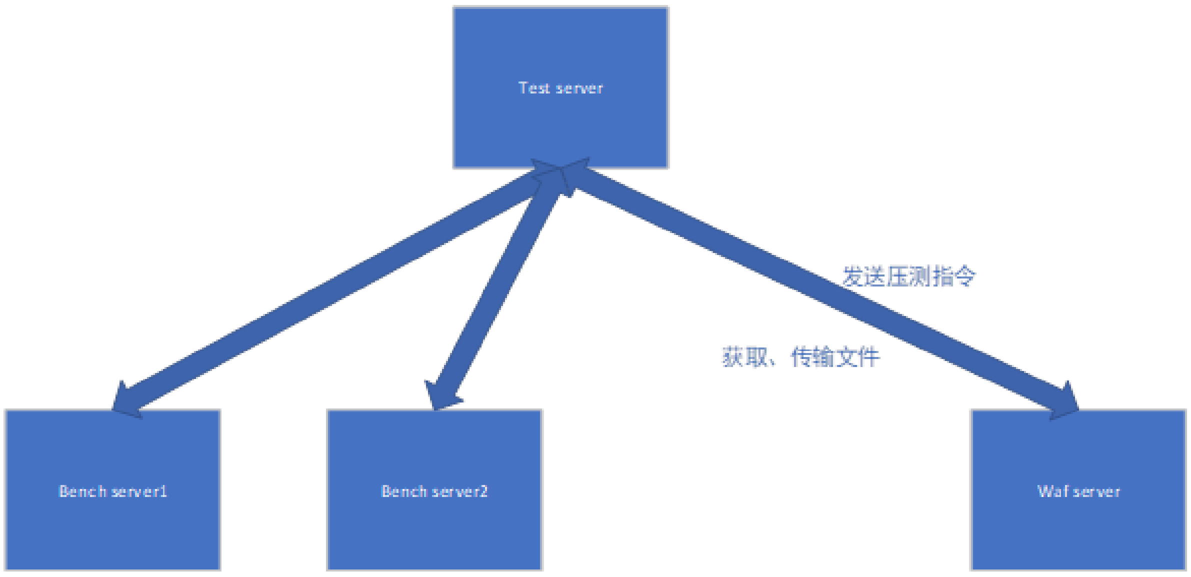
质量测试

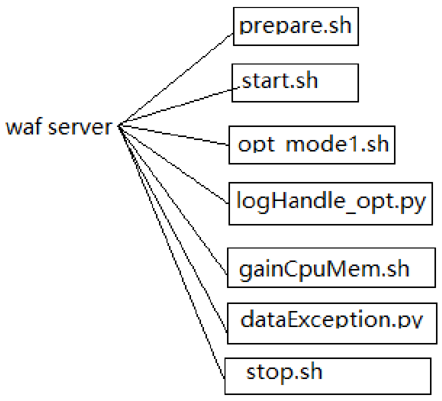
测试目的

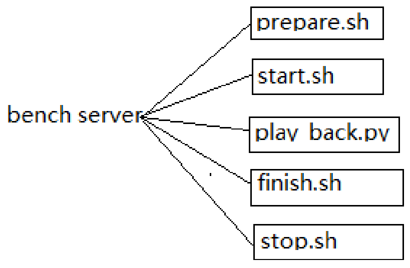
测试基线版waf版本的日志数据情况，对比相同请求数据通过测试版waf版本的日志数据情况，统计异常数据，得到测试版waf的工作情况。

测试构架



脚本框架

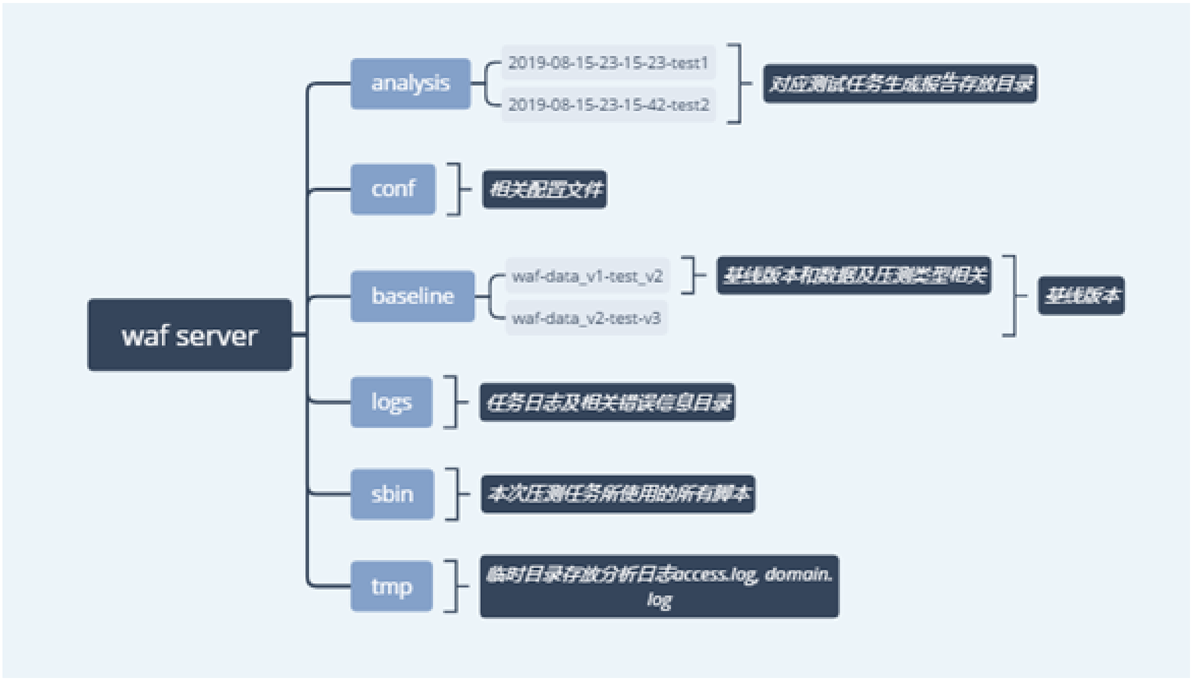




工作内容

waf_server

```
1. 实现内容：接收test_server下发的指令，执行质量测试分析代码，通过对比基线版waf数据和测试版
waf数据，判断测试版waf是否异常。
2. 环境构架
```


