ENHANCING WEB APP SECURITY USING MODSECURITY & NGINX

MODSECURITY INTEGRATION: A COMPREHENSIVE GUIDE

BY
MOHAMMAD NEMER

Abstract

This project report investigates the integration of ModSecurity, a robust Web Application Firewall (WAF), with the Nginx server to enhance cybersecurity measures for web applications. The primary objective is to evaluate the effectiveness of ModSecurity, including the implementation of the OWASP Core Rule Set (CRS), in mitigating common web-based attacks, particularly SQL injection and cross-site scripting (XSS). The methodology involves configuring ModSecurity to work seamlessly with Nginx, enabling comprehensive logging features, and testing it against various attack scenarios. Key findings indicate that ModSecurity, coupled with the OWASP CRS and detailed logging, significantly improves the security posture of web applications by effectively detecting and blocking malicious traffic while providing valuable insights through logs. The report concludes with recommendations for further optimization and highlights the importance of continuous security monitoring and detailed logging for proactive threat management.

Figure 1 components

Table of contents

Table of Figures

1.Introduction

1. Overview of the Project and Objectives:

In an era where cyber threats are becoming increasingly sophisticated and frequent, securing web applications is more critical than ever. This project aims to enhance cybersecurity by integrating ModSecurity, a robust Web Application Firewall (WAF), with the Nginx server. The primary objective is to safeguard web applications from a myriad of attacks, particularly those listed in the OWASP Top Ten, such as SQL injection, cross-site scripting (XSS), and denial of service. By implementing this solution, we aim to demonstrate how effective WAFs can be in protecting web applications from both common and advanced threats.

2. Importance of Implementing Defense Security Solutions:

Web applications are often the target of cyber-attacks due to their accessibility and the valuable data they hold. Implementing defense security solutions like ModSecurity WAF is essential for several reasons:

- Protection Against Common Vulnerabilities: ModSecurity helps mitigate risks associated with common vulnerabilities such as SQL injection and XSS, ensuring that the application is more secure against these types of attacks.
- Compliance with Security Standards: Utilizing a WAF aids in achieving compliance with various security standards and regulations, which often mandate protective measures against web application attacks.
- Enhanced Monitoring and Logging: ModSecurity provides extensive logging capabilities, allowing for better monitoring of web traffic and detection of suspicious activities.
- Layered Security Approach: Adding a WAF to the security architecture introduces an additional layer of defense, making it more difficult for attackers to breach the system.

3. Report Structure:

This report is structured to provide a comprehensive overview of the project, detailing each phase from conceptualization to implementation and analysis:

a. Product Presentation: This section introduces ModSecurity WAF, detailing its features, functionalities, and capabilities. It explains why this particular solution was chosen and includes visual aids or diagrams to enhance understanding.

b. Attack Scenarios on Security Solutions: This section describes various attack scenarios relevant to ModSecurity WAF. It explains the methodology of each attack, its potential impact, and includes real-world examples or case studies. Additionally, it details how the attacks were launched, the tools used, and provides evidence through screenshots.

c. Results of the Scenarios: Here, the report analyzes the performance of ModSecurity WAF against each attack scenario. It identifies any vulnerabilities or weaknesses observed during testing, discusses the effectiveness of the solution in mitigating attacks, and provides quantitative metrics or statistics to support the analysis.

d. Conclusion: The conclusion summarizes the key findings and observations from the report. It evaluates the effectiveness and efficiency of ModSecurity WAF, assesses the project's success in achieving its objectives, discusses any challenges faced and how they were overcome, and offers recommendations for further enhancements or areas of improvement.

e. References: This section provides proper citation of all sources referenced in the report.

f. Appendix: The appendix contains detailed instructions for installing, configuring, and deploying ModSecurity WAF, Nginx, and OWASP CRS. It includes screenshots, code snippets, command-line examples, and additional tips and guidance to aid understanding and implementation.

2.Product Presentation

1. ModSecurity:

ModSecurity is an open-source web-based firewall application (WAF), when integrated with the Nginx web server and OWASP CRS, it forms a powerful defense layer against web application attacks. By blocking SQL Injection (SQLi), Remote Code Execution (RCE), Local File Include (LFI), cross-site scripting (XSS), denial of service (DOS), and many other attacks. It acts as an intermediary between the user and the web application, inspecting incoming traffic, detecting malicious patterns, and taking appropriate actions to mitigate potential attacks along with logging feature to inspect any upcoming threats.

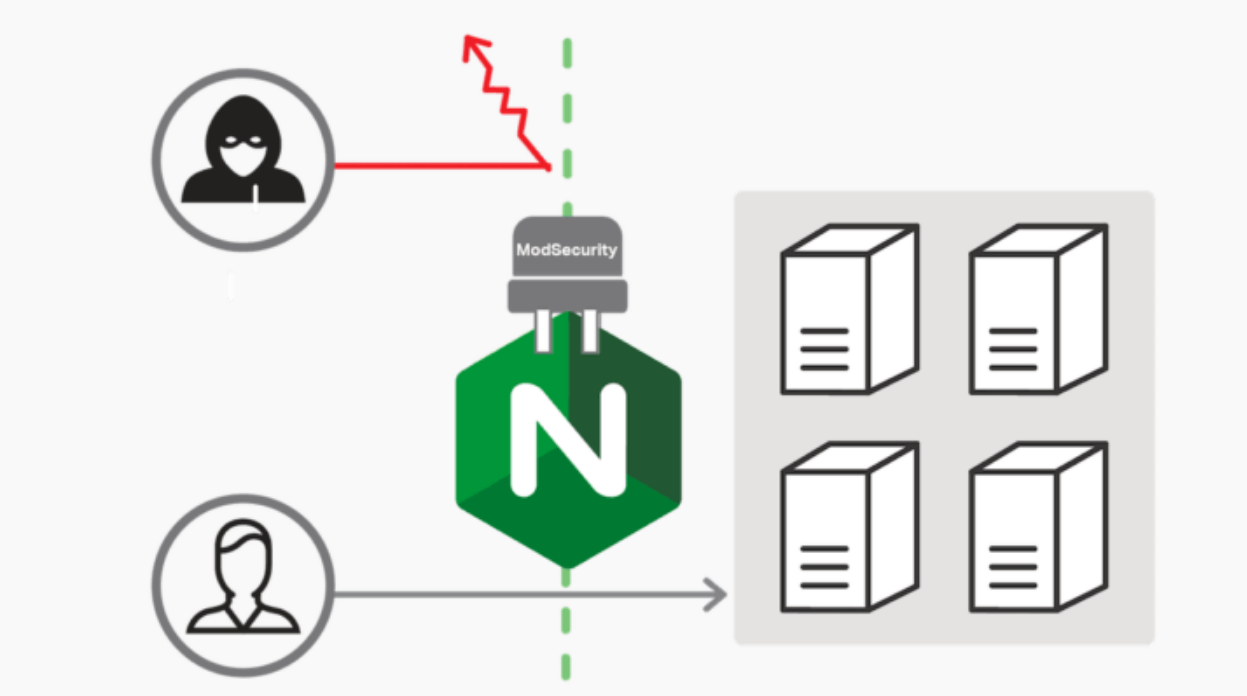


Figure 2 ModSecurity integrated with Nginx

2. Features, Functionalities, and Capabilities:

ModSecurity is renowned for its robust set of features and capabilities that significantly enhance the security of web applications:

- Real-time Monitoring and Logging: ModSecurity provides detailed logging capabilities, allowing for real-time monitoring of web traffic and capturing comprehensive data on each request and response. This aids in the identification and analysis of suspicious activities.
- Request and Response Filtering: With its powerful rule engine, ModSecurity can filter both inbound and outbound traffic based on a wide range of criteria. This includes blocking requests that match predefined patterns of known attacks, such as SQL injection and cross-site scripting (XSS).
- Anomaly Scoring: ModSecurity employs anomaly scoring to detect and block suspicious requests. Each request is assigned a score based on its behavior, and if the score exceeds a certain threshold, the request is blocked.
- Integration with OWASP Core Rule Set (CRS): The OWASP CRS is a set of generic attack detection rules designed to protect web applications from a broad range of vulnerabilities. When used with ModSecurity, these rules provide a robust defense against the most critical security risks identified by the OWASP Top Ten.
- Flexibility and Customization: ModSecurity allows administrators to define custom rules tailored to their specific security needs. This flexibility ensures that the WAF can adapt to unique threat landscapes and application requirements.

3. Why ModSecurity?

The decision to integrate ModSecurity with Nginx was driven by several compelling factors:

- Open Source and Community Support: ModSecurity is an open-source solution with a large, active community of developers and security professionals. This ensures continuous improvement, regular updates, and a wealth of resources for troubleshooting and optimization.
- Compatibility and Performance: Nginx is a high-performance web server known for its scalability and efficiency. Integrating ModSecurity with Nginx leverages the strengths of both platforms, resulting in a solution that is both powerful and efficient in handling high volumes of traffic.
- Comprehensive Protection with OWASP CRS: The inclusion of the OWASP Core Rule Set enhances ModSecurity's ability to detect and block a wide range of attacks. This ensures that the solution provides comprehensive protection against the most common and critical web application vulnerabilities.

below is an illustrative diagram depicting the integration of ModSecurity with Nginx in the security architecture:

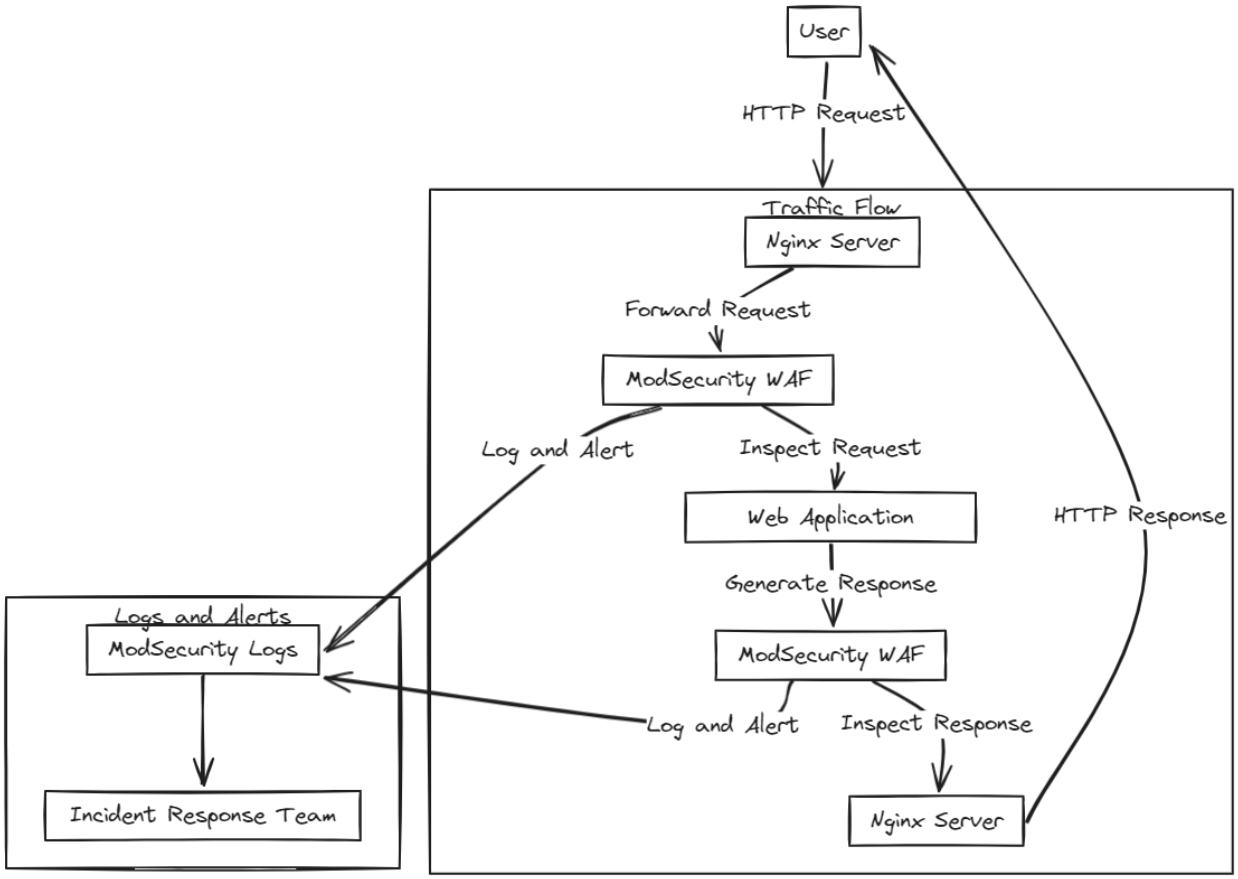


Figure 3 Nginx with ModSecurity workflow

3.Attack Scenarios on Security Solutions

SQL injection attack

1. Methodology:

SQL Injection (SQLi) involves injecting malicious SQL code into a query to manipulate the database. This can lead to unauthorized data access, data modification, or even complete database compromise.

