Báo cáo Nginx – Reverse proxy

1. Nginx

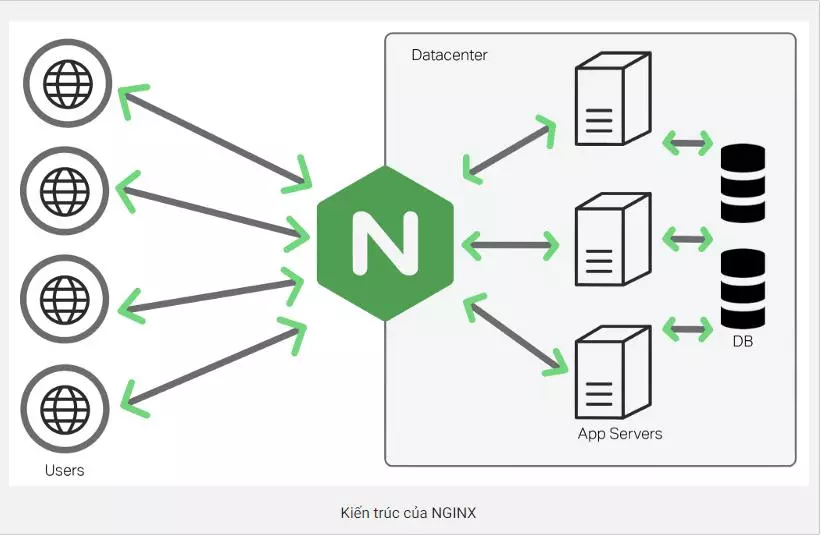
* Nginx là một web server sử dụng kiến trúc đơn luồng, được sử dụng để phục vụ web và đóng vai trò như load balancer, HTTP cache, hay sử dụng như một reverve proxy.

+ Web server: phân phối các tài nguyên web như html, css,… đến người dung

+ Reverse proxy: nhận request từ client chuyển tới server và ngược lại gửi response cho client khi server trả về.

+ Load balancer: phân phối tải giữa nhiều máy chủ và đảm bảo hiệu suất và độ ổn định của ứng dụng.

+ HTTP cache: nginx lưu trữ các phản hồi từ máy chủ gốc để giảm tải và tăng tốc độ truy cập cho các yêu cầu tiếp theo.

* Mô hình:
* Cài đặt: Ubuntu, Debian:

sudo apt-get install nginx

* Khởi động dịch vụ:
  + systemctl start nginx
* Kiểm tra trạng thái:
  + Systemctl status nginx
* Cấu hình: được lưu trong thư mục /etc/nginx, một số file thư mục quan trọng:

+ file nginx.conf: tệp cấu hình chính, quyết định cách Nginx xử lý các yêu cầu và hoạt động.

+ thư mục sites-available: thư mục cấu hình máy chủ ảo, mặc định được cấu hình trong file default

+thư mục sites-enabled: chứa liên kết đến tệp sites-available, nginx chỉ tải các tệp cấu hình trong thư mục này.

+ thư mục ssl: chứa chứng chỉ SSL và khóa riêng được sử dụng cho các kết nối HTTPS

1. Reverse proxy

* Reverse proxy là một server trung gian giữ client và server, kiểm soát các request từ client và điều phối những request đó tới server phù hợp để được xử lý, đồng thời sẽ nhận response từ server và trả về cho client.
* Ưu điểm:
  + Che giấu server thực được dung.
  + Load balancing, proxy có thể chia đều request của client tới server.
  + Có thể trở thành WAF cho các ứng dụng web.
* Cấu hình Nginx thành Reverse proxy

A diagram of a network

Description automatically generated

+ Đầu tiên cấu hình cài đặt DVWA hoạt động trên apache2 port 8080 làm web server

+ Cấu hình trong file sites-available/default:

* Port: Mặc định port hoạt động là 80, có thể thay đổi

listen 80 default\_server;

listen [::]:80 default\_server;

\* default\_server: khi không có server\_name nào trong cấu hình khớp với yêu cầu người dung thì khối server này được chọn.

* Server\_name: sử dụng để chỉ định tên miền hoặc địa chỉ ip mà nginx phục vụ trong khối server đó.

server\_name localhost;

* Location: xác định các chỉ thị nginx xử lí các yêu cầu cho từng URL đường dẫn cụ thể ví dụ:

Location /dvwa {

proxy\_pass http://localhost:8080/DVWA;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

+ Khởi động lại Nginx:

systemctl restart nginx

* Khi truy cập vào <http://localhost/dvwa>, nginx sẽ nhận request và chuyển tới server <http://localhost:8080/DVWA>, nginx trở thành reverse proxy.