waf server
analysis
2019-08-15-23-15-23-test1
2019-08-15-23-15-42-test2
对应测试任务生成报告存放目录
conf
相关配置文件
baseline
waf-data_v1-test_v2
waf-data_v2-test-v3
基线版本和数据及压测类型相关
基线版本
logs
任务日志及相关错误信息目录
sbin
本次压测任务所使用的所有脚本
tmp
临时目录存放分析日志access.log, domain.log

3. 相关代码说明

    1. prepare.sh

```
1.检查当前waf server的版本是否与下发指令的版本一致，如果不一致则安装对应版本的rpm包
2.nginx reload
3.关闭打包程序
4.清空access.log文件
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                        guid
# 压测类型  基线版本对应waf版本  当前需要测试的waf版本  压测数据版本   0: 测试基线
版本, 1: 测试waf版本    每次测试的唯一标识

# 检测当前waf版本是否对应，安装相应的waf版本
current_version=`/usr/local/nginx/sbin/nginx -v 2>&1 |grep 'wswaf'|awk -
F"[ /]" '{print $4}'|awk -F ".el6|.standalone" '{print $1}'`
compare=$(echo $current_version ${3} | awk '$1>$2 {print 1} $1==$2
{print 0} $1<$2 {print 2}')
if test -z $current_version
then
    # 当前机器没有waf版本，安装对应的rpm包
    rpm -ivh /usr/local/waf_test/sbin/wswaf-${3}.el6.x86_64.rpm
elif [ $compare = 1 ]
then
    # 当前机器waf版本高于要安装的版本,需要回退
    rpm -Uvh /usr/local/waf_test/sbin/wswaf-${3}.el6.x86_64.rpm --
oldpackage
elif [ $compare = 2 ]
then
    # 当前机器waf版本低于要安装的版本
    rpm -Uvh /usr/local/waf_test/sbin/wswaf-${3}.el6.x86_64.rpm
else
    echo "correct waf version!"
fi
# 再次检测版本
current_version=`/usr/local/nginx/sbin/nginx -v 2>&1 |grep 'wswaf'|awk -
F"[ /]" '{print $4}'|awk -F ".el6|.standalone" '{print $1}'`
if [ $current_version != ${3} ]
then
    dateStamp=$(date '+%Y-%m-%d:%H:%M:%S')
    echo "{\"error_type\":\"wswaf-${3} install failed \",
\"time\":\"$dateStamp\"}" >>/dev/stderr
    exit 1
fi

# 检查conf文件，替换成标准配置
sed -i '/server   127.0.0.1/c\        server   127.0.0.1:7101
max_fails=0  fail_timeout=30s;' /usr/local/nginx/conf/nginx.conf

# reload
service nginx reload
sleep 60

# 关闭打包程序
sh /usr/local/waf_test/sbin/rmcrontab.sh

# 如果tmp下存在未删掉的signal文件，则删去
if test -e /usr/local/waf_test/tmp/signal
then
    rm -f /usr/local/waf_test/tmp/signal
    echo "signal已删除"
fi

# 清空access.log
```

```
echo -n > /usr/local/nginx/logs/access/access.log

echo "ready to analyze data!"

```

2. start.sh

```
调用opt_mode1.sh脚本开始进行分析
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                       guid
# 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本    0: 测试基线
版本, 1: 测试waf版本    每次测试的唯一标识

# 运行数据分析代码

sh /usr/local/waf_test/sbin/opt_mode1.sh ${1} ${2} ${3} ${4} ${5} ${6}
```