2. Impact:

Successful SQLi attacks can expose sensitive information, disrupt operations, and lead to significant financial and reputational damage.

3. Launch Method:

To demonstrate SQLi, we used a vulnerable web application and injected SQL commands into input fields. For instance, entering ' or 1=1-- into a login field bypasses authentication.

4. Evidence:

As shown below

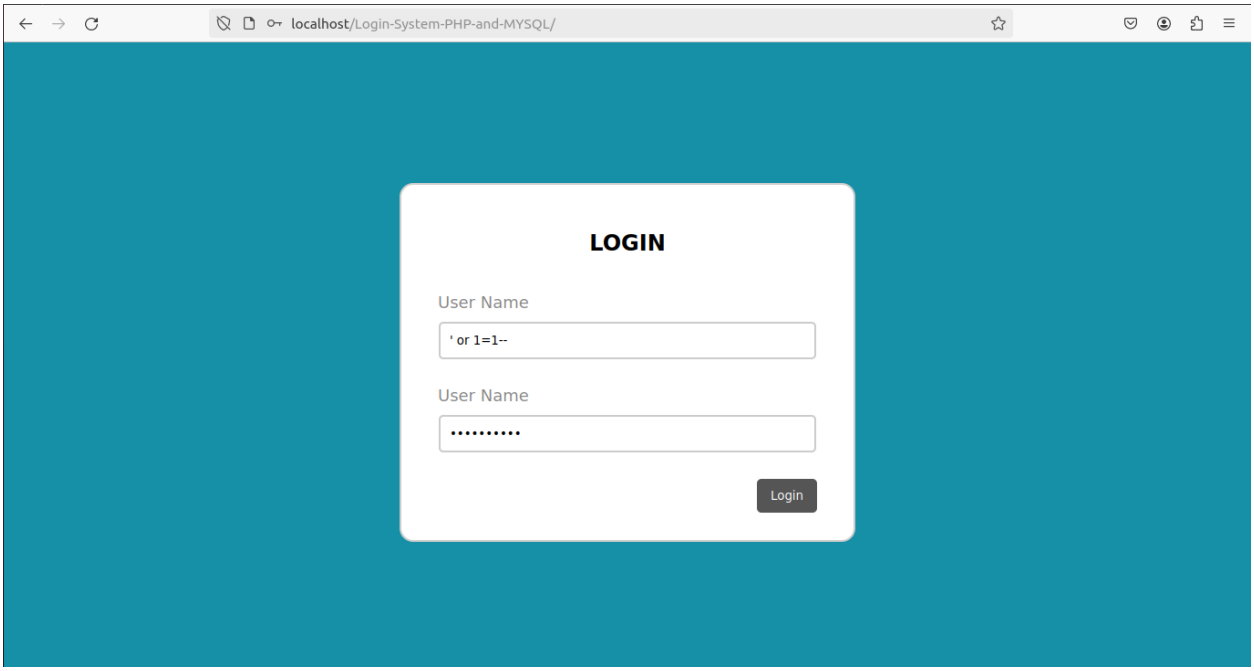


Figure 4 SQLi attack

Figure 5 WAF response to SQL injection attack

Figure 6 SQL Injection logs

URL SQL Injection

1. Methodology:

URL SQL Injection involves appending malicious SQL code to URL parameters. This type of attack targets applications that incorporate user input directly into SQL queries without proper sanitization.

2. Impact:

Similar to traditional SQLi, URL SQL Injection can result in unauthorized data access and database manipulation.

3. Launch Method:

I crafted URLs with SQL injection payloads, such as http://example.com/items?id=1 OR 1=1.

4. Evidence:

As shown below

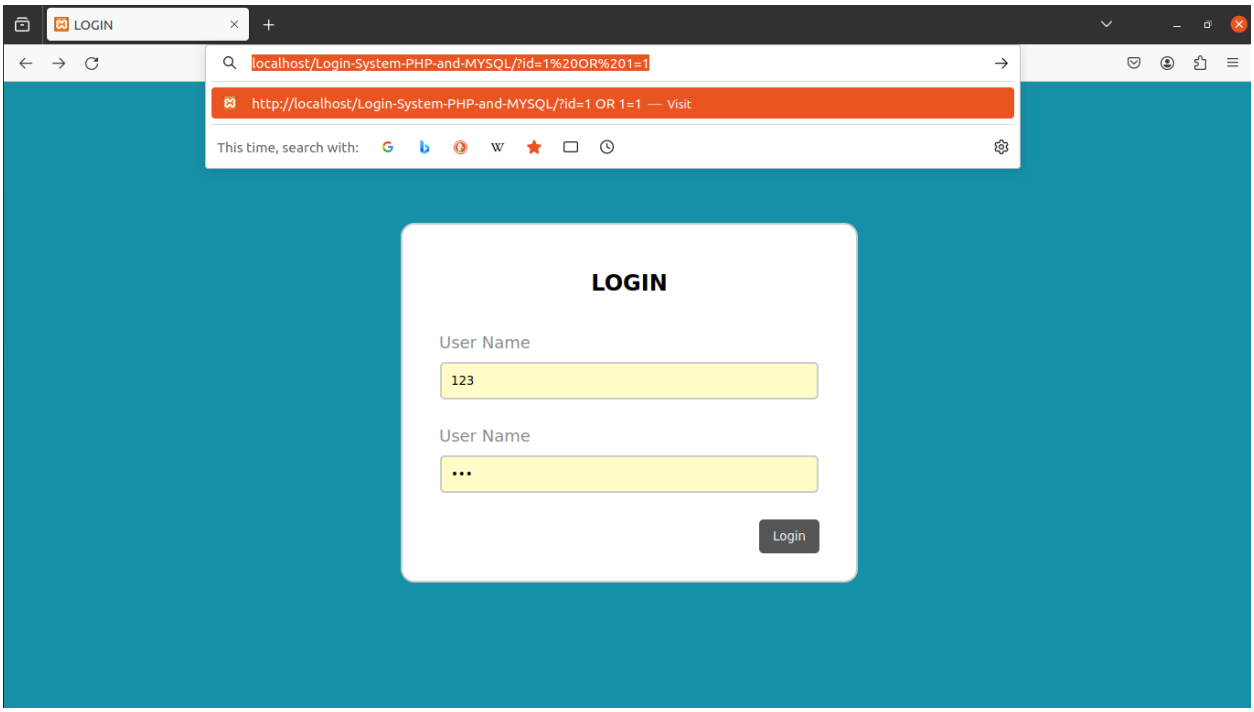


Figure 7 SQL injection attack 2

Figure 8 WAF response to URL SQL injection

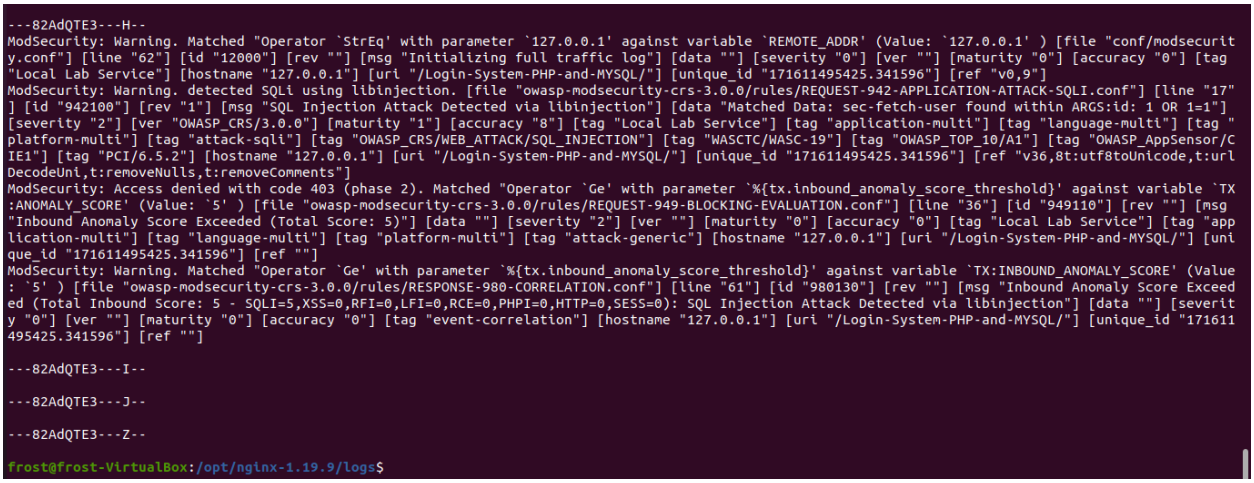


Figure 9 SQLi through url logs

XXS attack

1. Methodology:

XSS attacks inject malicious scripts into web pages viewed by other users. These scripts can steal cookies, session tokens, or perform actions on behalf of the user.

2. Impact:

XSS can lead to session hijacking, data theft, and unauthorized actions on behalf of users.

3. Launch Method:

I embedded JavaScript payloads into input fields, such as <script>alert('XSS');</script>.

4. Evidence:

As shown below

Figure 10 WAF response to XSS attack

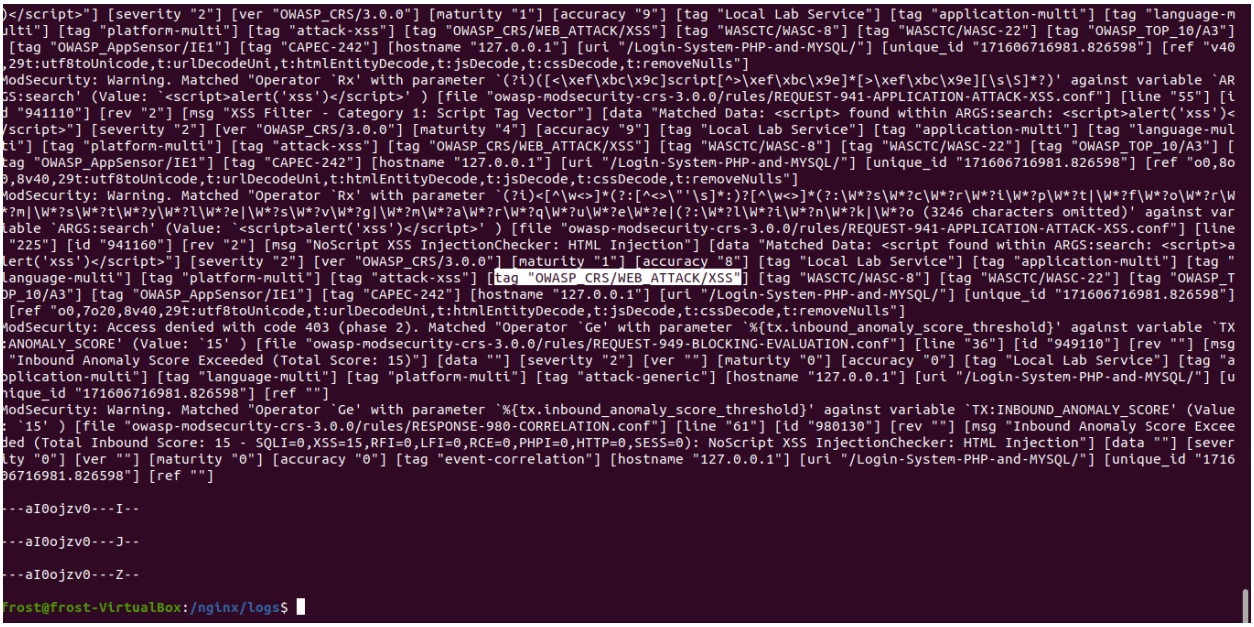


Figure 11 XSS logs

Whitelist and Blacklist Mechanisms

Methodology:

Whitelist and blacklist mechanisms are used to control access to resources based on predefined rules. Whitelists allow only specific entities, while blacklists block known malicious entities.