1. Modsecurity - WAF
2. Modsecurity

* Modsecurity là một tường lửa ứng dụng web mã nguồn mở, được thiết kế để bảo vệ ứng dụng web khỏi các cuộc tấn công như SQLi, XSS,… ngoài ra còn có thể tích hợp với các máy chủ web như apache2, nginx…
* Tính năng:

+ Phát hiện và ngăn chặn tấn công

+ Giám sát và ghi log

+ Dễ dàng tùy chỉnh các quy tắc

+ Bảo vệ theo thời gian thực

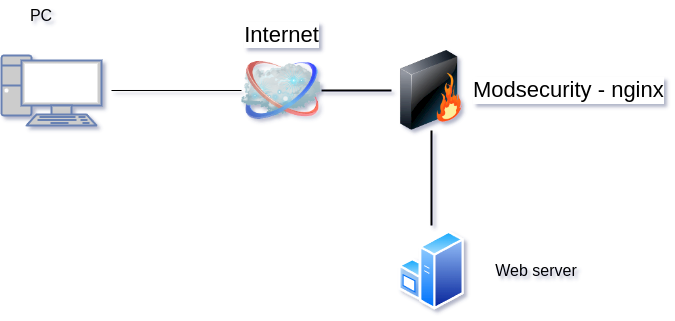
* Cách hoạt động:

+ Tiếp nhận yêu cầu HTTP

+ So sánh với các quy tắc: so sánh với các quy tắc để xác định xem yêu cầu hợp lệ hay không nếu yêu cầu khớp với một quy tắc nào đó sẽ bị chặn hoặc ghi log.

+ Ghi lại log: bao gồm các yêu cầu bị chặn hoặc cho phép, phục vụ cho việc giám sát và phân tích sau này.

1. Cài đặt cấu hình Modsecurity kết hợp nginx.



* Bước 1: tải xuống các thư viện cần thiết

sudo apt-get install bison build-essential ca-certificates curl dh-autoreconf doxygen flex gawk git iputils-ping libcurl4-gnutls-dev libexpat1-dev libgeoip-dev liblmdb-dev libpcre3-dev libpcre++-dev libssl-dev libtool libxml2 libxml2-dev libyajl-dev locales lua5.3-dev pkg-config wget zlib1g-dev zlibc libxslt libgd-dev

* Bước 2: tải xuống modsecurity từ github

+ sử dụng câu lệnh: cd /opt && sudo git clone <https://github.com/owasp-modsecurity/ModSecurity.git>

+ vào thư mục modsecurity vừa tải chạy command

sudo git submodule init

sudo git submodule update

+ chạy script: sudo ./build.sh

+ chạy file configure: sudo ./configure

+ chạy command: sudo make và sudo make install

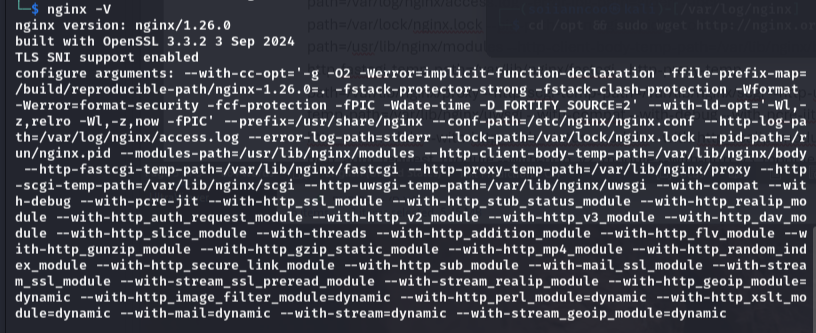
* Bước 3: tải xuống modsecurity-nginx connector

cd /opt && git clone https://github.com/owasp-modsecurity/ModSecurity-nginx

* Bước 4: dựng modsecurity module cho nginx

+ kiểm tra phiên bản nginx và tải xuống source nginx phiên bản đó trên trang chủ <http://nginx.org/download>