3. opt_mode1.sh

```
1.根据测试种类构造目录, 0表示测试的是基线版, 1表示测试的是测试版
2.不论是基线版还是测试版, 都执行相同的数据分析python脚本
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                     guid
# 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本    0: 测试基线
版本, 1: 测试waf版本    本次测试的唯一标识

# 创建不存在的新测试版本目录

if [ ${5} = 1 ]
then
    mkdir /usr/local/waf_test/analysis/${6}-${1}
    mkdir /usr/local/waf_test/analysis/${6}-${1}/mem
    mkdir /usr/local/waf_test/analysis/${6}-${1}/mem/baseline
    mkdir /usr/local/waf_test/analysis/${6}-${1}/mem/test
    mkdir /usr/local/waf_test/tmp/${6}-${1}
    mkdir /usr/local/waf_test/tmp/${6}-${1}/bak  # 存放测试的access.log
    mkdir /usr/local/waf_test/analysis/${6}-${1}/cpu
    mkdir /usr/local/waf_test/analysis/${6}-${1}/cpu/baseline
    mkdir /usr/local/waf_test/analysis/${6}-${1}/cpu/test
    mkdir /usr/local/waf_test/tmp/${6}-${1}/error_status  # 存放
domain.log
    cp -rf /usr/local/waf_test/baseline/${2}-${4}-${1}/cpu/*
/usr/local/waf_test/analysis/${6}-${1}/cpu/baseline
    cp -rf /usr/local/waf_test/baseline/${2}-${4}-${1}/mem/*
/usr/local/waf_test/analysis/${6}-${1}/mem/baseline
    mkdir /usr/local/waf_test/analysis/${6}-${1}/domain
    mkdir /usr/local/waf_test/analysis/${6}-${1}/domain/baseline
    mkdir /usr/local/waf_test/analysis/${6}-${1}/domain/test
else
```

```
    if [ -d "/usr/local/waf_test/baseline/${2}-${4}-${1}" ]
    then
        rm -rf /usr/local/waf_test/baseline/${2}-${4}-${1}
        rm -rf /usr/local/waf_test/tmp/${2}-${4}-${1}
        echo '基线版本已清空'
    fi
    mkdir /usr/local/waf_test/baseline/${2}-${4}-${1}
    mkdir /usr/local/waf_test/baseline/${2}-${4}-${1}/mem
    mkdir /usr/local/waf_test/baseline/${2}-${4}-${1}/cpu
    mkdir /usr/local/waf_test/tmp/${2}-${4}-${1}
    mkdir /usr/local/waf_test/tmp/${2}-${4}-${1}/bak
    mkdir /usr/local/waf_test/tmp/${2}-${4}-${1}/error_status
fi

# 每隔一分钟取日志的循环在这个Python脚本中执行,直接得到全部域名的状态码统计
python2.7 /usr/local/waf_test/sbin/logHandle_opt.py ${1} ${2} ${3} ${4}
${5} ${6}
```

4. logHandle_opt.py

```
1. 每隔一分钟从access.log拉取访问日志
2.统计源站的响应状态码为'-'而waf状态码不为'-'的日志，目前只统计['400', '403',
'408', '499', '500', '502', '503']这几种状态的数量，并且按域名为文件名记录这种异
常日志
3.同时每隔一分钟执行一次gainCpuMem.sh获取nginx各个进程的CPU和MEM
```

```
#!/usr/bin/python2.7
#coding=utf-8

import pandas as pd
import json
import os
import time
import sys
import logging
import commands


logging.basicConfig(filename =
"/usr/local/waf_test/logs/waf_test.log",level = logging.INFO, filemode =
'a', format = '%(asctime)s - %(levelname)s: %(message)s')


if __name__ == '__main__':
    # 获取压测版本
    mode = sys.argv[1]
    baseline_version = sys.argv[2]
    waf_version = sys.argv[3]
    data_version = sys.argv[4]
    operation = sys.argv[5]
    guid = sys.argv[6]

    # 获取对应版本的路径，文件需要放的位置
    if operation == "0":
        path = '/usr/local/waf_test/baseline/' + waf_version + '-' +
data_version + '-' + mode
```

```
    else:
        path = '/usr/local/waf_test/analysis/' + guid + '-' + mode

    errorDic = {}  # 按域名存储错误日志
    total_dic = {}  # 按域名存储状态码信息
    format_error = []
    header = ['wafStatus', 'original_status', 'channel']
    total = ['400', '403', '408', '499', '500', '502', '503']

    while True:
        res = []
        if os.path.exists('/usr/local/nginx/logs/access/access.log') and
os.path.getsize('/usr/local/nginx/logs/access/access.log'):
            # 拉取日志
            try:
                os.system('rm -f /usr/local/waf_test/tmp/tmp/*')
                os.system('mv /usr/local/nginx/logs/access/access.log
/usr/local/waf_test/tmp/tmp/access.log')
                print('拉取日志成功')
            except Exception as msg:
                logging.error(str(msg))

            # 通知进程重新产生日志
            count = 0
            while count < 3:
                _, pID = commands.getstatusoutput('cat
/usr/local/nginx/logs/nginx.pid')
                os.system('kill -s USR1 ' + pID)
                time.sleep(5)
                if
os.path.exists('/usr/local/nginx/logs/access/access.log'):
                    print('access.log regenerate succeed')
                    break
                count += 1
            if count == 3:
        print('access.log regenerate failed')
                logging.error('access.log regenerate failed')

            # 对本次拉取的日志进行分析
            with open("/usr/local/waf_test/tmp/tmp/access.log", "r") as
f:
                for line in f:
                    if line.find('pepp4_') == -1:
                        try:
                            lineList = []
                            index1 = line.index('HTTP/')
                            index2 = line[index1 + 10:].index(' ')
                            Li = line[index1:index1 + 10 +
index2].split()
                            index3 = line.index('WAF_')
                            li = line[index3:].split()
                            # 保存异常状态的日志信息，按域名分类
                            if li[5] == '-' and Li[1] in total:
                                lineList.append(Li[1])
                                lineList.append(li[5])
                                lineList.append(li[6])
                                res.append(lineList)
                                if lineList[2] in errorDic:
                                    errorDic[lineList[2]].append(line)
                                else:
                                    errorDic.setdefault(lineList[2],
[line])
                        except (Exception, ValueError):
                            format_error.append(line)

            pf = pd.DataFrame(res, columns=header)
            typeList = pf['channel'].unique()
            # 生成所有状态码统计文件
            if len(pf) > 0:
                for Type in typeList:
                    pf_ii = pf[pf['channel'].isin([Type])]
                    Se = pf_ii['wafStatus'].value_counts()
                    statusCode = eval(Se.to_json())
                    # 合并域名相同的状态码数量
                    if Type in total_dic:
                        for key in statusCode:
                            if key in total_dic[Type]:
                                total_dic[Type][key] += statusCode[key]
                            else:
                                total_dic[Type][key] = statusCode[key]
                    else:
                        total_dic[Type] = statusCode

            # 获取时间戳
            time_tup = time.localtime(time.time())
            format_time = '%Y%m%d%H%M%S'
            dateStamp = time.strftime(format_time, time_tup)
            if operation == "0":
                os.system('mv /usr/local/waf_test/tmp/tmp/access.log
/usr/local/waf_test/tmp/' + waf_version + '-' + data_version + '-' +
mode + '/bak/access.log.' + dateStamp)
            else:
                os.system('mv /usr/local/waf_test/tmp/tmp/access.log
/usr/local/waf_test/tmp/' + guid + '-' + mode + '/bak/access.log.' +
dateStamp)

        # 获取CPU及MEM,传参为测试类别和测试版本
            os.system('sh /usr/local/waf_test/sbin/gainCpuMem.sh' + ' '
+ mode + ' ' + baseline_version + ' ' + waf_version + ' ' + data_version
+ ' ' + operation + ' ' + guid)


        # 程序结束的标志
        if os.path.exists('/usr/local/waf_test/tmp/signal') and not
os.path.getsize('/usr/local/nginx/logs/access/access.log'):
            break

        # 每隔一分钟取一次日志
        print('sleep一分钟')
        time.sleep(60)

    # 按域名记录所有异常日志
    if operation == "0":
        pa = '/usr/local/waf_test/tmp/' + baseline_version+'-
'+data_version+'-'+mode
    else:
        pa = '/usr/local/waf_test/tmp/' + guid + '-' + mode
    for key in errorDic:
        with open(pa + '/error_status/' + key, 'w') as fp:
            for error in errorDic[key]:
                fp.write(error)
    print('异常日志记录成功')

    # 按域名记录状态码信息
    with open(path + '/totalData.json', 'w') as f:
        f.write(json.dumps(total_dic, indent=2) + '\n')
    print('状态码记录成功')

    # 记录格式错误的日志信息
    with open(path + '/format_error.log', 'w')as fp:
        for log in format_error:
            fp.write(log)
    print('格式错误日志记录成功')
```

