yum -y install epel-release perl pcre-devel openssl-devel gcc curl unzip wget zip

useradd www

cd /home && wget https://mirror.cnop.net/web/openresty/openresty-1.19.9.1.tar.gz

tar zxvf openresty-1.19.9.1.tar.gz && cd openresty-1.19.9.1

./configure --user=www --group=www --prefix=/usr/local/openresty --with-pcre --with-stream --with-threads --with-file-aio --with-http\_v2\_module --with-http\_ssl\_module --with-http\_realip\_module --with-http\_gzip\_static\_module --with-http\_stub\_status\_module

make && sudo make install

ln -sf /usr/local/openresty/nginx/sbin/nginx /usr/local/bin/nginx
ln -sf /usr/local/openresty/nginx/sbin/nginx /usr/sbin/nginx

cd /etc/init.d

wget https://mirror.cnop.net/web/openresty/openresty && mv openresty nginx

vi nginx #以下内容请根据自己的安装路径来

#!/bin/bash

# nginx Startup script for the Nginx HTTP Server

# chkconfig: - 85 15

# description: Nginx is a high-performance web and proxy server.

# It has a lot of features, but it's not for everyone.

# processname: nginx

# pidfile: /usr/local/openresty/nginx/logs/nginx.pid

# config: /usr/local/openresty/nginx/conf/nginx.conf

#nginx path

nginxd=/usr/local/openresty/bin/openresty

nginx\_config=/usr/local/openresty/nginx/conf/nginx.conf

nginx\_pid=/usr/local/openresty/nginx/logs/nginx.pid

RETVAL=0

prog="nginx"

# Source function library.

. /etc/rc.d/init.d/functions

# Source networking configuration.

. /etc/sysconfig/network

# Check that networking is up.

[ ${NETWORKING} = "no" ] && exit 0

[ -x $nginxd ] || exit 0

# Start nginx daemons functions.

start() {

if [ -e $nginx\_pid ];then

echo "nginx already running...."

exit 1

fi

echo -n $"Starting $prog: "

daemon $nginxd -c ${nginx\_config}

RETVAL=$?

echo

[ $RETVAL = 0 ] && touch /var/lock/subsys/nginx

return $RETVAL

}

# Stop nginx daemons functions.

stop() {

echo -n $"Stopping $prog: "

killproc $nginxd

RETVAL=$?

echo

[ $RETVAL = 0 ] && rm -f /var/lock/subsys/nginx /usr/local/openresty/nginx/logs/nginx.pid

}

# reload nginx service functions.

reload() {

echo -n $"Reloading $prog: "

#kill -HUP `cat ${nginx\_pid}`

killproc $nginxd -HUP

RETVAL=$?

echo

}

# See how we were called.

case "$1" in

start)

start

;;

stop)

stop

;;

reload)

reload

;;

restart)

stop

start

;;

status)

status $prog

RETVAL=$?

;;

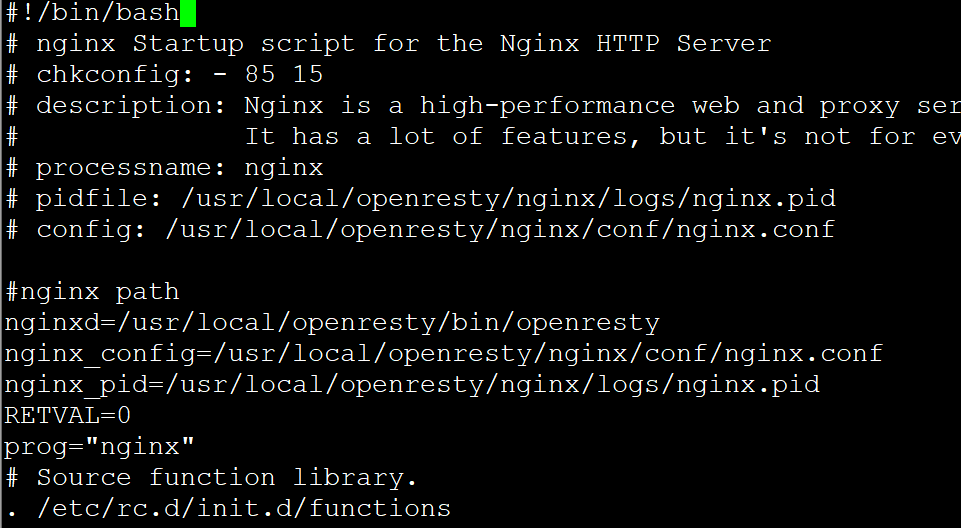
\*)

