nginx njs利用ipset集合实现黑名单（白名单）访问控制

Published on Jan 23, 2022 UTC by Wang Ye <https://wangye.org>

nginx作为反向代理服务器一直以轻巧高效著称，在日常实践中我将其作为项目的反向代理应用前端并取得了不错的效果，这里记录nginx其中利用njs模块的脚本支持读取ipset黑名单（白名单）从而实现访问控制列表（Access Control List）的方法，做法仅供参考，如有想法欢迎评论提出。

1 主要背景

1.1 问题的提出

互联网环境日益复杂和危险，对于新上线的服务器主机来说，每天都将面临大量的漏洞扫描、DDoS攻击，对于配置不当或者存在安全漏洞的主机，很容易就会沦为跳板机或者挖矿机，继而攻击网络中其他主机或者白白消耗宝贵的计算资源，对于互联网威胁我认为有必要采取积极的防御策略。

1.2 基本的防御策略

主要有这几种手段：① 关注安全公告并及时打补丁避免漏洞；② 复杂的身份认证策略，避免弱口令存在于SSH或者数据库系统中，建议采取证书或者key方式完成认证流程；③ 最小权限划分原则，避免系统超级管理员sa、root、Administrator账号的直接使用，只给当前操作或者应用系统必要的权限；④ 静态防御策略，防火墙只开放必要的端口，对于管理端口只开放白名单IP访问权限或者通过专用Tunnel访问，订阅IP黑名单并加入拦截列表；⑤ 动态防御策略，使用fail2ban、sshguard、蜜罐（honeypot）及入侵检测，动态识别端口扫描，认证尝试等疑似入侵操作，并自动加入拦截列表。

2 ipset与iptables

ipset与iptables均属于由Linux内核提供的Netfilter框架的用户层命令，Netfilter框架提供了诸如包过滤（packet filtering）、网络地址转换（network address translation, NAT）以及端口转换（port translation）等特性，充分利用这些特性将能够较好实现部分防火墙的功能。

2.1 Netfilter防火墙策略配置

Netfilter最简单直观的配置方式便是通过iptables命令（IPv6使用ip6tables命令）配置，例如禁止IP地址`111.111.111.111`访问443端口，可以写成如下：

```
iptables -A INPUT -s 111.111.111.111 -p tcp --dport 443 -j DROP
```

更多的iptables命令的用法我这里就不再举例了，可以参考手册或者互联网上其他专门的文章。

2.2 ipset创建IP或者IP范围的集合

现实场景中往往我们并不满足于屏蔽或者允许一个或者几个IP，比如说我们需要一个IP黑名单，可能这个黑名单里面有成百上千的IP地址，如果说手动去一条一条使用iptables创建记录显然不现实，而且如果这个IP列表可能随时会变动，那么维护Netfilter规则将会变得困难，ipset就是为了解决这个问题而诞生的，通过ipset我们可以建立一个地址列表，再由iptables命令引用地址列表实现拦截或者允许，后续的维护也只是对由ipset创建的列表进行操作就可以了。

创建一个包含黑名单IP地址的列表`blacklist_hosts`：

```
ipset create blacklist_hosts hash:ip
```

创建一条禁止这个地址列表访问443端口的规则：

```
iptables -A INPUT -m set --match-set blacklist_hosts src -p tcp --dport 443 -j RETURN
```

向这个IP地址列表添加或者移除IP地址：

```
# 添加IP地址111.111.111.111
ipset -exist add blacklist_hosts 111.111.111.111

# 移除IP地址111.111.111.111
ipset del blacklist_hosts 111.111.111.111
```

2.3 订阅IP黑名单

如何揪出网络中的坏小子们（Bad Guys），那些不怀好意的攻击者往往有个固定的特征，比如IP地址固定、使用了TOR网络等等，如何获取这些特征呢？通常网络安全研究人员通过设立蜜罐（honeypot）来引诱攻击流量并记录这些attackers的IP地址，如果我们自己设立蜜罐的话，由于我们并没有专业安全机构的庞大蜜罐网络，无法在短时期内收集大量IP地址，资源利用效率也比较低，很不划算，所以我这里选择订阅第三方收集的IP地址黑名单列表，以下介绍一些常用的IP地址黑名单订阅源：

