

# 自动封禁恶意访问 IP 方法探究

张 律

(九江职业大学, 江西 九江 332499)

摘要: 为了封禁恶意爬虫程序或者网络攻击程序对服务器的频繁请求, 我们可以建立动态的 IP 黑名单, 对于黑名单内的 IP 地址, 服务器将拒绝提供服务。本文通过介绍几种应用组合来对恶意访问系统的 IP 地址实现封禁, 能够有效地阻止网络攻击。

关键词: 自动封禁; 恶意 IP; 服务器安全

实现 IP 黑名单有多种途径, 例如在操作系统层面, 可以通过配置 Iptables 拒绝指定 IP 的网络请求, 此方式的优点是简单, 可以直接在服务器网络的物理层实现, 但是需要手动操作服务器配置文件; 在 Web Server 层面, 可以通过 Nginx 自身的 deny 选项或者 Lua 插件配置 IP 黑名单, 此方式的优点在于可动态实现封禁 IP, 而且通过设置封禁时间可以做到分布式封禁, 缺点在于操作者对于 Lua 脚本和 Nginx 配置需要比较熟悉; 在应用层面, 可以在客户端请求服务之前检查客户端 IP 是否在黑名单内, 此方式的优点在于代码简单、可维护, 缺点在于代码冗余且高并发会影响性能。

以下从方便、共享、自动化及动态管理的角度出发, 探讨几种解决方案。

## 一、通过 Nginx、Lua、Redis 实现动态封禁 IP

使用 Nginx、Lua、Redis 组合防止 CC 攻击实现的原理在于对于同一外网 IP、同一网址、同一客户端在某一段时间内访问某个网址超过指定次数, 则禁止这个外网 IP 和同一客户端访问这个网址一段时间。

### (一) 高并发环境搭建

本节采用 Nginx、Redis 及 Lua 搭建高并发环境。以下简述搭建要点, 具体步骤及操作细节有很多文献涉及, 本文不再赘述。

首先依次安装及配置 Lua 解释器、Nginx 开发插件 ngx\_devel\_kit、Nginx 的 Lua 模块 lua-nginx-module、Nginx 依赖包 (openssl、zlib、perengine)、Nginx 及 Redis, 注意解决安装过程中的不兼容问题及其他问题, 最后启动运行 Nginx。修改 redis.conf 并以后台形式启动 Redis 服务端, 再启动 Redis 客户端、安装 lua-resty-redis、连接 Redis 连接池后通过修改 Nginx 配置文件及编写 Lua 执行脚本进行整合运行操作。通过调整文件访问数量相关参数如每个子进程允许打开的文件数、调整 Socket 连接相关参数进行高并发优化。使用 Apache 自带的 ab 工具进行压力测试, 在实际测试中, 发生 30 万个请求, 并发

30000, 出错请求为 0, 92% 的请求在 1.4 秒内完成。

如果为了方便, 也可以采用基于 Nginx 与 Lua 的高性能 Web 平台 openresty 直接搭建高并发环境, openresty 内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项, 可用于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

### (二) 自动封禁功能的实现

传统方法只能实现手动添加黑名单进行 IP 封禁, 在无人监控服务器的场景下, 网站服务器被恶意攻击很可能导致服务器资源耗尽崩溃或影响业务。以下使用改进优化后的代码, 可以实现自动将访问频次过高的 IP 地址加入黑名单封禁一段时间, 同时超时后会自动解封, 实现了可管理、自动化及灵活性。

用 Redis 的 key 表示用户, value 表示用户的请求频次, 以下为要求 10 秒内只能访问 10 次请求, 如果超过次数则返回 403 提示为例展示代码。

首先配置 nginx.conf 文件, nginx.conf 部分内容如下:

```
location /frequency {
    access_by_lua_file usr/local/lua/access_by_limit_frequency.lua;
    echo "访问成功";
}
```

编辑 /usr/local/lua/access\_by\_limit\_frequency.lua, 此处假设 Redis 设置了密码, 需要用 red: auth() 方法进行验证, 因篇幅限制, 此处不展示代码。

```
#vim /usr/local/lua/access_by_limit_frequency.lua
```

以上实例测试客户端在 10 秒内访问服务器超出 10 次, 则服务器返回 403 消息, 封禁时间 10 秒到期后, 客户端恢复访问。

在 Nginx 需要限速的 location 中引用上述脚本即可。

对于有大量静态资源文件如 js、css、图片等的前端页面可以设置只有指定格式的请求才进行访问限速, 如果需要在整

个网站都加上限制条件，只需要将 `access_by_lua_file /usr/local/lua/access_by_limit_frequency.lua`；这个配置放在 `server` 部分即可让所有的 `location` 适用。

使用 Nginx、Lua、Redis 组合实现自动化设置 IP 黑名单功能，配置简便，不影响服务器性能，而且多台服务器可以通过 Redis 实例共享黑名单。

## 二、使用 Redis 调用 Iptables 实现封禁恶意 IP

RedisPushIptables 是一个 Redis 模块，用于更新防火墙规则，以在指定的时间内拒绝 IP 地址或永久拒绝，适用任意业务 API 调用，不需要分析应用日志，业务主动调用，缺点是手动编码，不适用普通使用者。