Impact:

Properly implemented whitelists and blacklists can significantly enhance security by restricting access to trusted entities and blocking malicious ones.

Launch Method:

To test these mechanisms, I attempted to access restricted resources using both allowed and disallowed inputs.

Evidence:

As shown in blacklist rule and in

Figure 12 blacklist rule

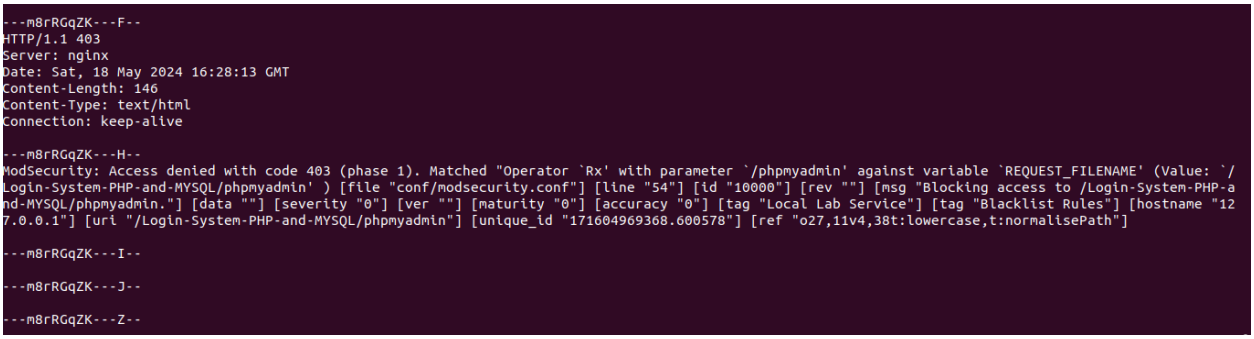


Figure 13 blacklist logs

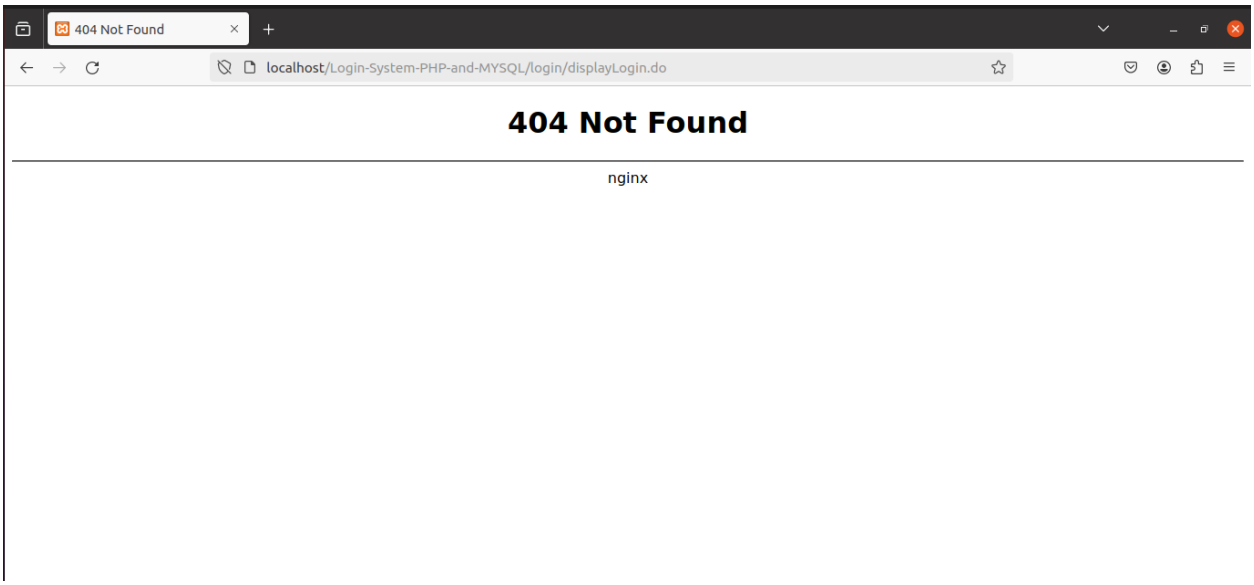


Figure 14 whitelist rule

Figure 15 whitelist logs

4.Results of the Scenarios

Introduction

This section provides a detailed analysis of how our security solution, comprising Nginx with ModSecurity and the OWASP Core Rule Set (CRS), performed against the selected attack scenarios. It identifies any vulnerabilities or weaknesses observed during testing, discusses the effectiveness of the solution in mitigating the attacks, and presents quantitative metrics to support the analysis.

1. SQL Injection

Performance Analysis:

ModSecurity, with the OWASP CRS, effectively detected and blocked SQL injection attempts. The predefined rules in the CRS identified malicious patterns in input fields and URL parameters, preventing unauthorized database access.

Effectiveness:

The solution was highly effective in mitigating SQL injection attacks, ensuring database integrity and preventing data leakage.

Metrics:

See the logs above

2. URL SQL Injection

Performance Analysis:

URL SQL injection attempts were also effectively intercepted and blocked by ModSecurity with the OWASP CRS. The rules efficiently parsed and sanitized URL parameters to neutralize malicious payloads.

Effectiveness:

The solution demonstrated robust protection against URL SQL injection attacks, maintaining secure access controls and data protection.

Metrics:

See the logs above

3. Cross-Site Scripting (XSS)

Performance Analysis:

The security solution effectively blocked XSS attempts by filtering out malicious scripts from user input fields. The CRS includes comprehensive XSS protection rules that prevent the execution of harmful scripts.

Effectiveness:

The solution was effective in mitigating XSS attacks, protecting user sessions and preventing unauthorized actions.

Metrics:

See the logs above

4. Whitelist and Blacklist Mechanisms

Performance Analysis:

Whitelist and blacklist mechanisms were tested by attempting to access restricted resources with both allowed and disallowed inputs. The security solution correctly allowed or blocked access based on the predefined rules.

Effectiveness:

The solution effectively enforced access control policies, enhancing overall security by restricting access to trusted entities and blocking known malicious ones.

Metrics:

See the logs blacklist logs and whitelist logs

5.Conclusion

Summary of Key Findings and Observations

The report investigated the integration of ModSecurity, a robust Web Application Firewall (WAF), with the Nginx server to enhance cybersecurity measures for web applications. The key findings indicate that ModSecurity, coupled with the OWASP Core Rule Set (CRS), effectively mitigates common web-based attacks such as SQL injection (SQLi) and cross-site scripting (XSS). The testing of various attack scenarios, including URL SQL injection and whitelist/blacklist mechanisms, demonstrated the solution's robustness in detecting and blocking malicious traffic.

Evaluation of the Effectiveness and Efficiency of the Chosen Defense Security Solution

The chosen security solution, comprising Nginx with ModSecurity and the OWASP CRS, performed exceptionally well against the selected attack scenarios. It efficiently detected and blocked SQL injection and XSS attacks, maintaining database integrity and protecting user sessions. The whitelist and blacklist mechanisms accurately controlled access based on predefined rules, further enhancing security. The comprehensive logging features provided valuable insights into suspicious activities, aiding in proactive threat management.

Overall Assessment of the Project's Success in Achieving Its Objectives

The project's primary objective of evaluating the effectiveness of ModSecurity in mitigating common web-based attacks was successfully achieved. The integration of ModSecurity with Nginx provided a powerful defense layer, significantly improving the security posture of web applications. However, some difficulties were encountered during the project, such as configuring ModSecurity to work seamlessly with Nginx and ensuring comprehensive logging features. Some minor issues, such as optimizing performance under high traffic loads, remain and require further attention.

Recommendations for Further Enhancements or Areas of Improvement

- Performance Optimization: Further optimization is needed to handle high volumes of traffic efficiently. This includes fine-tuning ModSecurity rules and enhancing server performance.
- Continuous Monitoring: Implement continuous security monitoring to promptly detect and respond to emerging threats.
- Rule Customization: Develop and implement custom rules tailored to specific security needs, ensuring adaptability to unique threat landscapes.
- Regular Updates: Ensure ModSecurity and OWASP CRS are regularly updated to incorporate the latest security patches and improvements.
- User Training: Conduct regular training sessions for administrators and users to stay informed about the latest security practices and threat trends.

References

Netnea's Nginx Tutorials:

- Tutorial 1: "Compiling Nginx," [Online]. Available: https://www.netnea.com/cms/nginx-tutorial-1_compiling-nginx/
- Tutorial 2: "Minimal Nginx Configuration," [Online]. Available: https://www.netnea.com/cms/nginx-tutorial-2_minimal-nginx-configuration/
- Tutorial 6: "Embedding ModSecurity," [Online]. Available: https://www.netnea.com/cms/nginx-tutorial-6_embedding-modsecurity/

OWASP ModSecurity Core Rule Set:

- OWASP Foundation, "OWASP ModSecurity Core Rule Set," [Online]. Available: https://coreruleset.org/docs/deployment/engine_integration_options/

Medium Article on Nginx:

- Dissanayake, Oct 20, 2021, "What is Nginx?", Medium. Available: https://medium.com/geekculture/what-is-nginx-2edfdad3722b

Namecheap Knowledgebase Article on ModSecurity:

- Namecheap Support, "What is ModSecurity and Why Do We Need It?," Namecheap Knowledgebase. Available: https://www.namecheap.com/support/knowledgebase/article.aspx/9542/22/what-is-modsecurity-and-why-do-we-need-it/

Wikipedia Page on ModSecurity:

- Wikipedia contributors. (Year, Month Day). "ModSecurity," Wikipedia, The Free Encyclopedia. Available: https://en.wikipedia.org/wiki/ModSecurity

OWASP ModSecurity GitHub Repository:

- OWASP ModSecurity, "ModSecurity GitHub Repository," [Online]. Available: https://github.com/owasp-modsecurity/ModSecurity

Nginx Official Download Page:

- Nginx, "Download Nginx," [Online]. Available: http://nginx.org/download

Appendix

1. Compiling NGINX

Step 1: Preparing the directory tree for the source code

```
frost@frost-VirtualBox:~$ sudo mkdir /usr/src/nginx
frost@frost-VirtualBox:~$ sudo chown `whoami` /usr/src/nginx
frost@frost-VirtualBox:~$ cd /usr/src/nginx
frost@frost-VirtualBox:/usr/src/nginx$ S
```

Step 2: Meeting the requirements for NGINX

Install these packages:

- build-essential
- binutils
- gcc
- libpcre3-dev
- libssl-dev
- zlib1g-devS

```
frost@frost-VirtualBox:/usr/src/nginx$ sudo apt-get install -y build-essential binutils gcc libpcre3-dev libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:9.3.0-1ubuntu2).
gcc set to manually installed.
binutils is already the newest version (2.34-6ubuntu1.9).
binutils set to manually installed.
build-essential is already the newest version (12.8ubuntu1.1).
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libpcre16-3 libpcre3-dev libpcre32-3 libpcrecpp0v5 libssl-dev
```

Step 3: Downloading the source code and verifying the integrity of the code

Download the code of nginx along with its key to insure the integrity of the code

check the signature on the code based on a publicly available key of one of the lead developers.

```
frost@frost-VirtualBox:/usr/src/nginx$ wget http://nginx.org/keys/mdounin.key
--2024-05-05 00:06:42--  http://nginx.org/keys/mdounin.key
Resolving nginx.org (nginx.org)... 3.125.197.172, 52.58.199.22, 2a05:d014:5c0:2601::6, ...
Connecting to nginx.org (nginx.org)|3.125.197.172|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1912 (1.9K) [application/octet-stream]
Saving to: ‘mdounin.key’

mdounin.key                     100%[=============================================================>]   1.87K  --.-KB/s    in 0s

2024-05-05 00:06:42 (255 MB/s) - ‘mdounin.key’ saved [1912/1912]

frost@frost-VirtualBox:/usr/src/nginx$ gpg --import mdounin.key
gpg: key 520A9993A1C052F8: 2 signatures not checked due to missing keys
gpg: key 520A9993A1C052F8: public key "Maxim Dounin <mdounin@mdounin.ru>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
frost@frost-VirtualBox:/usr/src/nginx$
```