BlockList.de
东北大学网络中心网络威胁黑名单系统

3 nginx利用ipset实现黑白名单访问控制

nginx是一款基于异步框架的网页服务器，也可以用作反向代理、负载平衡和HTTP缓存功能。

3.1 使用访问黑名单

默认情况下nginx提供了简单的基于IP的访问控制功能，比如我们要禁止`1.2.3.4`这个IP地址访问服务器，可以在配置文件中这样写：

```
deny 1.2.3.4;
```

这样配置并不是很灵活，和2.1节iptables所示那样，如果需要屏蔽的IP地址较多，必然会导致配置文件体积变大，nginx加载较慢，另外因为nginx是在启动时一次加载所有配置文件，所以后续对于配置文件的修改无法反应到nginx运行中，需要执行`nginx -s reload`告诉nginx重新加载配置文件，这样也带来了一定的开销。

如果我们已经通过2.3节将IP黑名单订阅到了ipset列表中，那么是否可以直接共享ipset列表，从而不必再维护一套nginx屏蔽列表呢？就我本人而言第一反应是查找是否存在nginx的ipset模块，这样让nginx也和iptables配置的那样可以直接读取ipset列表。

3.2 直接使用nginx ipset模块

使用ipset模块这个想法很好，简单粗暴，经过搜索发现了nginx_ipset_blacklist这个第三方模块，这个模块编译进nginx倒是没有什么问题，但是使用时总是提醒没有root权限，即使将nginx以root身份启动也是一样，经过排查发现这个模块已经11年没有更新，ipset的新版本已经不支持老的调用方式了，有关这点可以参考dnsmasq的代码，其中支持ipset的部分进行了版本判断，较新版本的使用的新的调用API。

后来又查找到另外一个第三方模块nginx_ipset_access_module，简单修改后编译进nginx，在调试时依然达不到要求，这时候猛然想起，作为Web前端程序集成ipset必然要将其升级为较高的权限（例如root），这个是安全实践中是不可取的，有没有一种方式将ipset数据的获取和nginx拦截分离开来呢？

3.3 njs脚本模块使用Shell HTTP服务方式

到这里我想起一个由nginx官方开发的的模块njs，这个模块可以嵌入JavaScript脚本以实现动态的可配置功能，比如官方给出的例子：

① 首先使用`nginx -V`命令检查一下nginx是否已经包含了njs模块，如果没有则需要安装，具体见官方手册，这里不再赘述。

② 在nginx配置文件路径下`/etc/nginx`创建`http.js`，内容如下：

```
function hello(r) {
```

```
    r.return(200, "Hello world!");
}

export default {hello};
```

③ 修改配置文件/etc/nginx/nginx.conf，配置如下：

```
load_module modules/ngx_http_js_module.so;

events {}

http {
    js_import http.js;

    server {
        listen 8000;

        location / {
            js_content http.hello;
        }
    }
}
```

完成上述步骤后，访问:8000端口的根路径不出意外就会得到Hello World！

设想在请求到达backend前由nginx查询ipset列表后再决定是放行还是拦截，这个动作完全可以脚本化，那么可以试试njs实现这样的场景。

首先我们需要为njs脚本提供查询的接口，对于系统管理员来说，查询ipset列表完全可以通过ipset -L 列表名实现，首先我们不考虑将nginx运行在最高权限，njs如何读取到ipset输出呢？

查阅了参考文档，我认为njs有两种方式适合这样的场景，一是HTTP模式；二是文件模式。对于第一种HTTP模式，我们可以在内部启动一个cgi Web服务，该Web服务调用ipset命令获取列表，再将列表封装在HTTP协议中传输给nginx的njs脚本处理；对于第二种方式，我们可以定期运行一个shell脚本，将ipset输出定位到文件，再由njs读取文件获取列表。