echo $"Usage: $prog {start|stop|restart|reload|status|help}"

exit 1

esac

exit $RETVAL



chmod +x nginx

chmod +x /etc/rc.d/rc.local

[Openresty实现WAF防火墙功能](https://www.cnblogs.com/cheyunhua/p/13395745.html)

yum install -y gcc-c++ libtool gmake make pcre pcre-devel openssl openssl-devel zlib zlib-devel readline readline-devel

本步骤可省略：

vi /usr/local/openresty/nginx/conf/nginx.conf

server中加入以下：

location /hello {

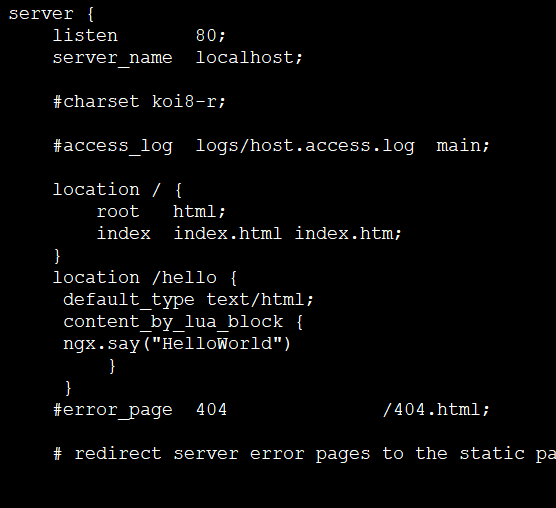
default\_type text/html;

content\_by\_lua\_block {

ngx.say("HelloWorld")

}

}



service nginx reload

访问以下链接可测试lua：

<http://ip/hello>

################################################################################################

**WAF的功能**

支持IP白名单和黑名单功能，直接将黑名单的IP访问拒绝。

支持URL白名单，将不需要过滤的URL进行定义。

支持User-Agent的过滤，匹配自定义规则中的条目，然后进行处理（返回403）。

支持CC攻击防护，单个URL指定时间的访问次数，超过设定值，直接返回403。

支持Cookie过滤，匹配自定义规则中的条目，然后进行处理（返回403）。

支持URL过滤，匹配自定义规则中的条目，如果用户请求的URL包含这些，返回403。

支持URL参数过滤，原理同上。

支持日志记录，将所有拒绝的操作，记录到日志中去。

日志记录为JSON格式，便于日志分析，例如使用ELKStack进行攻击日志收集、存储、搜索和展示。

**部署WAF:WAF已经有人通过lua写出了这个开源的功能，在此直接拿来用即可。GitHub地址：**<https://github.com/unixhot/waf>

下载waf模块：

git clone <https://github.com/unixhot/waf.git>

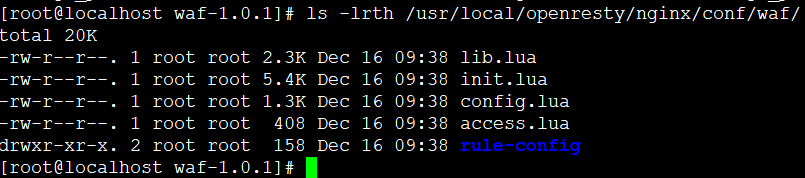
或

wget <https://mirror.cnop.net/web/module/waf-1.0.1.tar.gz> && tar zxvf [waf-1.0.1.tar.gz](https://mirror.cnop.net/web/module/waf-1.0.1.tar.gz)

cd waf-1.0.1 && cp -r waf /usr/local/openresty/nginx/conf

ln -s /usr/local/openresty/lualib/resty/ /usr/local/openresty/nginx/conf/waf/resty