The long id is: 520A9993A1C052F8, let’s check the signature

```
frost@frost-VirtualBox:/usr/src/nginx$ wget https://nginx.org/download/nginx-1.19.9.tar.gz.asc
--2024-05-05 00:21:51--  https://nginx.org/download/nginx-1.19.9.tar.gz.asc
Resolving nginx.org (nginx.org)... 3.125.197.172, 52.58.199.22, 2a05:d014:5c0:2600::6, ...
Connecting to nginx.org (nginx.org)|3.125.197.172|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 455 [application/octet-stream]
Saving to: ‘nginx-1.19.9.tar.gz.asc’

nginx-1.19.9.tar.gz.asc           100%[=======================================================>]     455  --.-KB/s    in 0s

2024-05-05 00:21:51 (73.6 MB/s) - ‘nginx-1.19.9.tar.gz.asc’ saved [455/455]

frost@frost-VirtualBox:/usr/src/nginx$ wget https://nginx.org/download/nginx-1.19.9.tar.gz
--2024-05-05 00:21:57--  https://nginx.org/download/nginx-1.19.9.tar.gz
Resolving nginx.org (nginx.org)... 52.58.199.22, 3.125.197.172, 2a05:d014:5c0:2601::6, ...
Connecting to nginx.org (nginx.org)|52.58.199.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1060580 (1.0M) [application/octet-stream]
Saving to: ‘nginx-1.19.9.tar.gz’

nginx-1.19.9.tar.gz               100%[=======================================================>]   1.01M   986KB/s    in 1.1s

2024-05-05 00:21:58 (986 KB/s) - ‘nginx-1.19.9.tar.gz’ saved [1060580/1060580]
```

```
frost@frost-VirtualBox:/usr/src/nginx$ gpg --trusted-key 520A9993A1C052F8 --verify nginx-1.19.9.tar.gz.asc nginx-1.19.9.tar.gz
gpg: Signature made 30 م 05:48:50 2021 آذار, EEST
gpg:                using RSA key 520A9993A1C052F8
gpg: key 520A9993A1C052F8 marked as ultimately trusted
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: Good signature from "Maxim Dounin <mdounin@mdounin.ru>" [ultimate]
```

Signature is checked successfully using the key

Step 4: Unpacking and configuring the compiler

```
frost@frost-VirtualBox:/usr/src/nginx$ tar -xvzf nginx-1.19.9.tar.gz
nginx-1.19.9/
nginx-1.19.9/auto/
nginx-1.19.9/conf/
nginx-1.19.9/contrib/
nginx-1.19.9/src/
nginx-1.19.9/configure
nginx-1.19.9/LICENSE
nginx-1.19.9/README
nginx-1.19.9/html/
nginx-1.19.9/man/
nginx-1.19.9/CHANGES.ru
```

configure compiler, install it on custom directory,

--prefix=/opt/nginx-1.19.9: directory where to install compiler, ***to keep our work on single branch***

--with-http_ssl_module: This option enables the SSL (Secure Sockets Layer) module in Nginx, allowing the server to support HTTPS connections.

--with-threads: Enabling threads support allows Nginx to handle multiple connections concurrently using threads. To deal with a large number of simultaneous requests.

--with-file-aio: This option enables asynchronous I/O (Input/Output) operations for file handling in Nginx. Asynchronous I/O allows Nginx to perform file operations without blocking the main process, which can enhance performance and scalability.

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ ./configure --prefix=/opt/nginx-1.19.9 --with-http_ssl_module --with-threads --with
-file-aio
checking for OS
 + Linux 5.15.0-105-generic x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found
checking for unistd.h ... found
```

We get the following error simply fix that by

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo apt-get install zlib1g-dev
[sudo] password for frost:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  zlib1g-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 155 kB of archives.
```

Rerun "./configure --prefix=/opt/nginx-1.13.9 --with-http_ssl_module --with-threads --with-file-aio"

If you get the following output then you're in mint condition

```
Configuration summary
  + using threads
  + using system PCRE library
  + using system OpenSSL library
  + using system zlib library

  nginx path prefix: "/opt/nginx-1.19.9"
  nginx binary file: "/opt/nginx-1.19.9/sbin/nginx"
  nginx modules path: "/opt/nginx-1.19.9/modules"
  nginx configuration prefix: "/opt/nginx-1.19.9/conf"
  nginx configuration file: "/opt/nginx-1.19.9/conf/nginx.conf"
  nginx pid file: "/opt/nginx-1.19.9/logs/nginx.pid"
  nginx error log file: "/opt/nginx-1.19.9/logs/error.log"
  nginx http access log file: "/opt/nginx-1.19.9/logs/access.log"
  nginx http client request body temporary files: "client_body_temp"
  nginx http proxy temporary files: "proxy_temp"
  nginx http fastcgi temporary files: "fastcgi_temp"
  nginx http uwsgi temporary files: "uwsgi_temp"
  nginx http scgi temporary files: "scgi_temp"
```

Step 5: Compile the nginx server

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ make
make -f objs/Makefile
make[1]: Entering directory '/usr/src/nginx/nginx-1.19.9'
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I src/event -I src/event/modules -I src/os/unix
 -I objs \
        -o objs/src/core/nginx.o \
        src/core/nginx.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I src/event -I src/event/modules -I src/os/unix
 -I objs \
        -o objs/src/core/ngx_log.o \
        src/core/ngx_log.c
```

Note: this may take some time

Step 6: Installing

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo make install
make -f objs/Makefile install
make[1]: Entering directory '/usr/src/nginx/nginx-1.19.9'
test -d '/opt/nginx-1.19.9' || mkdir -p '/opt/nginx-1.19.9'
test -d '/opt/nginx-1.19.9/sbin' \
        || mkdir -p '/opt/nginx-1.19.9/sbin'
test ! -f '/opt/nginx-1.19.9/sbin/nginx' \
        || mv '/opt/nginx-1.19.9/sbin/nginx' \
                '/opt/nginx-1.19.9/sbin/nginx.old'
cp objs/nginx '/opt/nginx-1.19.9/sbin/nginx'
test -d '/opt/nginx-1.19.9/conf' \
        || mkdir -p '/opt/nginx-1.19.9/conf'
cp conf/koi-win '/opt/nginx-1.19.9/conf'
cp conf/koi-utf '/opt/nginx-1.19.9/conf'
```

Since working with configuration files is sensitive, we change the owner of /opt/nginx-1.19.9 and everything inside it to the user who executes the command with root

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo chown -R `whoami` /opt/nginx-1.19.9
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ S
```

To ensure dynamicity of the work and to try different versions, modules, and patches. we make it easy by creating a soft link from /nginx to the current NGINX web server , now we can run nginx server from one place

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo ln -s /opt/nginx-1.19.9 /nginx
[sudo] password for frost:
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo chown `whoami` --no-dereference /nginx
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ cd /nginx
frost@frost-VirtualBox:/nginx$
```

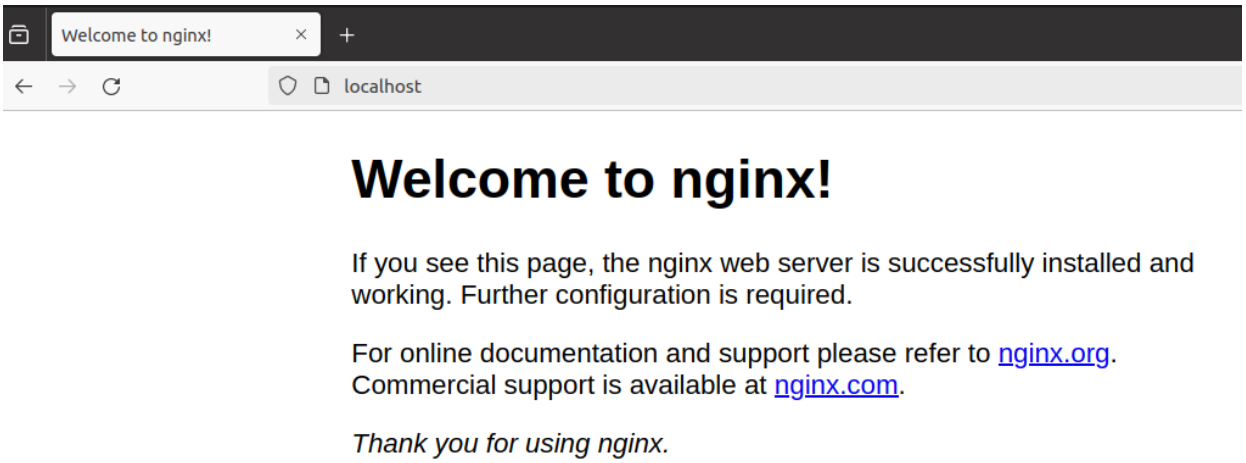
Step 7: Starting nginx

Starting nginx server is a super user accessibility

Let’s try to start nginx server

```
frost@frost-VirtualBox:/nginx$ sudo ./sbin/nginx
frost@frost-VirtualBox:/nginx$  ps -awuq $(cat logs/nginx.pid)
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       11516  0.0  0.0   8360   784 ?        Ss   13:32   0:00 nginx: master process ./sbin/nginx
frost@frost-VirtualBox:/nginx$
```

Check for details about the running nginx server



Our nginx server is running like a well-oiled machine

Now stop the server

```
frost@frost-VirtualBox:/nginx$ sudo ./sbin/nginx -s stop
frost@frost-VirtualBox:/nginx$
```

Step 8: Inspecting the server's binary and related libraries

```
frost@frost-VirtualBox:/nginx$ sudo ./sbin/nginx -V
[sudo] password for frost:
nginx version: nginx/1.19.9
built by gcc 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)
built with OpenSSL 1.1.1f  31 Mar 2020
T Ubuntu Software t enabled
configure arguments: --prefix=/opt/nginx-1.19.9 --with-http_ssl_module --with-threads --with-file-aio
frost@frost-VirtualBox:/nginx$ ldd sbin/nginx
        linux-vdso.so.1 (0x00007ffe50bf7000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f3fef1c5000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f3fef1a2000)
        libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007f3fef167000)
        libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f3fef0f4000)
        libssl.so.1.1 => /lib/x86_64-linux-gnu/libssl.so.1.1 (0x00007f3fef061000)
        libcrypto.so.1.1 => /lib/x86_64-linux-gnu/libcrypto.so.1.1 (0x00007f3feed8a000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f3feed6c000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3feeb7a000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f3fef2e3000)
frost@frost-VirtualBox:/nginx$
```

2. NGINX Configuration

Step 1: Creating a minimal configuration

Modify the nginx.conf file, and replace the default configuration with the following configuration

```
daemon            off;
worker_processes  2;
user              www-data;

events {
    use           epoll;
    worker_connections  128;
}

error_log         logs/error.log info;

http {
    server_tokens off;
    include       mime.types;
    charset       utf-8;

    access_log    logs/access.log  combined;

    server {
        server_name   localhost;
        listen        127.0.0.1:80;

        error_page    500 502 503 504  /50x.html;

        location      / {
            root      html;
        }

    }

}
:wq
```

Step 2: Starting and testing the server

Once our configuration is ready, we start the nginx server for testing

NOTE: the server started in the foreground and not as a daemon

```
frost@frost-VirtualBox:/nginx$ sudo ./sbin/nginx
```

Take a look

```
frost@frost-VirtualBox:/opt/nginx-1.19.9$ ps -awuq $(cat logs/nginx.pid)
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       13552  0.0  0.2   8360  5304 pts/0    S+   15:41   0:00 nginx: master process ./sbin/nginx
frost@frost-VirtualBox:/opt/nginx-1.19.9$
```

Communicate with the server using curl

```
frost@frost-VirtualBox:~$
frost@frost-VirtualBox:~$
frost@frost-VirtualBox:~$ curl http://localhost/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
frost@frost-VirtualBox:~$
```

The server responded to our request

```
frost@frost-VirtualBox:~$ curl --verbose http://localhost/index.html
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /index.html HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx
< Date: Sun, 05 May 2024 14:04:53 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 612
< Last-Modified: Sat, 04 May 2024 21:44:12 GMT
< Connection: keep-alive
< ETag: "6636ac2c-264"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

To check simple benchmark for the server performance