5. gainCpuMem.sh

> 按进程号获取nginx进程的CPU和MEM,存储为PID.CPU/PID.MEM的形式，同时统计每一分钟所有进程的CPU和MEM之和

```
#!/bin/bash

# test_type baseline_version   waf_version          analysis_version
operation                      guid
# 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本    0: 测试基线
版本，1: 测试waf版本   本次测试的唯一标识

totalC=0
totalM=0
PID_array=`ps -ef|grep nginx|grep worker|grep -v nobody|awk '{print
$2}'`
for PID in $PID_array
do
    if test -n $PID
    then
        CPU=`top -b -p $PID -n 1|grep nginx|awk '{print strtonum($(NF-
4))}'`
        MEM=`top -b -p $PID -n 1|grep nginx|awk '{print strtonum($(NF-
3))}'`
        totalC=$(echo "scale=2; $totalC+$CPU" | bc)
        totalM=$(echo "scale=2; $totalM+$MEM" | bc)
        if [ ${5} = 0 ]
        then
            echo $CPU >>
/usr/local/waf_test/baseline/${2}-${4}-${1}/cpu/$PID.CPU
            echo $MEM >>
/usr/local/waf_test/baseline/${2}-${4}-${1}/mem/$PID.MEM
        else
            echo $CPU >>
/usr/local/waf_test/analysis/${6}-${1}/cpu/test/$PID.CPU
            echo $MEM >>
/usr/local/waf_test/analysis/${6}-${1}/mem/test/$PID.MEM
```

```
        fi
    fi
done

# 存储总的MEM和CPU
if [ ${5} = 0 ]
then
    echo $totalC >>
/usr/local/waf_test/baseline/${2}-${4}-${1}/cpu/totalC
    echo $totalM >>
/usr/local/waf_test/baseline/${2}-${4}-${1}/mem/totalM
else
    echo $totalC >>
/usr/local/waf_test/analysis/${6}-${1}/cpu/test/totalC
    echo $totalM >>
/usr/local/waf_test/analysis/${6}-${1}/mem/test/totalM
fi
```

6. dataException_opt.py

```
1.在测试种类为测试版本的时候调用
2.对比基线版的统计数据和测试版的统计数据，如果测试版的相同域名的相同异常状态码比基线版
的数量多出10%的话，就记录下来，作为测试结果
```

```
#!/usr/bin/python2.7
#coding=utf-8


from __future__ import division
import json
import sys
import logging
import os

logging.basicConfig(filename =
"/usr/local/waf_test/logs/waf_test.log",level = logging.INFO, filemode =
'a', format = '%(asctime)s - %(levelname)s: %(message)s')


if __name__ == '__main__':
    mode = sys.argv[1]
    baseline_version = sys.argv[2]
    waf_version = sys.argv[3]
    data_version = sys.argv[4]
    operation = sys.argv[5]
    guid = sys.argv[6]
    # 获取对应版本的路径，获取文件的位置
    baselinePath = '/usr/local/waf_test/baseline/' + baseline_version+'-
'+data_version+'-'+mode
    testPath = '/usr/local/waf_test/analysis/' + guid + '-' + mode

    dic_normal = {}
    dic_test = {}
    # 基准版
    try:
        with open(baselinePath + "/totalData.json", "r") as f:
            dic_normal = json.load(f)
    except IOError as msg:
        logging.error(str(msg))
    # 测试版
    try:
        with open(testPath + "/totalData.json", "r") as fp:
            dic_test = json.load(fp)
    except IOError as msg:
        logging.error(str(msg))

    total_dic = {}
    with open(testPath + '/report.txt', 'a+') as fp:
        for key in set(dic_normal.keys()) | set(dic_test.keys()):
            if key in dic_normal and key not in dic_test:
                total_dic[key] = dic_normal[key]
            elif key in dic_test and key not in dic_normal:
                total_dic[key] = dic_test[key]
            else:
                for Key in (set(dic_normal[key].keys()) |
set(dic_test[key].keys())):
                    if Key in dic_test[key] and Key not in
dic_normal[key]:
                        total_dic[key] = dic_test[key]
                        break
                    elif Key in dic_test[key] and Key in
dic_normal[key]:
                        if (dic_test[key][Key] - dic_normal[key][Key]) /
dic_normal[key][Key] > 0.1:
                            total_dic[key] = [dic_normal[key],
dic_test[key]]
                            break
        fp.write(json.dumps(total_dic, indent=2, ensure_ascii=False) +
'\n')

    if total_dic:
        for key in total_dic:
            try:
        if os.path.exists('/usr/local/waf_test/tmp/'+baseline_version+'-
'+data_version+'-'+mode+'/error_status/'+key):
                    os.system('cp -rf
/usr/local/waf_test/tmp/'+baseline_version+'-'+data_version+'-
'+mode+'/error_status/'+key+' '+testPath+'/domain/baseline/')
                if os.path.exists('/usr/local/waf_test/tmp/' + guid + '-
' + mode + '/error_status/' + key):
            os.system('cp -rf /usr/local/waf_test/tmp/' + guid + '-' +
mode + '/error_status/' + key + ' ' + testPath + '/domain/test/')
            except Exception as msg:
                logging.error(msg)
```