3.3.1 HTTP模式的实现方式

能够运行shell的cgi服务端太多了，这里不一一罗列，这里推荐msoap/shell2http这款程序，采用Go语言编写，我个人在实践中依然选择了docker容器的方式。

首先参考官方说明，得知shell2http的调用方式如下（官方示例）：

```
shell2http /date date /ps "ps aux"
```

这样就在:8080端口启动了一个HTTP服务器程序，用浏览器或者curl工具分别访问/date和/ps将会获得date命令和ps aux命令的输出，那么依葫芦画瓢，对于ipset命令HTTP输出也可以通过这样的方式获取：

```
shell2http /ipset "ipset list 列表名"
```

这样访问路径/ipset就能得到指定列表名所包含的输出，当然我们可以让输出再简洁一些，比如仅包含IP地址成员（Members）：

```
shell2http /ipset "ipset list 列表名 | sed -n '/^[a-z0-9]/p' 2>/dev/null"
```

这种写法似乎不够灵活，是否可以将要查询的列表传参呢？shell2http支持-form开关，打开后就可以支持查询字符串（Query String），比如/ipset?list=列表名就会自动建立一个包含列表名的变量$v_list，我们可以创建一个shell脚本命名为ipset_cmd.sh，主要包含内容如下：

```
#!/usr/bin/env bash

if [ -z "$v_list" ]
  then
    exit 1
fi

IPSET=$(which ipset)

while IFS=',' read -ra _LISTS; do
    for _n in "${_LISTS[@]}"; do
      $IPSET list $_n | sed -n '/^[a-z0-9]/p' 2>/dev/null
      echo
    done
done <<< "$v_list"

exit 0
```

将上述脚本`ipset_cmd.sh`放在`/root/`路径下，这样调用shell2http就可以了，需要注意的是此脚本我加入了新的特性以支持多个列表，这样你可以通过`?list=`列表`1`,列表`2`合并多个列表。

```
shell2http -form /ipset /root/ipset_cmd.sh
```

修改刚才的njs脚本例子`http.js`如下：

```
function process(r) {
    ngx.fetch('http://172.15.0.1:8080/ipset?list=blacklist_ipv4,blacklist_ipv6')
        .then(reply => reply.text())
        .then(body => {
            var response = body.split('\n');
            if (!response.length) {
                throw new Error("configuration is not available, request ip:" + r.remoteAddress);
            }

            for (var i = 0; i < response.length; i++) {
                if (response[i] == r.remoteAddress || (
                    r.remoteAddress.indexOf(':') !== -1 &&
                    r.remoteAddress.indexOf(response[i]) === 0
                )) {
                    return true;
                }
            }

            return false;
        })
    .then(is_blacklisted => {
        if (is_blacklisted) {
            r.return(403, "Forbidden, request ip: " + r.remoteAddress);
        } else {
            r.internalRedirect("@backend");
        }
    })
    .catch(e => r.return(500, e));
}

export default {process};
```

解释一下，上面脚本中`172.15.0.1:8080`运行着shell2http以便于查询ipset列表，这里合并查询`blacklist_ipv4`和`blacklist_ipv6`黑名单，获得列表以后即比对访客IP与列表是否匹配，如果命中，则说明IP在黑名单中，返

回Forbidden，否则跳转到nginx配置文件的@backend节，假设要保护的后端服务器地址为172.15.0.11，端口为8081，那么nginx.conf的配置节增加如下：

```
location @backend {
        proxy_pass http://172.15.0.11:8081;
}
```

当然如果每次都请求shell2http并运行一次ipset的话，明显在流量比较大的情况下会给服务器造成负担，而ipset黑名单一般是定期更新的，可以使用shell2http的-cache参数，约定缓存一段时间结果，比如下面的调用将缓存输出约10分钟：

```
shell2http -cache=600 -form /ipset /root/ipset_cmd.sh
```