ls -lrth /usr/local/openresty/nginx/conf/waf/



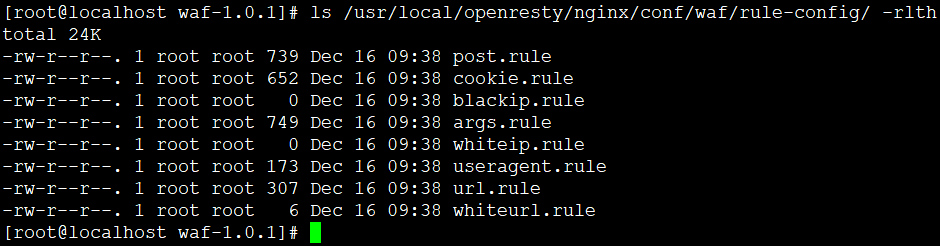
以上access.lua、lib.lua、init.lua都是功能实现的lua代码，如果不具备lua的开发能力，我们一般不会去进行改动

config.lua为各个功能的配置文件

rule-config目录存放了各种防御策略规则

我们需要经常改动config.lua和存储策略的文件

ls /usr/local/openresty/nginx/conf/waf/rule-config/ -rlth



 #Cookie策略文件

#异常Get参数策略文件

#白名单URL策略文件

#IP白名单策略文件

#异常UserAgent策略文件

#异常URL策略文件

#异常POST参数策略文件

#IP黑名单策略文件

**Openresty引入WAF模块**

1.修改nginx配置来引入WAF模块

如下在Nginx中加入以下配置来引入WAF模块

lua-resty-core 模块：

cd /usr/local/openresty/

git clone <https://github.com/openresty/lua-resty-core.git>

或

wget https://mirror.cnop.net/web/module/lua-resty-core.tar.gz && tar zxvf lua-resty-core.tar.gz

vi /usr/local/openresty/nginx/conf/nginx.conf

http中加入

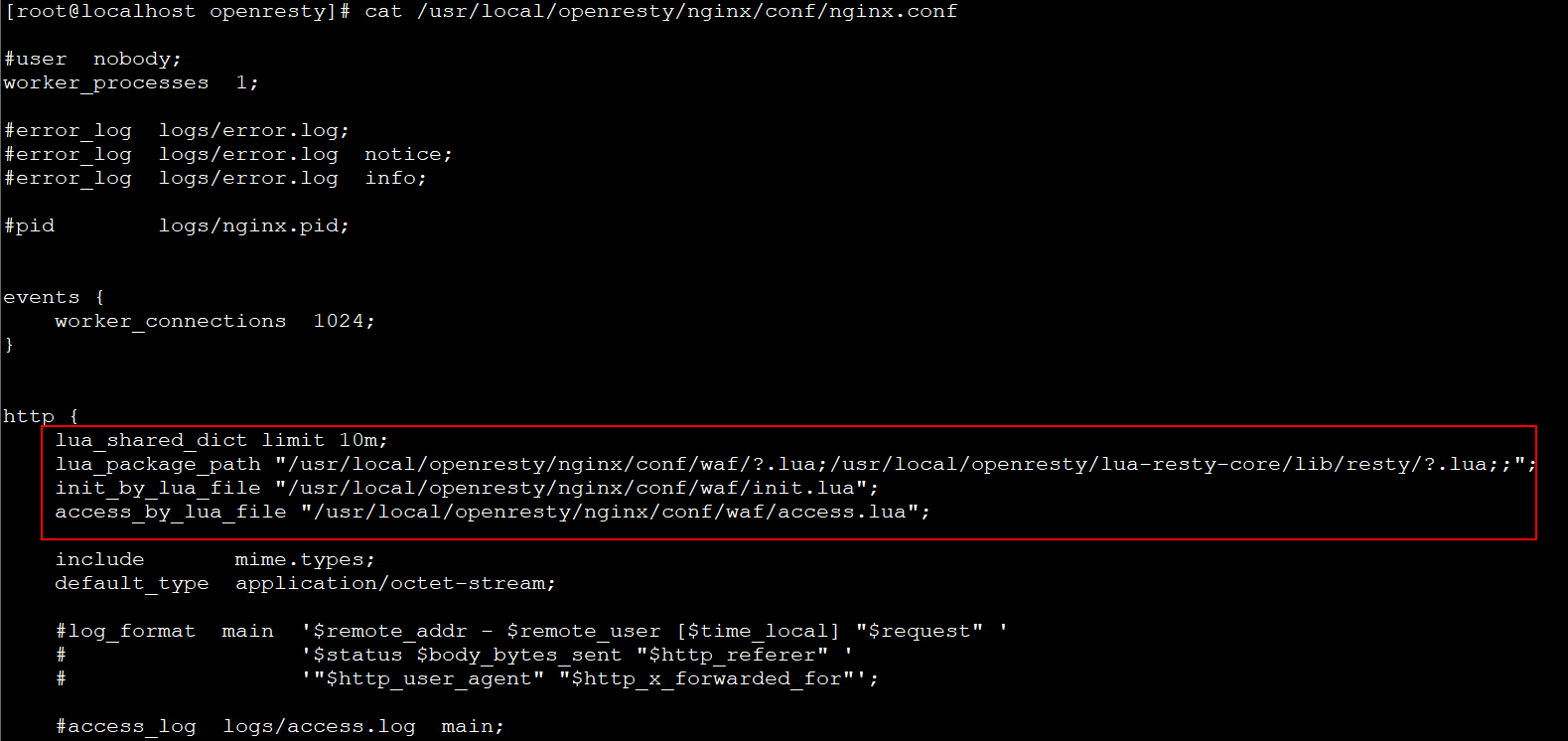
lua\_shared\_dict limit 50m;