7. finish.sh

```
1.waf server测试完成后调用
2.创建tmp目录下的guid文件，作为logHandle_opt.py结束的标志
3.打包测试结果，发送对应的目录的json文件
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                      guid
# 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本    0: 测试基线
版本, 1: 测试waf版本    本次测试的唯一标识

# 在tmp目录下创建signal文件, 作为logHandle.py结束的信号
touch /usr/local/waf_test/tmp/signal
sleep 100

echo -n > /usr/local/waf_test/tmp/info.json
if [ ${5} = 0 ]
then
    echo "{\"result\":\"success\"}" > /usr/local/waf_test/tmp/info.json
else
    # 基准版和测试版进行对比
    if [ -d "/usr/local/waf_test/baseline/${2}-${4}-${1}" ]
    then
        python2.7 /usr/local/waf_test/sbin/dataException_opt.py ${1}
${2} ${3} ${4} ${5} ${6}
    else
        dateStamp=$(date '+%Y-%m-%d:%H:%M:%S')
        echo "{\"error_type\":\"wswaf-${2} doesn't exist \",
\"time\":\"$dateStamp\"}" >> /dev/stderr
    fi
    tar zcPf /usr/local/waf_test/analysis/${6}-${1}/upload.tar
/usr/local/waf_test/analysis/${6}-${1}/format_error.log
/usr/local/waf_test/analysis/${6}-${1}/report.txt
/usr/local/waf_test/analysis/${6}-${1}/cpu
/usr/local/waf_test/analysis/${6}-${1}/mem
/usr/local/waf_test/analysis/${6}-${1}/domain
    echo "{\"result\":\"success\",
\"file_path\":\"/usr/local/waf_test/analysis/${6}-${1}/upload.tar\"}" >
/usr/local/waf_test/tmp/info.json
fi


curl -X POST http://127.0.0.1:60001/finished_bench -H "content-
type:application/json" -T /usr/local/waf_test/tmp/info.json
echo "finished"
```

8. stop.sh

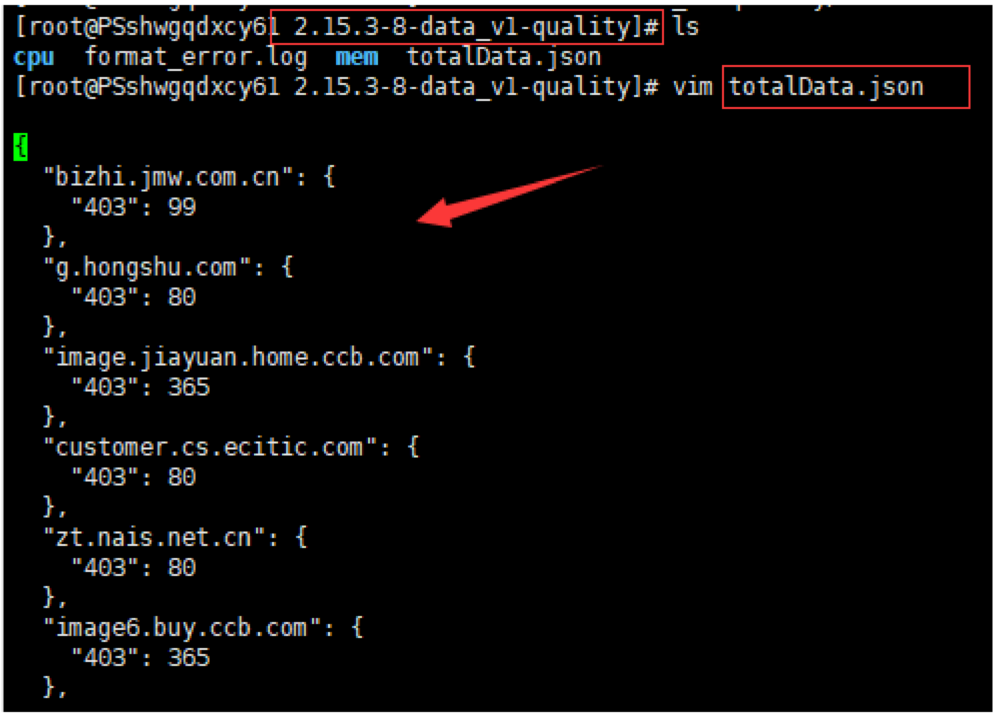
```
终止waf server端的所有测试脚本
```

```
#!/bin/bash

# test_type baseline_version    waf_version          data_version
operation                        guid
# 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本     0：测试基线
版本，1：测试waf版本     本次测试的唯一标识

kill `ps -ef|grep ${6}|grep -v grep|awk '{print $2}'`
echo "测试已中止"
```