+ vào thư mục source, sử dụng command: nginx -V, lưu lại đối số của configure



+ chạy file configure với đối số vừa nhận

sudo ./configure --add-dynamic-module=../ModSecurity-nginx <Configure Arguments>

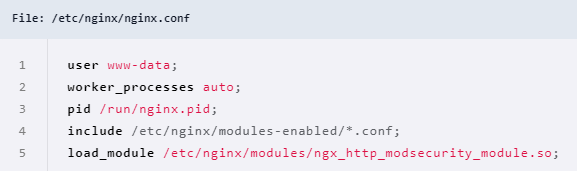
+ dựng module bằng lệnh: sudo make modules

* Bước 5: tải modsecurity modules vào trong nginx

+ trong file /etc/nginx/nginx.conf thêm dòng sau:

load\_module /etc/nginx/modules/ngx\_http\_modsecurity\_module.so;

(thay đường dẫn trên bằng đường dẫn file ngx\_http\_modsecurity\_module.so trong thư mục source nginx trên máy)



* Bước 6: Cấu hình Modsecurity

+ tạo thư mục mới modsec trong /etc/nginx: sudo mkdir -p /etc/nginx/modsec

+ copy file unicode.mapping và modsecurity.conf-recommended vào thư mục modsec vừa tạo:

sudo cp /opt/ModSecurity/unicode.mapping /etc/nginx/modsec

sudo cp /opt/ModSecurity/modsecurity.conf-recommended /etc/nginx/modsec/modsecurity.conf

+ đổi tên file modsecurity.conf-recommended thành modsecurity.conf

+ mở file và đặt SecRuleEngie thành on

A screenshot of a computer

Description automatically generated

+ tạo file mới main.conf trong /etc/nginx/modsec với nội dung sau:

Include /etc/nginx/modsec/modsecurity.conf

* Bước 7: cấu hình lại Nginx

+trong file /etc/nginx/sites-available/default thêm 2 dòng sau:

modsecurity on;

modsecurity\_rules\_file /etc/nginx/modsec/main.conf;



* Bước 8: viết rule cho modsecurity

Cấu trúc: SecRule <Variable> <Operator> <Action>

+ tạo thư mục rules trong /etc/nginx/modsec và viết rules trong thư mục này.

Ví dụ một số rules cơ bản cho các cuộc tấn công web phổ biến

* Tấn công SQL injection, tạo file sqli.conf nội dung:

SecRule ARGS "(?i:select.\*from|union.\*select|insert.\*into|drop\s+table)" \

"id:12345,phase:2,deny,status:403,msg:'Detected SQL Injection attempt.'"

SecRule ARGS "(?i:or\s+1=1|and\s+1=1)" \

"id:12346,phase:2,deny,status:403,msg:'Detected SQL Injection attempt.'"

* Tấn công XSS, tạo file xss.conf với nội dung:

SecRule ARGS "(<|%3C).\*script.\*(>|%3E)" \ "id:1003,phase:2,deny,status:403,msg:'XSS attack detected - script tags'"

SecRule ARGS "(?i:(alert|onload|onerror|onmouseover|document\.cookie))" \"id:1004,phase:2,deny,status:403,msg:'XSS attempt detected'"

SecRule RESPONSE\_BODY "<script.\*>.\*</script>" \

"id:1005,phase:4,deny,status:403,msg:'XSS attempt in response detected.'"

* Tấn công SSRF, tạo file ssrf.conf

SecRule ARGS:url "@rx ^https?://(127\.0\.0\.1|10\.\d+\.\d+\.\d+|172\.16\.\d+\.\d+|192\.168\.\d+\.\d+)" \"id:1000001,phase:2,t:lowercase,deny,status:403,msg:'Blocked SSRF'

* Tấn công command injection, tạo file cmdi.conf

SecRule ARGS "@rx ;|&&|\|\|" \ "id:2001,phase:2,deny,status:403,msg:'Potential command injection detected'"

* Tấn công Bruteforce

SecAction “id: 1000001, phase: 1, initcol:ip=%{REMOTE\_ADDR}, nolog”

SecRule IP:failed\_login “@gt 5” “deny, status:403,msg:’bruteforce attack’, setvar:count=+1”

SecAction “id:1000003, phase:5, deprecatevar:ip.failed\_login=1/300”

SecRule RESPONSE\_STATUS "@streq 401" "id:'1000004',phase:5,initcol:ip=%{REMOTE\_ADDR},setvar:ip.failed\_login=+1"

* Tấn công XXE injection

SecRule REQUEST\_BODY "<!DOCTYPE" "id:'1000',phase:2,deny,log,msg:'Detect XXE Attack "

