|    1 | 6283 | 3232235848 | 3232235868 |       4 |       5 |       0 |      40 | 37217 |         0 |       0 |     128 |         6 |    58745 |
```

4.6 GEO IP BLOCKING USING XTABLE

GeoIP blocking using xtables-addons is a method to filter or block network traffic based on the geographical location of IP addresses. The xtables-addons package provides extra extensions for iptables, including the xt_geoip module that allows you to match IP addresses against a GeoIP database and apply firewall rules accordingly.

Here's a general overview of how you can set up GeoIP blocking using the xtables-addons and xt_geoip module:

Install xtables-addons:
First, you need to install the xtables-addons package on your system. This package might not be available in all Linux distributions' official repositories, so you might need to install it from a third-party source.

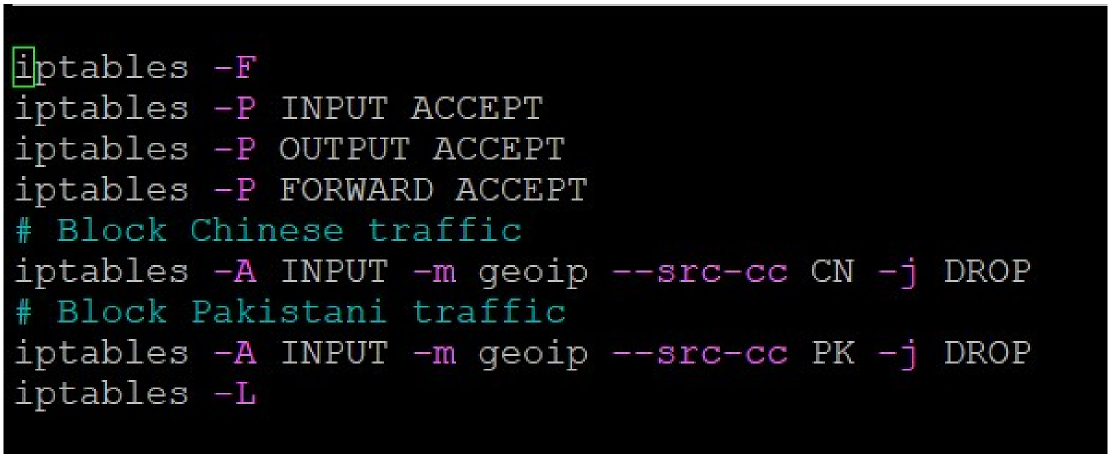
Download GeoIP Database:
You will need a GeoIP database to perform IP address to geographical location lookups. One popular source for such databases is MaxMind. You can download the database files from MaxMind's website.

Compile xt_geoip Module:
After installing xtables-addons, you might need to compile the xt_geoip module using the downloaded GeoIP database files. This step depends on the specific instructions provided by the package and your Linux distribution.

Load xt_geoip Module:
Once compiled, you need to load the xt_geoip module into the kernel:

modprobe xt_geoip
Create Firewall Rules:
Now you can create iptables rules using the xt_geoip module to match IP addresses based on their geographical location. For example, to block traffic from a specific country:

We have imposed some rules on iptables for banning the traffic from a specific country as follows :