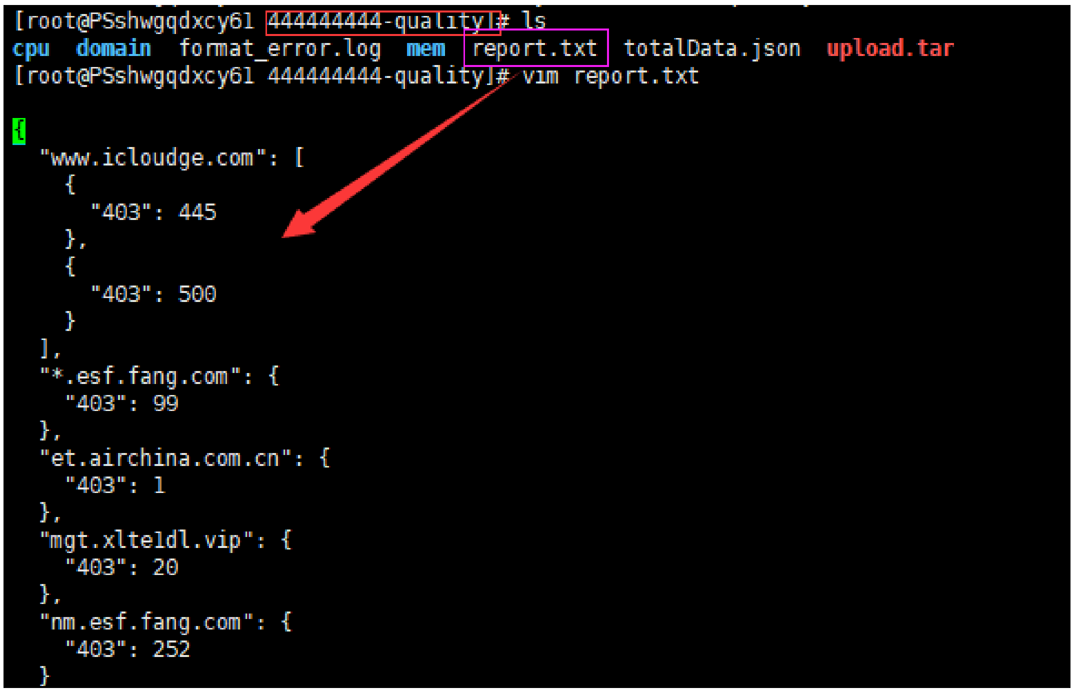
4. 测试结果
   1. 基线版



```
[root@PSshwgqdxcy61 2.15.3-8-data_v1-quality]# ls
cpu  format_error.log  mem  totalData.json
[root@PSshwgqdxcy61 2.15.3-8-data_v1-quality]# vim totalData.json

{
  "bizhi.jmw.com.cn": {
    "403": 99
  },
  "g.hongshu.com": {
    "403": 80
  },
  "image.jiayuan.home.ccb.com": {
    "403": 365
  },
  "customer.cs.ecitic.com": {
    "403": 80
  },
  "zt.nais.net.cn": {
    "403": 80
  },
  "image6.buy.ccb.com": {
    "403": 365
  },
```

   2. 测试版

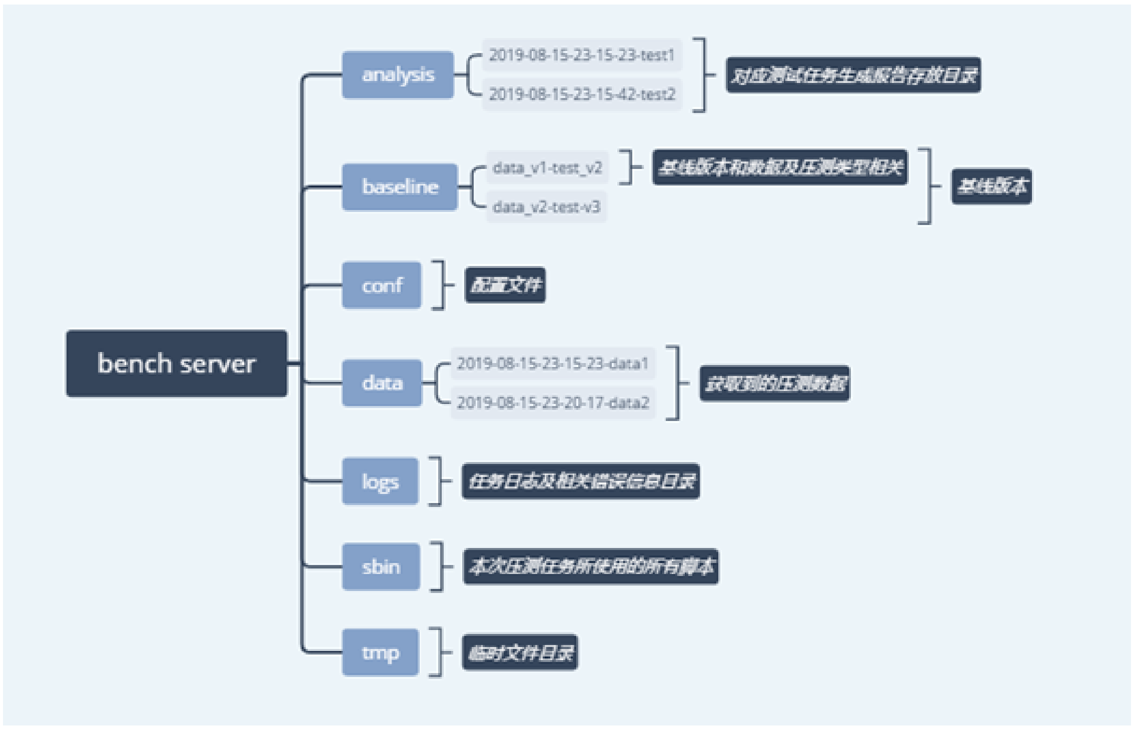


```
[root@PSshwgqdxcy61 444444444-quality]# ls
cpu  domain  format_error.log  mem  report.txt  totalData.json  upload.tar
[root@PSshwgqdxcy61 444444444-quality]# vim report.txt

{
  "www.icloudge.com": [
    {
      "403": 445
    },
    {
      "403": 500
    }
  ],
  "*.esf.fang.com": {
    "403": 99
  },
  "et.airchina.com.cn": {
    "403": 1
  },
  "mgt.xlteldl.vip": {
    "403": 20
  },
  "nm.esf.fang.com": {
    "403": 252
  }
```