```
frost@frost-VirtualBox:~$ siege --concurrent 100 --reps 10 http://localhost/index.html
New configuration template added to /home/frost/.siege
Run siege -C to view the current settings in that file
** SIEGE 4.0.4
** Preparing 100 concurrent users for battle.
The server is now under siege...
Transactions:                   1000 hits
Availability:                 100.00 %
Elapsed time:                   0.36 secs
Data transferred:               0.58 MB
Response time:                  0.03 secs
Transaction rate:            2777.78 trans/sec
Throughput:                     1.62 MB/sec
Concurrency:                   82.50
Successful transactions:        1000
Failed transactions:               0
Longest transaction:            0.22
Shortest transaction:           0.00
```

The siege command was executed with a concurrency level of 100 users making 10 repetitions of requests. The output shows that 1000 hits were made successfully, indicating an availability of 100%. The total elapsed time for these transactions was 0.36 seconds, during which 0.58 MB of data was transferred. The response time was very low at 0.03 seconds, showcasing efficient server performance under load. The transaction rate was 2777.78 transactions per second, with a throughput of 1.62 MB per second. The concurrency level during this test reached 82.50, demonstrating the system's ability to handle multiple concurrent requests. Importantly, there were no failed transactions, indicating a stable and robust server response. The longest transaction took 0.22 seconds, while the shortest was instantaneous at 0.00 seconds, reflecting consistent and fast response times across the board.

3. Embedding ModSecurity

Step 1: Downloading the source code and verifying the checksum

```
frost@frost-VirtualBox:~$ sudo mkdir /usr/src/modsecurity
frost@frost-VirtualBox:~$ sudo chown `whoami` /usr/src/modsecurity
frost@frost-VirtualBox:~$ cd /usr/src/modsecurity
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ wget https://github.com/SpiderLabs/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0
.tar.gz
--2024-05-05 17:59:30--  https://github.com/SpiderLabs/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/owasp-modsecurity/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz [following]
--2024-05-05 17:59:32--  https://github.com/owasp-modsecurity/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/1320594/c613977c-e101-11e7-9790-9efbef669024?X-A
mz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240505%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240505T1459
32Z&X-Amz-Expires=300&X-Amz-Signature=e88250ee6ed7b51005fec49883ebeea0e7dfcdf5071a5605b9f75d825c61f7d4&X-Amz-SignedHeaders=host&actor_i
d=0&key_id=0&repo_id=1320594&response-content-disposition=attachment%3B%20filename%3Dmodsecurity-v3.0.0.tar.gz&response-content-type=ap
plication%2Foctet-stream [following]
--2024-05-05 17:59:33--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/1320594/c613977c-e101-11e7-9790-9
efbef669024?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240505%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Dat
e=20240505T145932Z&X-Amz-Expires=300&X-Amz-Signature=e88250ee6ed7b51005fec49883ebeea0e7dfcdf5071a5605b9f75d825c61f7d4&X-Amz-SignedHeade
rs=host&actor_id=0&key_id=0&repo_id=1320594&response-content-disposition=attachment%3B%20filename%3Dmodsecurity-v3.0.0.tar.gz&response-
content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2921625 (2.8M) [application/octet-stream]
Saving to: 'modsecurity-v3.0.0.tar.gz'
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$  wget https://github.com/SpiderLabs/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.
0.tar.gz.sha256
--2024-05-05 18:00:52--  https://github.com/SpiderLabs/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz.sha256
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/owasp-modsecurity/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz.sha256 [following]
--2024-05-05 18:00:52--  https://github.com/owasp-modsecurity/ModSecurity/releases/download/v3.0.0/modsecurity-v3.0.0.tar.gz.sha256
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/1320594/c6497ebe-e101-11e7-8339-0b26e635b9d1?X-A
mz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240505%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240505T1500
52Z&X-Amz-Expires=300&X-Amz-Signature=649db5696fac0d55e23cb9d2d86abd96bfe61889ad3d826bd8e76596db69248c&X-Amz-SignedHeaders=host&actor_i
d=0&key_id=0&repo_id=1320594&response-content-disposition=attachment%3B%20filename%3Dmodsecurity-v3.0.0.tar.gz.sha256&response-content-
type=application%2Foctet-stream [following]
--2024-05-05 18:00:53--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/1320594/c6497ebe-e101-11e7-8339-0
b26e635b9d1?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240505%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Dat
e=20240505T150052Z&X-Amz-Expires=300&X-Amz-Signature=649db5696fac0d55e23cb9d2d86abd96bfe61889ad3d826bd8e76596db69248c&X-Amz-SignedHeade
rs=host&actor_id=0&key_id=0&repo_id=1320594&response-content-disposition=attachment%3B%20filename%3Dmodsecurity-v3.0.0.tar.gz.sha256&re
sponse-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 92 [application/octet-stream]
Saving to: 'modsecurity-v3.0.0.tar.gz.sha256'

modsecurity-v3.0.0.tar.gz.sha256  100%[=============================================================>]      92  --.-KB/s    in 0s

2024-05-05 18:00:53 (1.46 MB/s) - 'modsecurity-v3.0.0.tar.gz.sha256' saved [92/92]
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ sha256sum --check modsecurity-v3.0.0.tar.gz.sha256
modsecurity-v3.0.0.tar.gz: OK
frost@frost-VirtualBox:/usr/src/modsecurity$
```

Step 2: Unpacking and configuring the compiler

Insure the following packages are installed

- libxml2
- libxml2-dev

- libexpat1-dev
- libpcre3-dev
- libpcre++-dev
- libyajl-dev
- libgeoip-dev
- libcurl4-gnutls-dev
- dh-autoreconf

```
frost@frost-VirtualBox:~$ sudo apt-get install -y libxml2 libxml2-dev libexpat1-dev libpcre3-dev libpcre++-dev libyajl-dev libgeoip-dev
 libcurl4-gnutls-dev dh-autoreconf
[sudo] password for frost:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libpcre3-dev is already the newest version (2:8.39-12ubuntu0.1).
libxml2 is already the newest version (2.9.10+dfsg-5ubuntu0.20.04.7).
libxml2 set to manually installed.
The following additional packages will be installed:
  autoconf automake autopoint autotools-dev debhelper dh-strip-nondeterminism dwz geoip-bin geoip-database gettext icu-devtools
  intltool-debian libarchive-cpio-perl libarchive-zip-perl libcroco3 libdebhelper-perl libfile-stripnondeterminism-perl libgeoip1
  libicu-dev libltdl-dev libmail-sendmail-perl libnetaddr-ip-perl libpcre++0v5 libsigsegv2 libsocket6-perl libsub-override-perl
  libsys-hostname-long-perl libtool libyajl2 m4 po-debconf
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc dh-make gettext-doc libasprintf-dev libgettextpo-dev libcurl4-doc libgnutls28-dev
  libidn11-dev libkrb5-dev libldap2-dev librtmp-dev libssh2-1-dev icu-doc libtool-doc gfortran | fortran95-compiler gcj-jdk m4-doc
  libmail-box-perl
The following NEW packages will be installed:
  autoconf automake autopoint autotools-dev debhelper dh-autoreconf dh-strip-nondeterminism dwz geoip-bin geoip-database gettext
  icu-devtools intltool-debian libarchive-cpio-perl libarchive-zip-perl libcroco3 libcurl4-gnutls-dev libdebhelper-perl libexpat1-dev
  libfile-stripnondeterminism-perl libgeoip-dev libgeoip1 libicu-dev libltdl-dev libmail-sendmail-perl libnetaddr-ip-perl
  libpcre++-dev libpcre++0v5 libsigsegv2 libsocket6-perl libsub-override-perl libsys-hostname-long-perl libtool libxml2-dev
  libyajl-dev libyajl2 m4 po-debconf
```

Unpack Modsecurity

```
frost@frost-VirtualBox:~$ cd /usr/src/modsecurity
frost@frost-VirtualBox:/usr/src/modsecurity$ tar -xvzf modsecurity-v3.0.0.tar.gz
modsecurity-v3.0.0/
modsecurity-v3.0.0/LICENSE
modsecurity-v3.0.0/config.guess
modsecurity-v3.0.0/doc/
modsecurity-v3.0.0/doc/doxygen.cfg
modsecurity-v3.0.0/doc/Makefile.am
modsecurity-v3.0.0/doc/Makefile.in
modsecurity-v3.0.0/doc/.empty
modsecurity-v3.0.0/doc/ms-doxygen-logo.png
modsecurity-v3.0.0/compile
modsecurity-v3.0.0/others/
modsecurity-v3.0.0/others/modsec.png
modsecurity-v3.0.0/others/Makefile.am
modsecurity-v3.0.0/others/Makefile.in
modsecurity-v3.0.0/others/mbedtls/
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ cd modsecurity-v3.0.0/
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$ ./configure --prefix=/opt/modsecurity-3.0.0 --enable-mutex-on-pm
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for g++... g++
checking whether the C++ compiler works... yes
checking for C++ compiler default output file name... a.out
checking for suffix of executables...
```

Everything is fine

```
ModSecurity - v3.0.0 for Linux

 Mandatory dependencies
   + libInjection                                ....v3.0.0
   + SecLang tests                               ....c1cd668a

 Optional dependencies
   + GeoIP                                       ....found v1.6.12
      -lGeoIP, -I/usr/include/
   + LibCURL                                     ....found v7.68.0
      -lcurl,  -DWITH_CURL_SSLVERSION_TLSv1_2 -DWITH_CURL
   + YAJL                                        ....found v2.1.0
      -lyajl, -DWITH_YAJL -I/usr/include/yajl
   + LMDB                                        ....not found
   + LibXML2                                     ....found v2.9.10
      -lxml2, -I/usr/include/libxml2 -DWITH_LIBXML2
   + SSDEEP                                      ....not found
   + LUA                                         ....not found

 Other Options
   + Test Utilities                              ....enabled
   + SecDebugLog                                 ....enabled
   + afl fuzzer                                  ....disabled
   + library examples                            ....enabled
   + Building parser                             ....disabled
   + Treating pm operations as critical section  ....enabled
```

Step 3: Compiling and installing standalone ModSecurity

Note: This will take some time

```
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$ make
Making all in others
make[1]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/others'
depbase=`echo libinjection/src/libinjection_html5.lo | sed 's|[^/]*$|.deps/&|;s|\.lo$||'`;\
/bin/bash ../libtool  --tag=CC   --mode=compile gcc -DHAVE_CONFIG_H -I. -I../src     -g -O2 -MT libinjection/src
/libinjection_html5.lo -MD -MP -MF $depbase.Tpo -c -o libinjection/src/libinjection_html5.lo libinjection/src/li
binjection_html5.c &&\
mv -f $depbase.Tpo $depbase.Plo
libtool: compile:  gcc -DHAVE_CONFIG_H -I. -I../src -g -O2 -MT libinjection/src/libinjection_html5.lo -MD -MP -M
F libinjection/src/.deps/libinjection_html5.Tpo -c libinjection/src/libinjection_html5.c  -fPIC -DPIC -o libinje
ction/src/.libs/libinjection_html5.o
libtool: compile:  gcc -DHAVE_CONFIG_H -I. -I../src -g -O2 -MT libinjection/src/libinjection_html5.lo -MD -MP -M
F libinjection/src/.deps/libinjection_html5.Tpo -c libinjection/src/libinjection_html5.c -o libinjection/src/lib
injection_html5.o >/dev/null 2>&1
depbase=`echo libinjection/src/libinjection_sqli.lo | sed 's|[^/]*$|.deps/&|;s|\.lo$||'`;\
```

```
/bin/bash ../libtool  --tag=CXX   --mode=link g++  -g -O2  -lpcre -o rules_optimization optimization/rules_optim
ization-optimization.o ../src/.libs/libmodsecurity.a -lcurl  -lGeoIP -lpcre -lpcre  -lyajl       -lxml2 -lrt
libtool: link: g++ -g -O2 -o rules_optimization optimization/rules_optimization-optimization.o  ../src/.libs/lib
modsecurity.a /usr/lib/x86_64-linux-gnu/libcurl-gnutls.so -lGeoIP -lpcre -lyajl -lxml2 -lrt -pthread
make[2]: Leaving directory '/usr/src/modsecurity/modsecurity-v3.0.0/test'
make[1]: Leaving directory '/usr/src/modsecurity/modsecurity-v3.0.0/test'
make[1]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/usr/src/modsecurity/modsecurity-v3.0.0'
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$
```

```
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$ sudo make install
[sudo] password for frost:
Making install in others
make[1]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/others'
make[2]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/others'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/usr/src/modsecurity/modsecurity-v3.0.0/others'
make[1]: Leaving directory '/usr/src/modsecurity/modsecurity-v3.0.0/others'
Making install in src
make[1]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/src'
make[2]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/src'
make[3]: Entering directory '/usr/src/modsecurity/modsecurity-v3.0.0/src'
 /usr/bin/mkdir -p '/opt/modsecurity-3.0.0/lib'
 /bin/bash ../libtool   --mode=install /usr/bin/install -c   libmodsecurity.la '/opt/modsecurity-3.0.0/lib'
libtool: install: /usr/bin/install -c .libs/libmodsecurity.so.3.0.0 /opt/modsecurity-3.0.0/lib/libmodsecurity.so
.3.0.0
libtool: install: (cd /opt/modsecurity-3.0.0/lib && { ln -s -f libmodsecurity.so.3.0.0 libmodsecurity.so.3 || {
rm -f libmodsecurity.so.3 && ln -s libmodsecurity.so.3.0.0 libmodsecurity.so.3; }; })
```

```
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$ sudo chown -R `whoami` /opt/modsecurity-3.0.0
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$
```

Step 4: Compiling the connector module

ModSecurity 3.0 runs standalone. It is integrated via a NGINX module that organizes the exchange between the webserver and ModSecurity. This allows ModSecurity remain webserver agnostic while the whole integration happens in the connector module.