RedisPushIptables 可以让使用者不需要关注 Iptables 的复杂语法，只需要像使用 Redis 那样就可简单使用 Iptables 来阻挡恶意 IP 地址。该模块可以通过 Redis 来操作 Iptables 的 filter 表 INPUT 链规则的增加和删除，可以用来动态调用防火墙。比如用来防御攻击。但因为 Iptables 需要 root 权限，所以要以 root 身份来运行。

使用 `redis.conf` 的 `notify-keyspace-events` 或 `CONFIG SET` 参数启用默认情况下被禁用的键空间事件通知。

```
# redis-cli config set notify-keyspace-events Ex
```

也可以使用以下 `redis.conf` 配置指令加载模块：

```
notify-keyspace-events Ex
```

使用 root 用户运行 `ttliptables` 守护程序：

```
#!/usr/bin/perl
```

```
redis-cli TTL.DROP.INSERT 192.168.1.25 60
```

## 三、Redis 配合 ipset 实现 IP 黑名单的动态添加及解封禁

此方式的设计思路如下：使用 Redis 的 `string` 数据类型保存黑名单，通过设置 `key` 的过期时间，从而实现 IP 黑名单的实时解封禁；订阅 Redis 事件消息的程序通过 Python 进行封装，向 Redis 数据库添加数据时，Redis 发送 `key` 添加事件消息，添加 IP 黑名单，当数据过期时，Redis 发送 `expired` 事件消息，从而解封禁 IP 黑名单；IP 黑名单的动态添加或者删除可以通过 Python 调用 `ipset` 工具实现。

首先在 Redis 配置文件中修改配置项 `notify-keyspace-events` 的值，启用键事件通知模式，然后安装 `ipset` 工具（如在 ubuntu 中使用命令 `sudo apt install ipset`），再打开服务器的防火墙功能。在以上基础上，创建 `ipset` 和 `Iptables` 限制访问规则，在 Redis 中添加需要封禁的 IP 黑名单，使用 Redis 的 `string` 数据类型，`key` 值为 IP，`value` 为空字符串。最后通过设

置 `key` 的过期时间的具体数值，实现 IP 到期解封。在 Nginx 实现 IP 黑名单，在一定程度上，仍然会占用服务器资源，使用 `ipset` 工具进行访问限制会进一步减小对服务器资源的占用。

## 四、使用 PHP 实现 Redis 限制单 IP、单用户的访问次数

通过 `key` 记录 IP：`rate.limiting:$IP`，同时初始时设置期限为 60 秒，如果超时则重新设置，否则进行判断，当一分钟内访问超过 100 次，则禁止访问。但考虑到如果请求的频率在每秒 10 次，每分钟请求 9 次，那么即使它是有问题的访问，但是依然限制不了。因此可以计算该 IP 的 `list` 长度，即该 IP 访问的时间队列，如果小于 10 次，那么将每次访问的时间入栈。否则，当访问次数满了 10 次，计算当前时间与最近一次访问的时间差，如果小于 60 秒，那么禁止访问，否则重置队列，重新存储访问时间。

## 五、其他自动化方式

封禁 IP 持久化到 MySQL 常见的方案是使用 Redis，但如不同的 IP 设置不同的有效期、IP 的 CRUD、统计等等不利于控制。可通过 `lua-nginx-module`，在 Nginx 中开辟一块内存（`lua_shared_dict`），Lua 将封禁 IP 定期从 MySQL 全量刷新至 `lua_shared_dict` 中，所有请求都与 `lua_shared_dict` 中的 IP 对照检查。

具体措施包括安装 `lua-nginx-module` 以及在安装 Nginx 时将 Lua 模块编译进去后，要对 Nginx 进行配置，主要目的一是管理访问 URL 以便可以查看 Nginx 中的 IP 封禁信息，二是设置同步 URL，通过定时任务调用该 URL，实现 IP 黑名单从 MySQL 到 Nginx 的定时刷新，三是进行生产域名配置，需要 IP 封禁功能的 `location` 都要包含以下语句：`access_by_lua_file conf/lua/check_realip.lua`。其次要编写 Lua 脚本，以及进行数据库设计。封禁 IP 可以手工及自动方式，手工方式可以人工查看 `elk` 请求日志，发现恶意 IP 后手工填入 MySQL，自动方式可通过 Python 分析 `elk` 日志，将恶意 IP 自动写入 MySQL。

## 六、结语

本文讨论了几种封禁恶意访问 IP 的方法，读者可以根据自己的需要进行选择，从而保证服务器系统的安全。

## 参考文献：

- [1] 刘伟 .web 网站防 DDOS 攻击方式解析 [J]. 微计算机信息, 2018 (136).
- [2] 顾源 . 分布式令牌桶及高并发限流的实现 [J]. 山东信息技术, 2019 (58).