可能有同学会提出，如果列表很长，这种脚本遍历的方式是否存在性能瓶颈？其实ipset可以直接判断IP地址是否在列表中，通过ipset test 列表名 IP地址这种形式，也就是说我们不必每次拉取整个列表，只需要将目标IP传给ipset，由ipset返回结果就可以了。

例如下面的代码，如果只指定?list=列表名则只会输出列表名，如果指定ip参数，形如?list=列表名&ip=127.0.0.1，则会判断127.0.0.1是否在列表中，如果存在则输出exists，否则什么都不输出。

```
#!/usr/bin/env bash

if [ -z "$v_list" ]
  then
    exit 1
fi

IPSET=$(which ipset)

while IFS=',' read -ra _LISTS; do
    for _n in "${_LISTS[@]}"; do
      if [ -z "$v_ip" ]
        then
         $IPSET list $_n | sed -n '/^[a-z0-9]/p' 2>/dev/null
         echo
        else
           $IPSET test $_n $v_ip >/dev/null 2>&1
          if [[ $? -eq 0 ]]; then
            echo -n "exists"
            break
          fi
        fi
      done
done <<< "$v_list"

exit 0
```

对应的njs脚本代码也仅仅需要判断输出是否存在exists即可。

3.3.2 文件模式的实现方式

文件方式实现的方式虽然不是很优雅，但也高效，首先将ipset命令输出定位到某个文件，建议使用tmpfs文件系统，这样存储在内存中会更加高效，然后njs通过文件API读取这个文件再进行判断，这里的关键点是ipset命令需要周期性执行，所以建议将其加入到计划任务cron中，njs读写文件可以参考下面官方的例子，再修改3.3.1中的示例代码。

```
var fs = require('fs');
var STORAGE = "/tmp/njs_storage"
```

```
function push(r) {
 fs.appendFileSync(STORAGE, r.requestBody);
 r.return(200);
}

function flush(r) {
 fs.writeFileSync(STORAGE, "");
 r.return(200);
}

function read(r) {
 var data = "";
 try {
  data = fs.readFileSync(STORAGE);
 } catch (e) {
 }

 r.return(200, data);
}

export default {push, flush, read};
```

4 总结

到这里算是草草介绍完了，实际上我们可以思考一下便能够给3.3.1的njs代码添加更多的特性，比如将3.3.2节的文件功能与3.3.1节的HTTP请求结合起来，实现nginx文件缓存，避免每次请求shell2http带来的开销（虽然shell2http也能缓存），另外如何阻止自动化机器人和爬虫程序的攻击，可以与fail2ban结合起来，比如通过ngx.log方法，将攻击者IP写入日志，再由fail2ban监控日志获取IP并直接由netfilter网络层拦截等等。njs以其轻便和嵌入式大大便捷了我们对nginx功能的拓展，本文仅仅是通过抛砖引玉，也希望读者能够发挥其更为强大的能力。

External References (Links)

Netfilter
https://zh.wikipedia.org/wiki/Netfilter
蜜罐（honeypot）
https://zh.wikipedia.org/wiki/%E8%9C%9C%E7%BD%90_(%E9%9B%BB%E8%85%A6%E7%A7%91%E5%AD%B8)
BlockList.de
http://www.blocklist.de/en/index.html
东北大学网络中心网络威胁黑名单系统
http://antivirus.neu.edu.cn/ssh/lists/
nginx
https://zh.wikipedia.org/wiki/Nginx
nginx_ipset_blacklist
https://github.com/Vasfed/nginx_ipset_blacklist
nginx_ipset_access_module
https://github.com/mehdi-roozitalab/nginx_ipset_access_module
官方手册
https://nginx.org/en/docs/njs/install.html
msoap/shell2http
https://github.com/msoap/shell2http
Go语言
https://zh.wikipedia.org/wiki/Go
官方的例子
https://github.com/nginx/njs-examples/blob/master/njs/misc/file_io.js
ngx.log

https://nginx.org/en/docs/njs/reference.html