lua\_package\_path "/usr/local/openresty/nginx/conf/waf/?.lua";

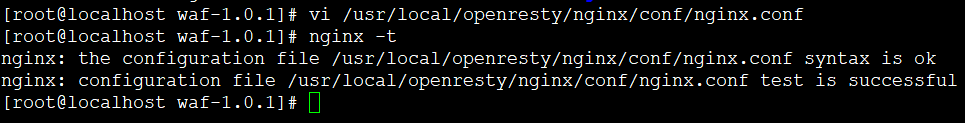
init\_by\_lua\_file "/usr/local/openresty/nginx/conf/waf/init.lua";

access\_by\_lua\_file "/usr/local/openresty/nginx/conf/waf/access.lua";

参考 https://github.com/unixhot/waf



chown -R www.www /usr/local/openresty/nginx



nginx -t #保存后测试配置有没错误

service nginx reload

###### 以下是配置内容，可根据情况自行更改 #####

cat /usr/local/openresty/nginx/conf/waf/config.lua #此为配置文件,可以在里面配置以下内容

config\_waf\_enable = "on" --是否启用waf模块，值为 on 或 off

config\_log\_dir = "/usr/local/openresty/nginx/logs" --waf的日志位置，日志格式默认为json,此文件位置可自行更改，默认/tmp文件夹

config\_rule\_dir = "/usr/local/openresty/nginx/conf/waf/rule-config" --策略规则目录位置，可根据情况变动

config\_white\_url\_check = "on" --是否开启URL检测

config\_white\_ip\_check = "on" --是否开启IP白名单检测

config\_black\_ip\_check = "on" --是否开启IP黑名单检测

config\_url\_check = "on" --是否开启URL过滤

config\_url\_args\_check = "on" --是否开启Get参数过滤

config\_user\_agent\_check = "on" --是否开启UserAgent客户端过滤

config\_cookie\_check = "on" --是否开启cookie过滤

config\_cc\_check = "on" --是否开启cc攻击过滤

config\_cc\_rate = "20/60" --cc攻击的速率/时间；默认单个IP60秒内访问同一个页面超过10次则认为是cc攻击，则自动禁止此IP地址访问此页面60秒(封禁过程中此IP地址依然可以访问其它页面，如果同一个页面访问次数超过10次依然会被禁止)

config\_post\_check = "on" --是否开启POST检测

config\_waf\_output = "html" --定义违反规则的请求跳转到一个自定义html页面（默认值为 html ），即下面的提示网页，也可以设置 “redirect” 值，跳转到其他指定网站页面。

config\_waf\_redirect\_url = "https://www.unixhot.com" --指定违反请求后跳转到的其他网站页面的位置

--指定违反规则后跳转的自定义html页面，下面是内容：

config\_output\_html=[[

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta http-equiv="Content-Language" content="zh-cn" />