```
frost@frost-VirtualBox:/usr/src/modsecurity/modsecurity-v3.0.0$ cd /usr/src/modsecurity
frost@frost-VirtualBox:/usr/src/modsecurity$ wget https://github.com/SpiderLabs/ModSecurit
y-nginx/releases/download/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz
--2024-05-05 18:36:29--  https://github.com/SpiderLabs/ModSecurity-nginx/releases/download
/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/owasp-modsecurity/ModSecurity-nginx/releases/download/v1.0.0/
modsecurity-nginx-v1.0.0.tar.gz [following]
--2024-05-05 18:36:30--  https://github.com/owasp-modsecurity/ModSecurity-nginx/releases/d
ownload/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/379
31803/37cf71a4-e59b-11e7-9e37-fc02a43d1f7c?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credenti
al=AKIAVCODYLSA53PQK4ZA%2F20240505%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240505T153
630Z&X-Amz-Expires=300&X-Amz-Signature=a289ac5446e2ffbc452faf8e77697b588a283c1c602b32f4ce4
6d31ddef4a401&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=37931803&response-conte
nt-disposition=attachment%3B%20filename%3Dmodsecurity-nginx-v1.0.0.tar.gz&response-content
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ wget https://github.com/SpiderLabs/ModSecurit
y-nginx/releases/download/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz.sha256
--2024-05-05 18:37:11--  https://github.com/SpiderLabs/ModSecurity-nginx/releases/download
/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz.sha256
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/owasp-modsecurity/ModSecurity-nginx/releases/download/v1.0.0/
modsecurity-nginx-v1.0.0.tar.gz.sha256 [following]
--2024-05-05 18:37:12--  https://github.com/owasp-modsecurity/ModSecurity-nginx/releases/d
ownload/v1.0.0/modsecurity-nginx-v1.0.0.tar.gz.sha256
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/379
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ sha256sum --check modsecurity-nginx-v1.0.0.ta
r.gz.sha256
modsecurity-nginx-v1.0.0.tar.gz: OK
```

Unpack the connector

```
frost@frost-VirtualBox:/usr/src/modsecurity$  tar -xvzf modsecurity-nginx-v1.0.0.tar.gz
modsecurity-nginx-v1.0.0/
modsecurity-nginx-v1.0.0/release.sh
modsecurity-nginx-v1.0.0/AUTHORS
modsecurity-nginx-v1.0.0/tests/
modsecurity-nginx-v1.0.0/tests/nginx-tests-cvt.pl
modsecurity-nginx-v1.0.0/tests/modsecurity.t
modsecurity-nginx-v1.0.0/tests/modsecurity-scoring.t
modsecurity-nginx-v1.0.0/tests/modsecurity-response-body.t
modsecurity-nginx-v1.0.0/tests/modsecurity-request-body.t
modsecurity-nginx-v1.0.0/tests/modsecurity-proxy.t
modsecurity-nginx-v1.0.0/tests/modsecurity-config.t
modsecurity-nginx-v1.0.0/tests/modsecurity-config-debuglog.t
modsecurity-nginx-v1.0.0/tests/modsecurity-config-auditlog.t
modsecurity-nginx-v1.0.0/tests/README.md
modsecurity-nginx-v1.0.0/src/
```

```
frost@frost-VirtualBox:/usr/src/modsecurity$ cd /usr/src/nginx/nginx-1.19.9/
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ export MODSECURITY_LIB="/usr/src/modse
curity/modsecurity-v3.0.0/src/.libs/"
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ export MODSECURITY_INC="/usr/src/modse
curity/modsecurity-v3.0.0/headers/"
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ ./configure --prefix=/opt/nginx-1.19.9
 --with-http_ssl_module --with-threads --with-file-aio --with-compat --add-dynamic-module=
/usr/src/modsecurity/modsecurity-nginx-v1.0.0
checking for OS
 + Linux 5.15.0-105-generic x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found
checking for unistd.h ... found
```

```
Configuration summary
  + using threads
  + using system PCRE library
  + using system OpenSSL library
  + using system zlib library

  nginx path prefix: "/opt/nginx-1.19.9"
  nginx binary file: "/opt/nginx-1.19.9/sbin/nginx"
  nginx modules path: "/opt/nginx-1.19.9/modules"
  nginx configuration prefix: "/opt/nginx-1.19.9/conf"
  nginx configuration file: "/opt/nginx-1.19.9/conf/nginx.conf"
  nginx pid file: "/opt/nginx-1.19.9/logs/nginx.pid"
  nginx error log file: "/opt/nginx-1.19.9/logs/error.log"
  nginx http access log file: "/opt/nginx-1.19.9/logs/access.log"
  nginx http client request body temporary files: "client_body_temp"
  nginx http proxy temporary files: "proxy_temp"
  nginx http fastcgi temporary files: "fastcgi_temp"
  nginx http uwsgi temporary files: "uwsgi_temp"
  nginx http scgi temporary files: "scgi_temp"
```

We need to build nginx server again

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ make
make -f objs/Makefile
make[1]: Entering directory '/usr/src/nginx/nginx-1.19.9'
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I
src/event -I src/event/modules -I src/os/unix -I /usr/src/modsecurity/modsecurity-v3.0.0/h
eaders/ -I objs \
        -o objs/src/core/nginx.o \
        src/core/nginx.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I
src/event -I src/event/modules -I src/os/unix -I /usr/src/modsecurity/modsecurity-v3.0.0/h
eaders/ -I objs \
        -o objs/src/core/ngx_log.o \
        src/core/ngx_log.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I
```

```
cc -o objs/ngx_http_modsecurity_module.so \
objs/addon/src/ngx_http_modsecurity_module.o \
objs/addon/src/ngx_http_modsecurity_pre_access.o \
objs/addon/src/ngx_http_modsecurity_header_filter.o \
objs/addon/src/ngx_http_modsecurity_body_filter.o \
objs/addon/src/ngx_http_modsecurity_log.o \
objs/addon/src/ngx_http_modsecurity_rewrite.o \
objs/ngx_http_modsecurity_module_modules.o \
-Wl,-rpath,/usr/src/modsecurity/modsecurity-v3.0.0/src/.libs/ -L/usr/src/modsecurity/modse
curity-v3.0.0/src/.libs/ -lmodsecurity \
-shared
sed -e "s|%%PREFIX%%|/opt/nginx-1.19.9|" \
        -e "s|%%PID_PATH%%|/opt/nginx-1.19.9/logs/nginx.pid|" \
        -e "s|%%CONF_PATH%%|/opt/nginx-1.19.9/conf/nginx.conf|" \
        -e "s|%%ERROR_LOG_PATH%%|/opt/nginx-1.19.9/logs/error.log|" \
        < man/nginx.8 > objs/nginx.8
make[1]: Leaving directory '/usr/src/nginx/nginx-1.19.9'
```

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo make install
make -f objs/Makefile install
make[1]: Entering directory '/usr/src/nginx/nginx-1.19.9'
test -d '/opt/nginx-1.19.9' || mkdir -p '/opt/nginx-1.19.9'
test -d '/opt/nginx-1.19.9/sbin' \
        || mkdir -p '/opt/nginx-1.19.9/sbin'
test ! -f '/opt/nginx-1.19.9/sbin/nginx' \
        || mv '/opt/nginx-1.19.9/sbin/nginx' \
                '/opt/nginx-1.19.9/sbin/nginx.old'
cp objs/nginx '/opt/nginx-1.19.9/sbin/nginx'
test -d '/opt/nginx-1.19.9/conf' \
        || mkdir -p '/opt/nginx-1.19.9/conf'
cp conf/koi-win '/opt/nginx-1.19.9/conf'
```

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ sudo chown -R `whoami` /opt/nginx-1.19
.9
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ make modules
make -f objs/Makefile modules
make[1]: Entering directory '/usr/src/nginx/nginx-1.19.9'
make[1]: Nothing to be done for 'modules'.
make[1]: Leaving directory '/usr/src/nginx/nginx-1.19.9'
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$
```

copy module to Nginx destination

```
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$  [ ! -d /nginx/modules ] && mkdir /ngi
nx/modules
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$ cp objs/ngx_http_modsecurity_module.so
 /nginx/modules
frost@frost-VirtualBox:/usr/src/nginx/nginx-1.19.9$
```

Step 5: Creating the base configuration

First we need to add the load_module statement in the top level of the configuration.

```
frost@frost-VirtualBox:~$ cd /opt/nginx-1.19.9/conf/
frost@frost-VirtualBox:/opt/nginx-1.19.9/conf$ sudo vim nginx.conf
[sudo] password for frost:
frost@frost-VirtualBox:/opt/nginx-1.19.9/conf$
```

```
daemon            off;
worker_processes  1;
user              www-data;
load_module modules/ngx_http_modsecurity_module.so;

events {
    use           epoll;
    worker_connections  128;
}

error_log         logs/error.log info;
```

enable ModSecurity and do the baseline configuration within http block.

```
http {
    server_tokens off;
    include       mime.types;
    charset       utf-8;
    modsecurity on;
    access_log    logs/access.log  combined;

    server {
        server_name   localhost;
        listen        127.0.0.1:80;

        error_page    500 502 503 504  /50x.html;
```

Create configuration file for modsecurity in “/opt/nginx-1.19.9/conf” directory

```
frost@frost-VirtualBox:/opt/nginx-1.19.9/conf$ sudo vim modsecurity.conf
frost@frost-VirtualBox:/opt/nginx-1.19.9/conf$
```

Paste the following configuration:

SecRuleEngine On

SecRequestBodyAccess On

SecRequestBodyLimit 13107200

SecRequestBodyNoFilesLimit 64000

SecResponseBodyAccess On

SecResponseBodyLimit 10000000

SecTmpDir /tmp/

SecDataDir /tmp/

SecUploadDir /tmp/

SecAuditEngine RelevantOnly

SecAuditLogRelevantStatus "^(?:5|4(?!04))"