```
iptables -F
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
# Block Chinese traffic
iptables -A INPUT -m geoip --src-cc CN -j DROP
# Block Pakistani traffic
iptables -A INPUT -m geoip --src-cc PK -j DROP
iptables -L
```

4.7 FAIL2BAN

Fail2Ban is an open-source intrusion prevention software framework that helps protect computer servers from brute-force attacks and other types of malicious activity. It operates by monitoring log files generated by various services, such as SSH, FTP, Apache, and more, for signs of suspicious or unauthorized behavior. Whenit detects repeated failed login attempts or other predefined patterns that indicate malicious activity, it takes action to block the offending IP addresses from accessing the server.

Here's how Fail2Ban typically works:

Log Monitoring: Fail2Ban continuously monitors log files generated by various services. These logs contain information about login attempts, access requests, and other activities on the server.

Pattern Matching: Fail2Ban uses regular expressions and predefined patterns to identify specific events, such as repeated failed login attempts, HTTP error codes, and more. These patterns are defined in configuration files.

Thresholds: The software tracks the occurrence of events that match the predefined patterns. If the number of occurrences exceeds a specified threshold within a defined time period, Fail2Ban considers it as a potential attack.

Action: Once an attack is detected, Fail2Ban can perform various actions. The most common action is to dynamically update firewall rules (e.g., iptables on Linux) to block traffic from the source IP address. This prevents the attacker from further accessing the server.

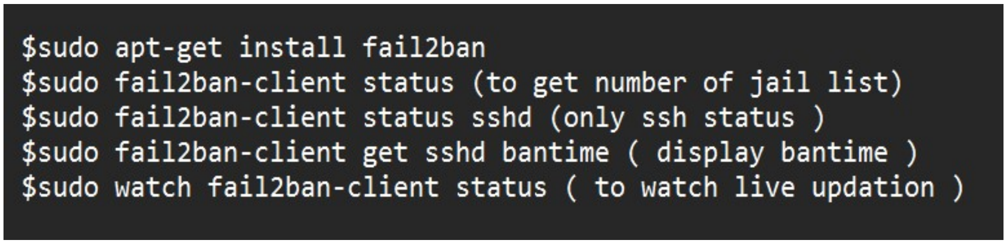
Temporary Bans: Fail2Ban usually imposes temporary bans. After a certain amount of time (configurable), the ban is lifted, allowing legitimate traffic from that IP address to resume.

Notification: Fail2Ban can also send notifications to administrators when it takes action against an IP address. This helps administrators stay informed about potential security threats.

Fail2Ban is highly configurable, and administrators can define their own custom filters, actions, and ban periods based on their server's requirements and the types of attacks they want to mitigate. It's important to notethat while Fail2Ban can be an effective tool in preventing certain types of attacks, it should be used as part of abroader security strategy that includes regular software updates, strong authentication mechanisms, and other security practices.

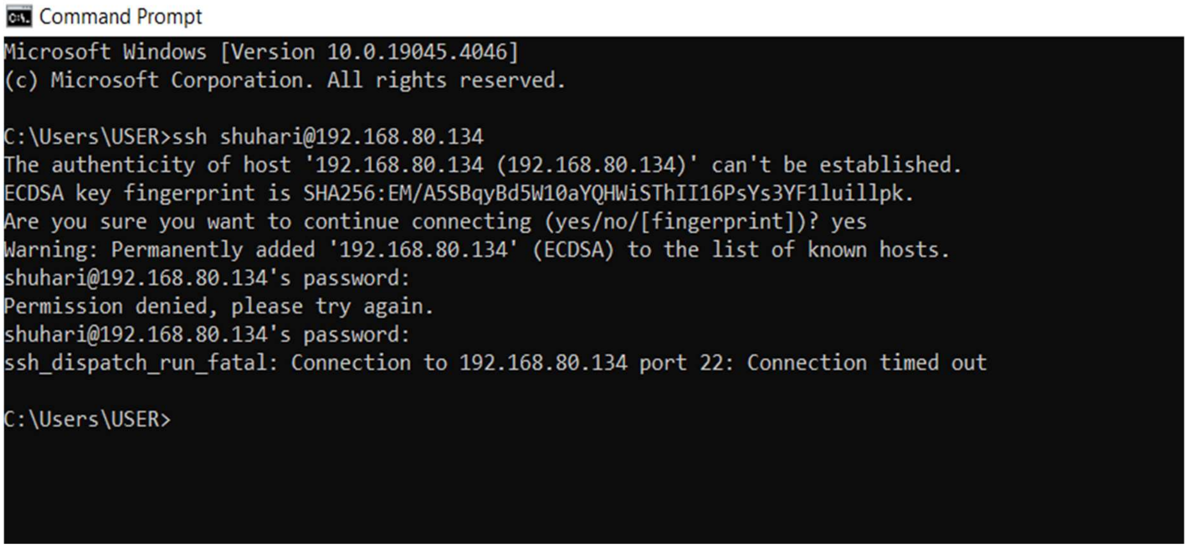
Please keep in mind that my information is based on the state of knowledge as of September 2021, and there might have been updates or changes to Fail2Ban since then. Always refer to the latest documentation and resources for the most up-to-date information.

Following are some commands used after installing Fail2ban on your server :