<title>网站防火墙</title>

</head>

<body>

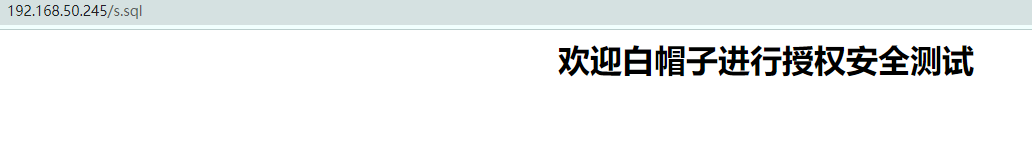
<h1 align="center"> 欢迎白帽子进行授权安全测试，安全漏洞请联系QQ：xxxxxx

</body>

</html>

]]

http://192.168.50.245/s.sql #访问一个sql文件（这里以我的ip为例）



常用：

**IP黑名单配置：需要在config.lua中开启config\_black\_ip\_check = "on"参数**

需开启config\_black\_ip\_check = "on" ，ip加入以下文件，即可返回403被禁止（无需重启，直接生效，格式如下）：

cat /usr/local/openresty/nginx/conf/waf/rule-config/blackip.rule

192.168.31.14

**IP白名单配置:需要在config.lua中开启config\_white\_ip\_check = "on"参数**

cat /usr/local/openresty/nginx/conf/waf/rule-config/whiteip.rule

**CC攻击过滤：需要在config.lua中开启config\_cc\_check = "on"参数，然后指定config\_cc\_rate = "10/60"速率和时间（60秒自动解封，请根据情况自行选择）**

**异常URL策略配置：需在config.lua中开启config\_url\_check = "on"参数然后定义rule-config/url.rule文件，url.rule文件默认为如下，如果匹配到规则的将跳转到由config.lua中config\_waf\_output = "html"参数指定的页面**

cat /usr/local/openresty/nginx/conf/waf/rule-config/url.rule

禁止URL访问包含以下内容的文件，当被访问时跳转到config.lua中指定的页面：

\.(htaccess|bash\_history)

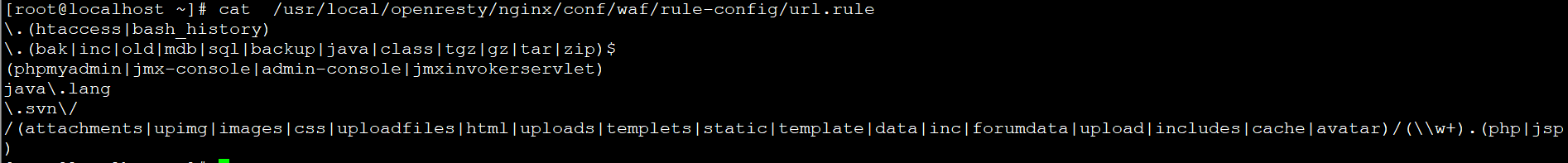
\.(bak|inc|old|mdb|sql|backup|java|class|tgz|gz|tar|zip)$

(phpmyadmin|jmx-console|admin-console|jmxinvokerservlet)

java\.lang

\.svn\/

/(attachments|upimg|images|css|uploadfiles|html|uploads|templets|static|template|data|inc|forumdata|upload|includes|cache|avatar)/(\\w+).(php|jsp)