SecAuditLogParts ABEFHIJZ

```
SecAuditLogType Serial
SecAuditLog logs/modsec_audit.log
SecAuditLogStorageDir logs/audit

SecPcreMatchLimit 500000
SecPcreMatchLimitRecursion 500000

SecDebugLog logs/modsec_debug.log
SecDebugLogLevel 0

SecDefaultAction "phase:1,pass,log,tag:'Local Lab Service'"
SecDefaultAction "phase:2,pass,log,tag:'Local Lab Service'"

# == ModSec Rule ID Namespace Definition
# Service-specific before Core-Rules:    10000 -  49999
# Service-specific after Core-Rules:     50000 -  79999
# Locally shared rules:                  80000 -  99999
# Recommended ModSec Rules (few):        200000 - 200010
# OWASP Core-Rules:                      900000 – 999999

# === ModSec Recommended Rules (in modsec src package) (ids: 200000-200010)

SecRule REQUEST_HEADERS:Content-Type "(?:application(?:/soap\+|/)|text/)xml" \
  "id:200000,phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=XML"

SecRule REQUEST_HEADERS:Content-Type "application/json" \
  "id:200001,phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=JSON"

SecRule REQBODY_ERROR "!@eq 0" \
  "id:200002,phase:2,t:none,deny,status:400,log,\
  msg:'Failed to parse request body.',logdata:'%{reqbody_error_msg}',severity:2"

SecRule MULTIPART_STRICT_ERROR "!@eq 0" \
  "id:200003,phase:2,t:none,deny,status:403,log, \
  msg:'Multipart request body failed strict validation: \
```

```
  PE %{REQBODY_PROCESSOR_ERROR}, \
  BQ %{MULTIPART_BOUNDARY_QUOTED}, \
  BW %{MULTIPART_BOUNDARY_WHITESPACE}, \
  DB %{MULTIPART_DATA_BEFORE}, \
  DA %{MULTIPART_DATA_AFTER}, \
  HF %{MULTIPART_HEADER_FOLDING}, \
  LF %{MULTIPART_LF_LINE}, \
  SM %{MULTIPART_MISSING_SEMICOLON}, \
  IQ %{MULTIPART_INVALID_QUOTING}, \
  IP %{MULTIPART_INVALID_PART}, \
  IH %{MULTIPART_INVALID_HEADER_FOLDING}, \
  FL %{MULTIPART_FILE_LIMIT_EXCEEDED}'"

SecRule TX:/^MSC_/ "!@streq 0" \
  "id:200005,phase:2,t:none,deny,status:500,\
  msg:'ModSecurity internal error flagged: %{MATCHED_VAR_NAME}'"
# === ModSecurity Rules
#
# ...
```

```
SecRuleEngine On
SecRequestBodyAccess On
SecRequestBodyLimit 13107200

SecRequestBodyNoFilesLimit 64000

SecResponseBodyAccess On
SecResponseBodyLimit 10000000

SecTmpDir /tmp/
SecDataDir /tmp/
SecUploadDir /tmp/

SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"
SecAuditLogParts ABEFHIJZ

SecAuditLogType Serial
SecAuditLog logs/modsec_audit.log
SecAuditLogStorageDir logs/audit

SecPcreMatchLimit 500000
SecPcreMatchLimitRecursion 500000

SecDebugLog logs/modsec_debug.log
                                                                    1,16          Top
```

Figure 16 ModSecurity configuration sample

Include modsecurity.conf file in nginx.conf

```
http {
    server_tokens off;
    include       mime.types;
    charset       utf-8;
    modsecurity on;
    modsecurity_rules_file conf/modsecurity.conf;
    access_log    logs/access.log  combined;

    server {
        server_name   localhost;
        listen        127.0.0.1:80;
```

For auditing and logging configuration, let's do the following

```
frost@frost-VirtualBox:~$ sudo mkdir /nginx/logs/audit
frost@frost-VirtualBox:~$ sudo chown www-data:www-data /nginx/logs/audit
frost@frost-VirtualBox:~$
```

Add some extra rule for whitelist control

```
SecMarker BEGIN_WHITELIST_login


# Make sure there are no URI evasion attempts
SecRule REQUEST_URI "!@streq %{REQUEST_URI_RAW}" \
   "id:11000,phase:1,deny,t:normalizePathWin,log,\
   msg:'URI evasion attempt'"


# START whitelisting block for URI /login
SecRule REQUEST_URI "!@beginsWith /login" \
   "id:11001,phase:1,pass,t:lowercase,nolog,skipAfter:END_WHITELIST_login"
SecRule REQUEST_URI "!@beginsWith /login" \
   "id:11002,phase:2,pass,t:lowercase,nolog,skipAfter:END_WHITELIST_login"


# Validate HTTP method
SecRule REQUEST_METHOD "!@pm GET HEAD POST OPTIONS" \
   "id:11100,phase:1,deny,log,tag:'Login Whitelist',\
   msg:'Method %{MATCHED_VAR} not allowed'"


# Validate URIs
SecRule REQUEST_FILENAME "@beginsWith /login/static/css" \
   "id:11200,phase:1,pass,nolog,tag:'Login Whitelist',\
   skipAfter:END_WHITELIST_URIBLOCK_login"
SecRule REQUEST_FILENAME "@beginsWith /login/static/img" \
```

```
    "id:11201,phase:1,pass,nolog,tag:'Login Whitelist',\
    skipAfter:END_WHITELIST_URIBLOCK_login"
SecRule REQUEST_FILENAME "@beginsWith /login/static/js" \
    "id:11202,phase:1,pass,nolog,tag:'Login Whitelist',\
    skipAfter:END_WHITELIST_URIBLOCK_login"
SecRule REQUEST_FILENAME \
    "@rx ^/login/(displayLogin|login|logout).do$" \
    "id:11250,phase:1,pass,nolog,tag:'Login Whitelist',\
    skipAfter:END_WHITELIST_URIBLOCK_login"

# If we land here, we are facing an unknown URI,
# which is why we will respond using the 404 status code
SecAction "id:11299,phase:1,deny,status:404,log,tag:'Login Whitelist',\
    msg:'Unknown URI %{REQUEST_URI}'"

SecMarker END_WHITELIST_URIBLOCK_login

# Validate parameter names
SecRule ARGS_NAMES "!@rx ^(username|password|sectoken)$" \
    "id:11300,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'Unknown parameter: %{MATCHED_VAR_NAME}'"

# Validate each parameter's cardinality
SecRule &ARGS:username  "@gt 1" \
    "id:11400,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'%{MATCHED_VAR_NAME} occurring more than once'"
SecRule &ARGS:password  "@gt 1" \
    "id:11401,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'%{MATCHED_VAR_NAME} occurring more than once'"
SecRule &ARGS:sectoken  "@gt 1" \
    "id:11402,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'%{MATCHED_VAR_NAME} occurring more than once'"

# Check individual parameters
SecRule ARGS:username "!@rx ^[a-zA-Z0-9.@-]{1,32}$" \
    "id:11500,phase:2,deny,log,tag:'Login Whitelist',\
```

```
    msg:'Invalid parameter format: %{MATCHED_VAR_NAME} (%{MATCHED_VAR})'"
SecRule ARGS:sectoken "!@rx ^[a-zA-Z0-9]{32}$" \
    "id:11501,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'Invalid parameter format: %{MATCHED_VAR_NAME} (%{MATCHED_VAR})'"
SecRule ARGS:password "@gt 64" \
    "id:11502,phase:2,deny,log,t:length,tag:'Login Whitelist',\
    msg:'Invalid parameter format: %{MATCHED_VAR_NAME} too long (%{MATCHED_VAR} bytes)'"
SecRule ARGS:password "@validateByteRange 33-244" \
    "id:11503,phase:2,deny,log,tag:'Login Whitelist',\
    msg:'Invalid parameter format: %{MATCHED_VAR_NAME} (%{MATCHED_VAR})'"

SecMarker END_WHITELIST_login
```

4. Include OWASP ModSecurity Core Rule Set v3

Step 1: Download the source

```
frost@frost-VirtualBox:/nginx$ wget https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/refs/tags/v3.0.0.tar.gz
--2024-05-11 23:15:22--  https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/refs/tags/v3.0.0.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/SpiderLabs/owasp-modsecurity-crs/tar.gz/refs/tags/v3.0.0 [following]
--2024-05-11 23:15:23--  https://codeload.github.com/SpiderLabs/owasp-modsecurity-crs/tar.gz/refs/tags/v3.0.0
Resolving codeload.github.com (codeload.github.com)... 140.82.121.10
Connecting to codeload.github.com (codeload.github.com)|140.82.121.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: ‘v3.0.0.tar.gz’

v3.0.0.tar.gz                       [    <=>                                                ] 150.23K   200KB/s    in 0.8s

2024-05-11 23:15:24 (200 KB/s) - ‘v3.0.0.tar.gz’ saved [153838]

frost@frost-VirtualBox:/nginx$
```

Extract the source code in Nginx directory

```
frost@frost-VirtualBox:/nginx$ sudo tar -xzf v3.0.0.tar.gz
frost@frost-VirtualBox:/nginx$ cd owasp-modsecurity-crs-3.0.0/
frost@frost-VirtualBox:/nginx/owasp-modsecurity-crs-3.0.0$ sudo mv
CHANGES                 .gitignore              id_renumbering/         LICENSE                 .travis.yml
crs-setup.conf.example  .gitmodules             INSTALL                 README.md               util/
documentation/          IDNUMBERING             KNOWN_BUGS              rules/
frost@frost-VirtualBox:/nginx/owasp-modsecurity-crs-3.0.0$ sudo mv crs-setup.conf.example crs-setup.conf
```

Step 2: Include CRS rules

Include the crs-setup.conf in the Nginx configuration file as a modsecurity_rules_file

Do the same for other configuration files that end with ‘.conf’ in the directory owasp-modsecurity-crs-3.0.0/rules/

```
http {
    server_tokens off;
    include       mime.types;
    charset       utf-8;
    modsecurity on;
    modsecurity_rules_file conf/modsecurity.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/crs-setup.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-901-INITIALIZATION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-903.9001-DRUPAL-EXCLUSION-RULES.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-903.9002-WORDPRESS-EXCLUSION-RULES.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-905-COMMON-EXCEPTIONS.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-910-IP-REPUTATION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-911-METHOD-ENFORCEMENT.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-912-DOS-PROTECTION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-913-SCANNER-DETECTION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-921-PROTOCOL-ATTACK.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-931-APPLICATION-ATTACK-RFI.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-933-APPLICATION-ATTACK-PHP.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/REQUEST-949-BLOCKING-EVALUATION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-950-DATA-LEAKAGES.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-951-DATA-LEAKAGES-SQL.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-952-DATA-LEAKAGES-JAVA.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-953-DATA-LEAKAGES-PHP.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-954-DATA-LEAKAGES-IIS.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-959-BLOCKING-EVALUATION.conf;
    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-980-CORRELATION.conf;
    access_log    logs/access.log  combined;