SecRule REQUEST\_BODY "(file|gopher|http|ftp)://(localhost|127\.0\.0\.1|::1)" \

"id:'1001',phase:2,deny,log,msg:'Detect XXE Attack'"

SecRule REQUEST\_BODY "<!ENTITY" "id:'1002',phase:2,deny,log,msg:'Detect XXE Attack "

* File upload vulnerabilities

SecRule REQUEST\_CONTENT\_LENGTH "@gt 1048576" "id:'2003',phase:2,deny,log,msg:'Uploaded file size limit (1MB)'"

SecRule FILES\_NAMES "@rx \.(php|sh)$" \

"id:'3000',phase:2,deny,log,msg:'File upload vulnerabilities attack detected'"

* Tấn công CSRF

SecRule REQUEST\_METHOD "POST" \

"id:'4000',phase:2,chain,deny,status:403,log,msg:'CSRF Detected "

SecRule REQUEST\_HEADERS:Referer "!@beginsWith https://localhost" \

"id:'4001',phase:2,deny,status:403,log,msg:'CSRF Detected "

SecRule REQUEST\_HEADERS:Origin "!@streq https://localhost" \

"id:'4002',phase:2,deny,status:403,log,msg:'CSRF Detected "

* Tấn công lfi-rfi

SecRule REQUEST\_URI "@rx ^/(etc/passwd|config|log|backup|shell)" \

"id:'5003',phase:2,deny,status:403,log,msg:'Access file blocked'"

SecRule REQUEST\_URI ".\*(http|https|ftp|file|php://|data:).\*" \

"id:'5001',phase:2,deny,status:403,log,msg:'RFI attack detected'"

* Header injection

SecRule REQUEST\_HEADERS "@rx [\n\r]" \

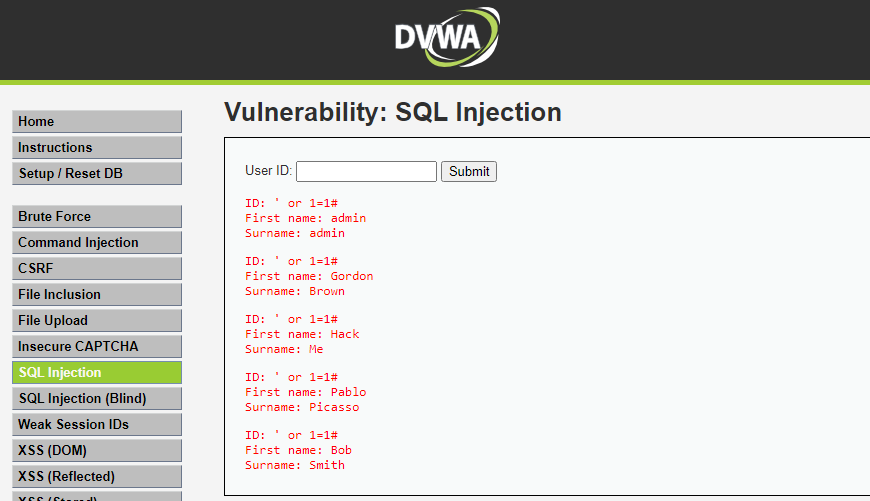
"id:'6000',phase:2,deny,status:403,log,msg:'Header Injection detected'"

Thêm các file rules trên vào file main.conf:

Include /etc/nginx/modsec/rules/\*.conf

Khởi động lại nginx: systemctl restart nginx

Ví dụ khi chưa bật modsecurity hoặc chưa thêm các file rules vào file main.conf



* Sau khi thêm rules hoặc bật modsecurity sẽ bị block và chuyển về trang 403

1. Modsecurity-crs

* Là bộ quy tắc được phát triển cho modsecurity chống lại các cuộc tấn công XSS, SQL injection và nhiều loại hình tấn công khác.
* Cài đặt:

+ tải xuống từ trang github:

sudo git clone https://github.com/coreruleset/coreruleset /usr/local/modsecurity-crs

+ đổi tên file crs-setup.conf.example thành crs-setup.conf

+ xóa các đuôi file .example trong thư mục rules.

+ trong file main.conf thêm các file configure của modsecurity-crs sau:

Include /usr/local/modsecurity-crs/crs-setup.conf

Include /usr/local/modsecurity-crs/rules/\*.conf

+ Khởi động lại nginx và sử dụng