例，当访问 http://192.168.50.245/1.mdb 时，跳转到以下界面：

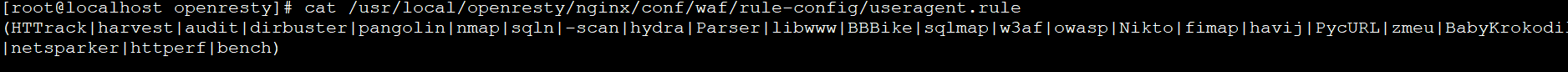


以上文件可根据情况自行增减。

**异常UserAgent策略配置：需要在config.lua中开启config\_user\_agent\_check = "on"参数**

WAF模块中默认封锁了以下UserAgent，如HTTrack网站下载，namp网络扫描，audit网络审计，dirbuster网站目录扫描，pangolin SQL注入工具，scan网络扫描，hydra密码暴力破解，libwww漏洞工具，sqlmap自动SQL注入工具，w3af网络扫描，Nikto Web漏洞扫描等等。

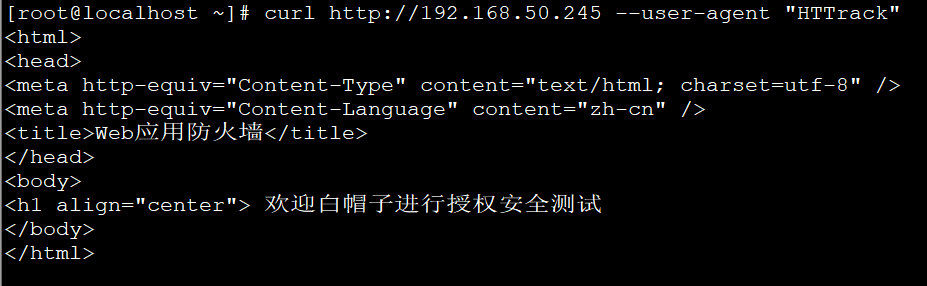
cat /usr/local/openresty/nginx/conf/waf/rule-config/useragent.rule



我们正常访问URL是没问题的，下面来模拟一个非法的UserAgent进行访问.

模拟网站下载：

curl http://192.168.50.245 --user-agent "HTTrack"



模拟nmap网络扫描: curl http://192.168.50.245 --user-agent "nmap"

**异常Get参数策略配置:需要在config.lua配置中开启config\_url\_args\_check = "on"参数**

cat /usr/local/openresty/nginx/conf/waf/rule-config/args.rule

默认封锁了如下：

\.\./

\:\$

\$\{

select.+(from|limit)

(?:(union(.\*?)select))

having|rongjitest

sleep\((\s\*)(\d\*)(\s\*)\)

benchmark\((.\*)\,(.\*)\)

base64\_decode\(

(?:from\W+information\_schema\W)

(?:(?:current\_)user|database|schema|connection\_id)\s\*\(

(?:etc\/\W\*passwd)

into(\s+)+(?:dump|out)file\s\*

group\s+by.+\(

xwork.MethodAccessor

(?:define|eval|file\_get\_contents|include|require|require\_once|shell\_exec|phpinfo|system|passthru|preg\_\w+|execute|echo|print|print\_r|var\_dump|(fp)open|alert|showmodaldialog)\(

xwork\.MethodAccessor

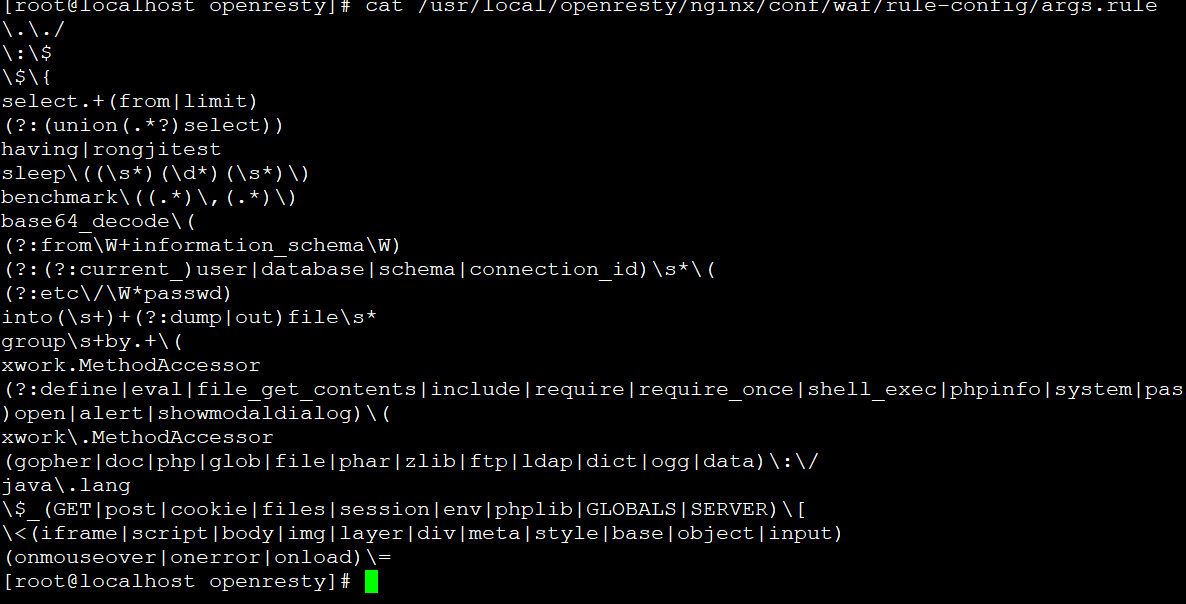
(gopher|doc|php|glob|file|phar|zlib|ftp|ldap|dict|ogg|data)\:\/