```

Setting up grafana monitoring tool

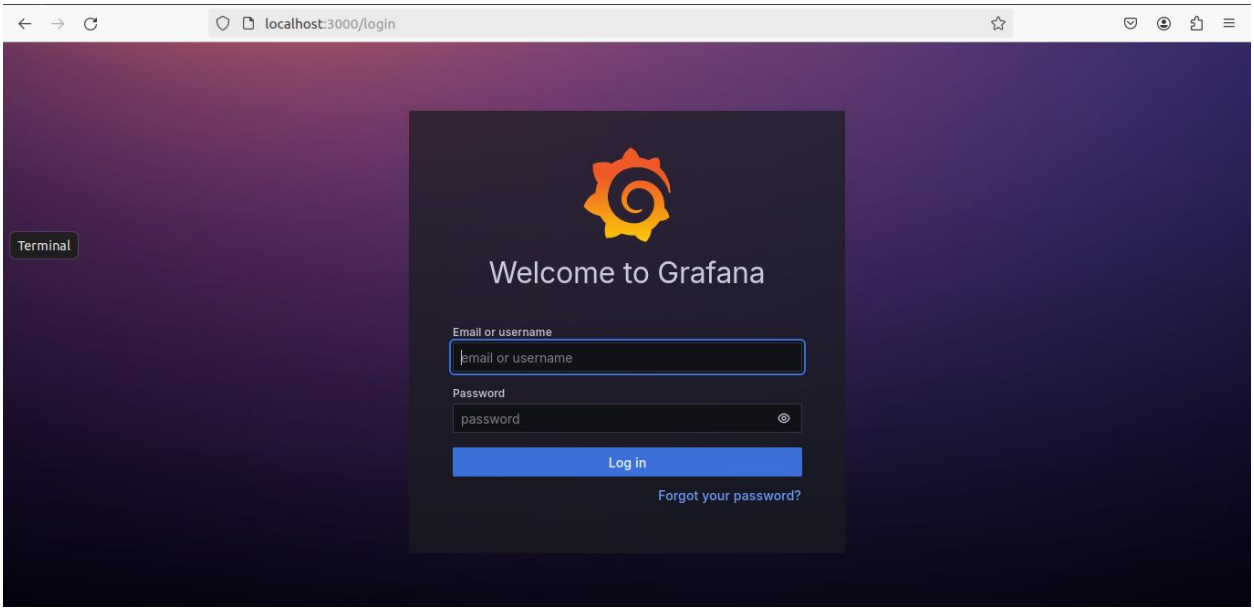
```
frost@frost-VirtualBox:~$ sudo apt-get install -y adduser libfontconfig1 musl
[sudo] password for frost:
Reading package lists... Done
Building dependency tree
Reading state information... Done
adduser is already the newest version (3.118ubuntu2).
adduser set to manually installed.
libfontconfig1 is already the newest version (2.13.1-2ubuntu3).
libfontconfig1 set to manually installed.
The following NEW packages will be installed:
  musl
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 377 kB of archives.
After this operation, 790 kB of additional disk space will be used.
Get:1 http://lb.archive.ubuntu.com/ubuntu focal/universe amd64 musl amd64 1.1.24-1 [377 kB]
Fetched 377 kB in 2s (188 kB/s)
Selecting previously unselected package musl:amd64.
(Reading database ... 168377 files and directories currently installed.)
Preparing to unpack .../musl_1.1.24-1_amd64.deb ...
Unpacking musl:amd64 (1.1.24-1) ...
Setting up musl:amd64 (1.1.24-1) ...
Processing triggers for man-db (2.9.1-1) ...
frost@frost-VirtualBox:~$
```

```
frost@frost-VirtualBox:~$ wget https://dl.grafana.com/enterprise/release/grafana-enterprise_10.4.2_amd64.deb
--2024-05-12 12:06:54--  https://dl.grafana.com/enterprise/release/grafana-enterprise_10.4.2_amd64.deb
Resolving dl.grafana.com (dl.grafana.com)... 199.232.82.217, 2a04:4e42:54::729
Connecting to dl.grafana.com (dl.grafana.com)|199.232.82.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 119658678 (114M) [application/octet-stream]
Saving to: ‘grafana-enterprise_10.4.2_amd64.deb’

grafana-enterprise_10.4.2_amd64.deb   100%[=========================================================================>] 114.12M

2024-05-12 12:10:45 (512 KB/s) - ‘grafana-enterprise_10.4.2_amd64.deb’ saved [119658678/119658678]
```

```
frost@frost-VirtualBox:~$ sudo dpkg -i grafana-enterprise_10.4.2_amd64.deb
Selecting previously unselected package grafana-enterprise.
(Reading database ... 168390 files and directories currently installed.)
Preparing to unpack grafana-enterprise_10.4.2_amd64.deb ...
Unpacking grafana-enterprise (10.4.2) ...
Setting up grafana-enterprise (10.4.2) ...
Adding system user `grafana' (UID 128) ...
Adding new user `grafana' (UID 128) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
Processing triggers for systemd (245.4-4ubuntu3.23) ...
frost@frost-VirtualBox:~$
```

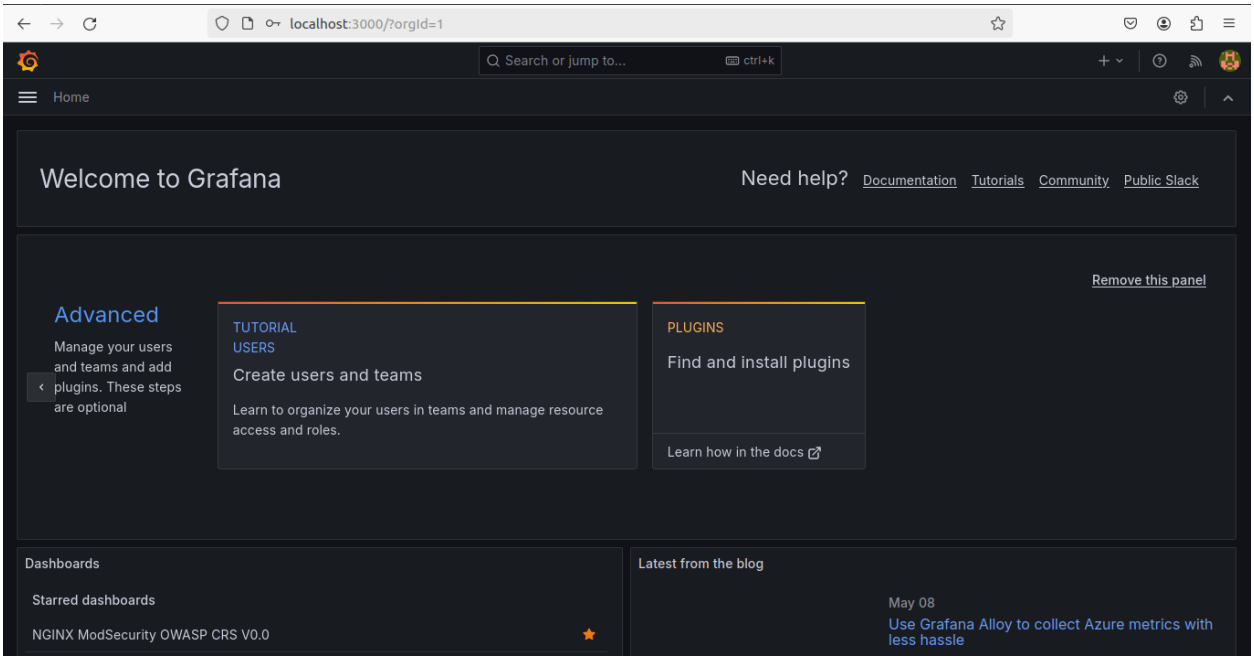
```
frost@frost-VirtualBox:~$ sudo /bin/systemctl daemon-reload
frost@frost-VirtualBox:~$  sudo /bin/systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /lib/systemd/system/grafana-server.service.
frost@frost-VirtualBox:~$
```

Enter http://localhost:3000/



Default username: “admin”

Default password: “admin”



Installing Prometheus

```
frost@frost-VirtualBox:/nginx$ git clone https://github.com/knyar/nginx-lua-prometheus.git
Cloning into 'nginx-lua-prometheus'...
remote: Enumerating objects: 567, done.
remote: Counting objects: 100% (261/261), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 567 (delta 228), reused 222 (delta 216), pack-reused 306
Receiving objects: 100% (567/567), 191.79 KiB | 535.00 KiB/s, done.
Resolving deltas: 100% (350/350), done.
frost@frost-VirtualBox:/nginx$
```

```
frost@frost-VirtualBox:/nginx$ sudo apt-get -y install libnginx-mod-http-lua
[sudo] password for frost:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libluajit-5.1-2 libluajit-5.1-common libnginx-mod-http-ndk nginx-common
Suggested packages:
  fcgiwrap nginx-doc
The following NEW packages will be installed:
  libluajit-5.1-2 libluajit-5.1-common libnginx-mod-http-lua libnginx-mod-http-ndk nginx-common
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 473 kB of archives.
After this operation, 1,579 kB of additional disk space will be used.
Get:1 http://lb.archive.ubuntu.com/ubuntu focal/universe amd64 libluajit-5.1-common all 2.1.0~beta3+dfsg-5.1build1 [44.3 kB]
Get:2 http://lb.archive.ubuntu.com/ubuntu focal/universe amd64 libluajit-5.1-2 amd64 2.1.0~beta3+dfsg-5.1build1 [228 kB]
Get:3 http://lb.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-common all 1.18.0-0ubuntu1.4 [37.7 kB]
Get:4 http://lb.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libnginx-mod-http-ndk amd64 1.18.0-0ubuntu1.4 [10.5 kB]
Get:5 http://lb.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libnginx-mod-http-lua amd64 1.18.0-0ubuntu1.4 [153 kB]
Fetched 473 kB in 1s (530 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libluajit-5.1-common.
(Reading database ... 180604 files and directories currently installed.)
Preparing to unpack .../libluajit-5.1-common_2.1.0~beta3+dfsg-5.1build1_all.deb ...
Unpacking libluajit-5.1-common (2.1.0~beta3+dfsg-5.1build1) ...
Selecting previously unselected package libluajit-5.1-2:amd64.
Preparing to unpack .../libluajit-5.1-2_2.1.0~beta3+dfsg-5.1build1_amd64.deb ...
Unpacking libluajit-5.1-2:amd64 (2.1.0~beta3+dfsg-5.1build1) ...
Selecting previously unselected package nginx-common.
Preparing to unpack .../nginx-common_1.18.0-0ubuntu1.4_all.deb ...
```

https://github.com/knyar/nginx-lua-prometheus

README MIT license

OpenResty users will find this library in opm. It is also available via luarocks.

Quick start guide

To track request latency broken down by server name and request count broken down by server name and status, add the following to the `http` section of `nginx.conf`:

```
lua_shared_dict prometheus_metrics 10M;
lua_package_path "/path/to/nginx-lua-prometheus/?.lua;;";

init_worker_by_lua_block {
  prometheus = require("prometheus").init("prometheus_metrics")

  metric_requests = prometheus:counter(
    "nginx_http_requests_total", "Number of HTTP requests", {"host", "status"})
  metric_latency = prometheus:histogram(
    "nginx_http_request_duration_seconds", "HTTP request latency", {"host"})
  metric_connections = prometheus:gauge(
    "nginx_http_connections", "Number of HTTP connections", {"state"})
}

log_by_lua_block {
  metric_requests:inc(1, {ngx.var.server_name, ngx.var.status})
  metric_latency:observe(tonumber(ngx.var.request_time), {ngx.var.server_name})
}
```

This:

Open nginx.conf /opt/nginx-1.19.9/conf Save

```
38    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-950-DATA-LEAKAGES.conf;
39    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-951-DATA-LEAKAGES-SQL.conf;
40    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-952-DATA-LEAKAGES-JAVA.conf;
41    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-953-DATA-LEAKAGES-PHP.conf;
42    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-954-DATA-LEAKAGES-IIS.conf;
43    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-959-BLOCKING-EVALUATION.conf;
44    modsecurity_rules_file owasp-modsecurity-crs-3.0.0/rules/RESPONSE-980-CORRELATION.conf;
45    access_log    logs/access.log  combined;
46
47    lua_shared_dict prometheus_metrics 10M;
48    lua_package_path "/path/to/nginx-lua-prometheus/?.lua;;";
49
50    init_worker_by_lua_block {
51    prometheus = require("prometheus").init("prometheus_metrics")
52    metric_requests = prometheus:counter("nginx_http_requests_total", "Number of HTTP requests", {"host", "status"})
53    metric_latency = prometheus:histogram("nginx_http_request_duration_seconds", "HTTP request latency", {"host"})
54    metric_connections = prometheus:gauge("nginx_http_connections", "Number of HTTP connections", {"state"})
55    }
56
57    log_by_lua_block {
58    metric_requests:inc(1, {ngx.var.server_name, ngx.var.status})
59    metric_latency:observe(tonumber(ngx.var.request_time), {ngx.var.server_name})
60    }
61
62    server {
63        server_name   localhost;
64        listen        127.0.0.1:80;
```

Change this path to the directory where Prometheus is installed

```
46
47      lua_shared_dict prometheus_metrics 10M;
48      lua_package_path "/path/to/nginx-lua-prometheus/?.lua;;";
49
50      init_worker_by_lua_block {
51      prometheus = require("prometheus").init("prometheus_metrics")
52      metric_requests = prometheus:counter("nginx_http_requests_total", "Number of HTTP requests", {"host", "status"})
53      metric_latency = prometheus:histogram("nginx_http_request_duration_seconds", "HTTP request latency", {"host"})
```

Find the path of module installed

```
frost@frost-VirtualBox:/usr/lib/nginx/modules$ ls
ndk_http_module.so  ngx_http_lua_module.so
frost@frost-VirtualBox:/usr/lib/nginx/modules$
```

Load the module manually

```
3 user                www-data;
4 load_module modules/ngx_http_modsecurity_module.so;
5 load_module /usr/lib/nginx/modules/ngx_http_lua_module.so;
6 events {
7     use           epoll;
8     worker_connections  128;
9 }
```