bench_server

1. 实现内容：接收test_server下发的指令，向waf服务器发送请求
2. 环境构架


bench server
analysis
2019-08-15-23-15-23-test1
2019-08-15-23-15-42-test2
对应测试任务生成报告存放目录
baseline
data_v1-test_v2
data_v2-test-v3
基线版本和数据及压测类型相关
基线版本
conf
配置文件
data
2019-08-15-23-15-23-data1
2019-08-15-23-20-17-data2
获取到的压测数据
logs
任务日志及相关错误信息目录
sbin
本次压测任务所使用的所有脚本
tmp
临时文件目录

3. 相关代码说明
   1. prepare.sh

      ```
      探测waf服务器响应状态是否成功
      ```

      ```bash
      #!/bin/bash

      # test_type baseline_version   waf_version          data_version
      operation                         guid
      # 压测类型   基线版本对应waf版本   当前需要测试的waf版本   压测数据版本    0: 测试基线
      版本，1: 测试waf版本    每次测试的唯一标识

      status=`curl http://121.46.247.123/testcdn.htm -H "Host:lvs.lxdns.net" -
      i|sed -n 1p|awk '{print $2}'`
      if [ $status = '200' ]
      then
          echo 'connect successfully'
      else
          dateStamp=$(date '+%Y-%m-%d:%H:%M:%S')
          echo "{\"error_type\":\"connect wafserver failed \",
      \"time\":\"$dateStamp\"}" >>/dev/stderr
          exit 1
      fi

      echo 'ready to send data!'
      ```

   2. start.sh

```
调用发送数据的脚本
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                          guid
# 压测类型  基线版本对应waf版本  当前需要测试的waf版本  压测数据版本   0: 测试基线
版本, 1: 测试waf版本    每次测试的唯一标识

python /usr/local/waf_test/sbin/play_back.py 121.46.247.123

sh /usr/local/waf_test/sbin/finish.sh
```

3. play_back.py

```
向waf服务器发送数据
```

```
#!/usr/bin/python
#coding=utf-8

import logging
import
os,sys,time,socket,threading,copy,StringIO,cPickle,subprocess,fcntl,stru
ct,signal,shutil,commands,logging,datetime,json,platform,re
import re,ConfigParser
import time
#import redis
import Queue
import thread



work_dir = "/usr/local/waf_test/"
nginx_work_dir = "/usr/local/nginx/conf/servers/"
logging.basicConfig(filename = work_dir + "logs/waf_test.log",level =
logging.INFO, filemode = 'a', format = '%(asctime)s - %(levelname)s: %
(message)s')
data_dir = work_dir + "datas/"
worker_num = 5
exit_worker = 0
status_list={}
total_num=0
ok_num=0
err_num=0
use_time=0
q = Queue.Queue(maxsize = 0)
lock=threading.Lock()
conf_list=[]

def main():
    global use_time
    #file_path=sys.argv[1]
    ip=sys.argv[1]
    init_conf_list()
    init_data_list()
    #sock=connect_web(ip)
    start_time=time.time()
    init_work_list(ip)
    #do_replay(sock,file_path,ip)
    while 1:
        if exit_worker == worker_num:
            break
        time.sleep(1)
    end_time=time.time()
    use_time= end_time - start_time
    play_summary(use_time)


def init_data_list():
    try:
        dir_list = os.listdir(data_dir)
        if not dir_list:
            return
        else:
            for file_name in dir_list :
                q.put(data_dir + file_name)
    except Exception, e:
        logging.error(e)

    time.sleep(1)

def init_conf_list():
    global conf_list
    try:
        dir_list = os.listdir(nginx_work_dir)
        if not dir_list:
            return
        else:
            for file_name in dir_list :
                server_name = file_name[:-5]
                conf_list.append(server_name)
    except Exception, e:
        logging.error(e)


def init_work_list(ip):
    thread.start_new_thread(replay_work, (ip,""))
    thread.start_new_thread(replay_work, (ip,""))
    thread.start_new_thread(replay_work, (ip,""))
    thread.start_new_thread(replay_work, (ip,""))
    thread.start_new_thread(replay_work, (ip,""))

def replay_work(ip,data):
     global exit_worker
     global lock

     while q.empty() == False:
         try:
             file_path = q.get()
             do_replay(ip,file_path)
         except Exception, e:
             logging.error(e)
```

```
      print "no file"
      if lock.acquire():
           exit_worker=exit_worker+1
           lock.release()

def do_replay(ip,file_path):
    global total_num
    split_num = 0;
    request_data=""
    print file_path
    with open(file_path,"r") as f:
        for line in f.readlines():
            if line=="====****\n":
                #print re.sub("Host:.*\r\n","Host:
www.linsd.com\r\n",request_data)
                send_http_request(request_data,ip)
                #request_data = request_data + "###@@@"
                #total_num=total_num+1
                #split_num = split_num + 1
                #if split_num == 100:
                #   thread.start_new_thread(replay_work,
(ip,request_data))
                #   split_num = 0
                request_data = ""
            else:
                request_data = request_data + line


def fPopen(aCmd):
    p=subprocess.Popen(aCmd, shell=True,
bufsize=4096,stdin=subprocess.PIPE, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, close_fds=True)
    sOut = p.stdout.read()
    sErr = p.stderr.read()
    return (sOut,sErr)

def send_http_request(request_data,ip):
    for i in conf_list:
        url = "https://" + i + "/"
        request_data = re.sub("https://(.+?)/",url,request_data)
        host_name = "Host: "+ i + "\r\n"
        #print host_name
        request_data = re.sub("Host:.*\r\n",host_name,request_data)
        #print request_data
        send_request(request_data,ip)


def send_request(data,ip):
    global ok_num
    global err_num
    global status_list
    global lock
    try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((ip, 9101))
            http_request = data
            sock.send(http_request)
            rev_data = sock.recv(1024)
```

```
                status = rev_data[9:12]
                print status
                if lock.acquire():
                    if status_list.has_key(status):
                        status_list[status]=status_list[status] + 1
                    else:
                        status_list[status]=1
                    ok_num=ok_num+1
                    print str(ok_num)
                    lock.release()

                sock.close()
        except socket.error, msg:
                err_num=err_num+1
                logging.error(msg)
        except Exception, e:
                err_num=err_num+1
                logging.error(e)
def play_summary(use_time):
    print status_list
    print "total_num: " + str(total_num)
    print "ok_num: " + str(ok_num)
    print "err_num: " + str(err_num)
    print "use_time: " + str(use_time)

if __name__ == "__main__":
    main()
    sys.exit(0)
```