java\.lang

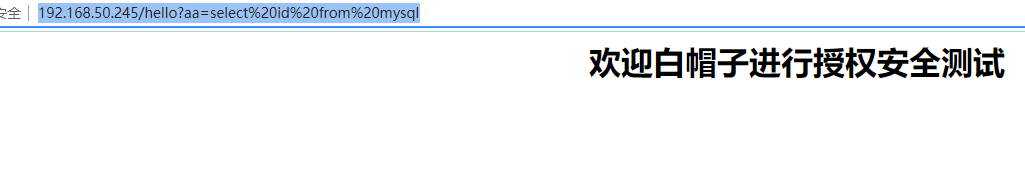
\$\_(GET|post|cookie|files|session|env|phplib|GLOBALS|SERVER)\[

\<(iframe|script|body|img|layer|div|meta|style|base|object|input)

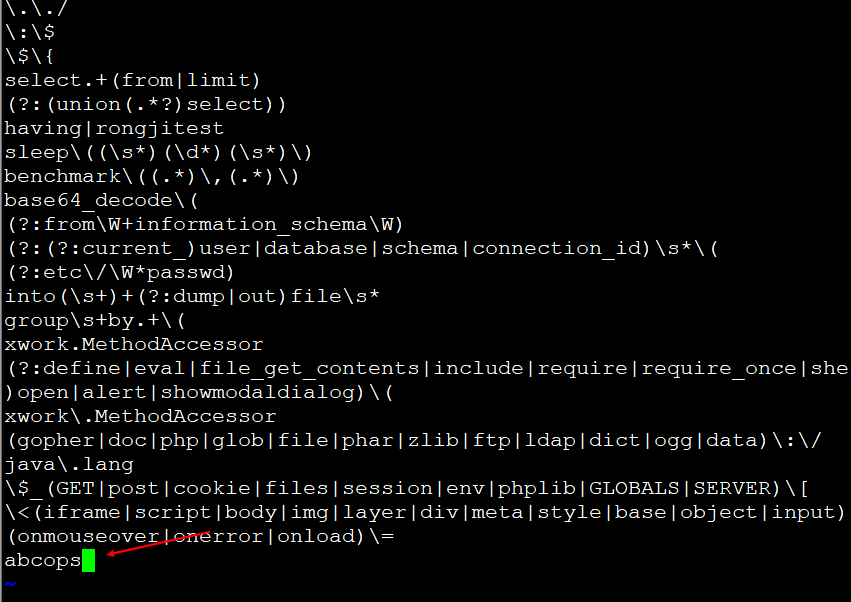
(onmouseover|onerror|onload)\=



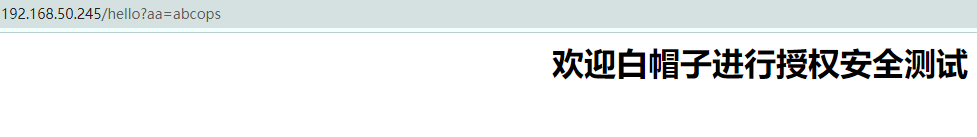
我们进行访问  http://192.168.50.245 /hello?aa=select id from mysql,得到如下，进行匹配:



我们也可以根据自己需求去配置，如下最后添加abcops(无需重启nginx):



然后我们进行访问: http://192.168.50.245/hello?aa=abcops也会匹配到规则



**异常POST参数策略配置：需要在config.lua中开启config\_post\_check = "on"选项。**

cat /usr/local/openresty/nginx/conf/waf/rule-config/post.rule

默认POST请求封禁如下，POST封禁内容与GET相似：

\.\./

select.+(from|limit)

(?:(union(.\*?)select))

having|rongjitest

sleep\((\s\*)(\d\*)(\s\*)\)

benchmark\((.\*)\,(.\*)\)

base64\_decode\(

(?:from\W+information\_schema\W)

(?:(?:current\_)user|database|schema|connection\_id)\s\*\(

(?:etc\/\W\*passwd)

into(\s+)+(?:dump|out)file\s\*

group\s+by.+\(

xwork.MethodAccessor

(?:define|eval|file\_get\_contents|include|require|require\_once|shell\_exec|phpinfo|system|passthru|preg\_\w+|execute|echo|print|print\_r|var\_dump|(fp)open|alert|showmodaldialog)\(

xwork\.MethodAccessor

(gopher|doc|php|glob|file|phar|zlib|ftp|ldap|dict|ogg|data)\:\/

java\.lang

\$\_(GET|post|cookie|files|session|env|phplib|GLOBALS|SERVER)\[

\<(iframe|script|body|img|layer|div|meta|style|base|object|input)

(onmouseover|onerror|onload)\=

参考 <https://www.cnblogs.com/cheyunhua/p/13395745.html>

########## 负载均衡 ########

upstream foo.com {

server 205.185.124.167:80;

server 103.79.79.167:8866;

#server 127.0.0.1:12356 backup;

}

lua\_shared\_dict healthcheck 1m;

lua\_socket\_log\_errors off;

init\_worker\_by\_lua\_block {

local hc = require "resty.upstream.healthcheck"

local ok, err = hc.spawn\_checker{

shm = "healthcheck", -- defined by "lua\_shared\_dict"

upstream = "foo.com", -- defined by "upstream"

type = "http",

http\_req = "GET / HTTP/1.0\r\nHost: foo.com\r\n\r\n",

-- raw HTTP request for checking 定义检测页面

interval = 2000, -- run the check cycle every 2 sec 检测间隔时间

timeout = 2000, -- 1 sec is the timeout for network operations 检测超时时间

fall = 3, -- # of successive failures before turning a peer down 失败3次认为down

rise = 2, -- # of successive successes before turning a peer up 成功2次认为up

valid\_statuses = {200, 302}, -- a list valid HTTP status code 定义http健康状态码

concurrency = 10, -- concurrency level for test requests

}

if not ok then

ngx.log(ngx.ERR, "failed to spawn health checker: ", err)

return

end

-- Just call hc.spawn\_checker() for more times here if you have

-- more upstream groups to monitor. One call for one upstream group.

-- They can all share the same shm zone without conflicts but they

-- need a bigger shm zone for obvious reasons.

}

server

{

listen 80;

server\_name 0.0.0.0;

index index.html index.php;

root html;

location / {

proxy\_pass http://foo.com;

#root html;

#index index.html index.htm;

}

# status page for all the peers:

location = /status {

access\_log off;

#allow 127.0.0.1;

#deny all;

default\_type text/plain;

content\_by\_lua\_block {

local hc = require "resty.upstream.healthcheck"

ngx.say("Nginx Worker PID: ", ngx.worker.pid())

ngx.print(hc.status\_page())

}

}

location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf)$

{

expires 30d;

error\_log /dev/null;

access\_log /dev/null;

}

location ~ .\*\.(js|css)?$

{

expires 12h;

error\_log /dev/null;

access\_log /dev/null;

}

}

访问状态页：

<http://192.168.50.245/status>

