WAF使用说明

1. 架构说明
   1. 技术实现
      1. WAF使用openresty嵌入nginx来扩展出lua接口。通过lua实现WAF主要功能
   2. 匹配形式
      1. 基于性能考虑，暂时只有基于规则的匹配，还没有基于上下文的规则识别。
   3. 功能
      1. WAF功能分为检测、行为、管理三部分：
         1. 通过配置规则来检测每一条访问。
         2. 匹配规则后首先记录日志，然后根据配置来决定是否拦截。
         3. Nginx返回消息后记录统计数据。
         4. 有单独的配置接口来管理配置。
   4. 存储
      1. Openresty基本原则为尽量避免产生阻塞，因为阻塞发生后会卡掉整个worker。由于mysql产生阻塞的可能性远远高于redis，因此所有配置文件以及可能产生的记录数据全部由redis作为存储。
      2. Openresty有单独的内存存储管理。可以跨worker共享，reload后不会消失，但是master进程restart后会清空。基于性能考虑，会有单独的iworker绑定在第一个worker进程，定时轮询redis读取配置写进共享内存。
      3. WAF运行中产生的需要持久的数据暂时也存储在共享内存。后期可以根据需求持久化到redis。
   5. 管理
      1. WAF配置管理有单独的管理后台。
      2. 一定要使用nginx的认证机制为管理后台添加验证。
      3. 管理后台也是由lua编写，通过nginx设置单独server访问。
      4. 后台全部为api形式访问。
      5. 有一个展示用的html页面，可以使用gui形式访问各个api。可以通过nginx指定单独的server目录访问。
   6. Nginx各阶段Hook流程
      1. Nginx启动会首先声明共享内存,主要是指定内存变量的名称和大小。
      2. Hook掉Init阶段，加载Init脚本，初始化一些启动配置。
      3. 在init\_worker阶段加载iworker脚本，为第一个worker绑定iworker，主要是一些后台的定时任务，例如定时从redis更新配置进内存。
      4. 在单独的server内部需要hook住access阶段，指定access处理脚本，用以执行WAF的核心任务。
   7. 运行流程
      1. WAF目前主要工作在access阶段。
      2. 请求进入access后，会拆分请求体，方便后面的分析。
      3. 将客户IP与内存中的黑白名单匹配，如果命中黑名单则进行阻止行为，如果命中白名单则直接跳过后面的环节。
      4. 从内存取出所有规则依次匹配请求，目前主要匹配请求的uri，useragent，post内容，cookie。
      5. 根据配置的阻拦方式进行命中处理。
         1. 如果全局配置为检测模式，则记录日志然后返回。
         2. 如果全局模式为阻拦模式，规则模式为阻拦模式，则记录并拦截。
         3. 如果全局模式为阻拦模式，规则模式为检测模式，则记录然后放过该次请求。
      6. 记录过程：
         1. 将命中的规则名称，请求详细内容以nginx错误日志的形式记录。
         2. 根据阻拦规则进行拦截或者放过，返回客户端响应，结束掉客户端会话。
         3. 将需要记录的统计数据写进nginx共享内存。
         4. 命中两条以上规则时，如果首先命中的规则已经处于拦截模式，不进行后面的规则匹配。
         5. 命中两条以上规则时，如果首先命中的规则没有处于拦截模式，则记录进行记录，然后继续向后匹配，直到请求被阻拦或者匹配结束。
2. 配置和内存存储说明
   1. WAF存储主要分为redis存储和openresty内存存储。
   2. Redis配置文件在 “lib/init\_config.lua”。
   3. Redis主要存储配置文件，key说明：
      1. Redis：
         1. Key:“waf.conf.deny\_model”:全局的阻拦模式，”1”为阻拦模式，”0”为检测模式。
         2. List: “waf.ip\_deny\_list”: ip黑名单。
         3. List: “waf.ip\_white\_list”: ip白名单。
         4. Key: “waf.rules\_list”: 规则列表，json形式存储所有的规则，为所有规则对象的数组。一个规则对象的格式：

{

“name”: ”规则名称”,

“content”: ”规则内容，正则匹配“,

“model”: ”0 或1，0为检测模式，1为阻拦模式”，

“position”: ”uri、ua、post、cookie，为规则要应用的位置“，

“remark“： ”备注“

}

* + 1. 内存：
       1. Key: ”ip\_deny\_list” ip黑名单。
       2. Key： ”ip\_white\_list” ip白名单。
       3. Key： ”waf\_conf” 配置项。
          1. “waf\_conf” 🡪 “deny\_model” 从redis取到的waf.conf.deny\_model压入。
          2. “waf\_conf” 🡪 “rules” 从redis取到所有的规则压入。
       4. Key： ”waf\_monitor” 监控数据。
          1. “waf\_monitor” 🡪 “monitor.starttime” 记录nginx启动时间
          2. “waf\_monitor” 🡪 “monitor.count.allcount. 1511863753” 每10分钟一个key记录10分钟该时间戳向后10分钟拦截的所有数量，每一次记录incr一次。
          3. “waf\_monitor” 🡪 “monitor.count.规则名. 1511863753” 每条规则每10分钟一个key记录每条规则从该时间戳向后10分钟拦截的所有数量，每一次记录incr一次。

1. 使用说明
   1. 安装openresty。
   2. Nginx配置参见配置文件。两个server，一个是waf进行防护的，一个是管理后台，包括一个web页面的location，指向web目录，一个api的location，直接用脚本hook住，使用lua处理。
   3. 导入redis数据。
   4. 新建一个web目录，将web文件放进去。Web目录在nginx中配置管理后台的时候指定。