```
$sudo apt-get install fail2ban
$sudo fail2ban-client status (to get number of jail list)
$sudo fail2ban-client status sshd (only ssh status )
$sudo fail2ban-client get sshd bantime ( display bantime )
$sudo watch fail2ban-client status ( to watch live updation )
```

Lets do brute force on server whose ip address is 192.168.80.134 :

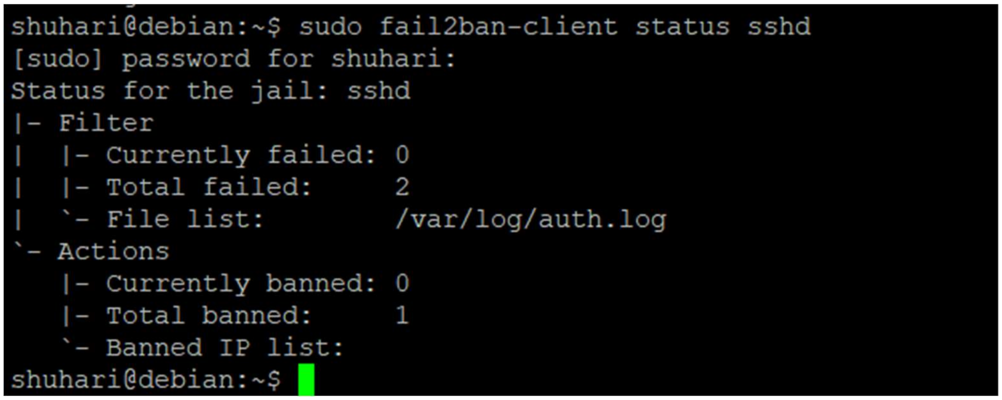


```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>ssh shuhari@192.168.80.134
The authenticity of host '192.168.80.134 (192.168.80.134)' can't be established.
ECDSA key fingerprint is SHA256:EM/A5SBqyBd5W10aYQHWiSThII16PsYs3YF1luillpk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.80.134' (ECDSA) to the list of known hosts.
shuhari@192.168.80.134's password:
Permission denied, please try again.
shuhari@192.168.80.134's password:
ssh_dispatch_run_fatal: Connection to 192.168.80.134 port 22: Connection timed out

C:\Users\USER>
```

The attacker is banned by the Fail2ban lets check :



```
shuhari@debian:~$ sudo fail2ban-client status sshd
[sudo] password for shuhari:
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     2
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
shuhari@debian:~$
```

5. CONCLUSION

"CyberShield Web" project emerges as a dynamic and comprehensive solution to address the growing concerns surrounding web server security. By harnessing the synergy of cutting-edge technologies such as Snort intrusion detection, iptables firewall, and Fail2ban automated threat response, this initiative demonstrates a steadfast commitment to fortifying digital ecosystems against the relentless tide of cyber threats.

In an era where the stakes of cybersecurity have never been higher, the project's multi-layered security paradigm stands as a formidable defense strategy. By combining real-time threat detection, meticulous networktraffic filtering, and the agility of intelligent banning mechanisms, it not only defends against an extensive arrayof malevolent activities but also upholds the pillars of data integrity, user privacy, and uninterrupted business operations.

"CyberShield Web" transcends the conventional and resonates with innovation, poised to empower organizations to navigate the complex landscape of the digital realm confidently. As adversaries continue to evolve, this project signifies a steadfast commitment to safeguarding the foundations of trust and reliability in the online sphere, ensuring that the security of web servers remains an unwavering cornerstone of modern digital enterprise.

6. REFERENCE

1. Books:

Look for books on network security, web server protection, intrusion detection systems, and related topics. Some potential titles might include "Network Security Essentials" by William Stallings and "Intrusion Detection: A Machine Learning Approach" by Siddharth Shekhar.

2. Research Papers and Articles:

Search for scholarly articles and research papers on web server security, intrusion detection, firewall systems,and automation in threat response. You can find these on academic databases like IEEE Xplore, ACM Digital Library, and Google Scholar.

Official Documentation:

The official documentation for Snort, iptables, Fail2ban, and other technologies mentioned in the project can provide technical details, use cases, and best practices.

3. Security Blogs and Websites:

Websites like OWASP (Open Web Application Security Project), SecurityFocus, and KrebsOnSecurity often publish articles and insights about web security trends and practices.

4. White Papers and Case Studies:

Organizations that provide cybersecurity solutions often publish white papers and case studies showcasing theeffectiveness of their products. These resources can provide real-world examples and data.

5. Security Conferences:

Proceedings from security conferences like DEFCON, Black Hat, and RSA Conference often feature presentations and research on web server security and related topics.

6. Online Courses and Tutorials:

Online learning platforms like Coursera, edX, and Udemy offer courses on cybersecurity and web server protection.