4. finish.sh

```
发送数据结束，向test server发送result:success消息
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                      guid
# 压测类型  基线版本对应waf版本  当前需要测试的waf版本  压测数据版本   0：测试基线
版本，1：测试waf版本    每次测试的唯一标识

curl -X POST http://localhost:60002/finished_bench -H "content-
type:application/json" -d "{\"result\": \"success\"}"
echo "finished"
```

5. stop.sh

```
终止waf server端的所有相关测试进程
```

```
#!/bin/bash

# test_type baseline_version   waf_version          data_version
operation                          guid
# 压测类型  基线版本对应waf版本  当前需要测试的waf版本  压测数据版本   0: 测试基线
版本, 1: 测试waf版本    每次测试的唯一标识

kill `ps -ef|grep ${6}|grep -v grep|awk '{print $2}'`
echo "测试已中止"
```

压力测试

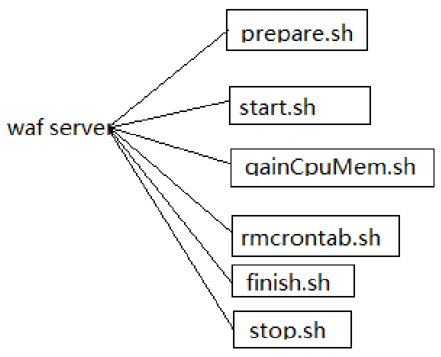
测试目的

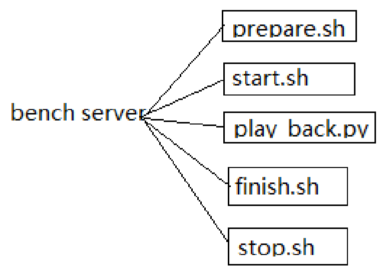
通过对单域名的压测，查看waf服务器的各个性能

测试框架

同上质量测试

脚本框架





工作内容

waf_server

1. 实现内容：接收请求的同时，记录自身机器的各个性能参数，包括nginx、shark、squid的CPU均值和MEM均值
2. 相关代码
    1. prepare.sh

```
1.查看waf版本是否正确，不正确则安装相应版本的rpm包
2.修改shark和squid的配置文件
3.关闭打包程序
```

```
#!/bin/bash

# waf_version host        concurrent_thread  repeat_time
data_version    guid        mode
# waf版本          压测的域名      并发数                   重复发送数据次数   测试数据
       唯一标识      模式


# 检测当前waf版本是否对应，安装相应的waf版本
current_version=`/usr/local/nginx/sbin/nginx -v 2>&1 |grep 'wswaf'|awk -
F"[ /]" '{print $4}'|awk -F ".el6|.standalone" '{print $1}'`
compare=$(echo $current_version ${1} | awk '$1>$2 {print 1} $1==$2
{print 0} $1<$2 {print 2}')
if test -z $current_version
then
    # 当前机器没有waf版本，安装对应的rpm包
    rpm -ivh /usr/local/waf_test/sbin/wswaf-${1}.el6.x86_64.rpm
elif [ $compare = 1 ]
then
    # 当前机器waf版本高于要安装的版本,需要回退
    rpm -Uvh /usr/local/waf_test/sbin/wswaf-${1}.el6.x86_64.rpm --
oldpackage
elif [ $compare = 2 ]
```

```
then
    # 当前机器waf版本低于要安装的版本
    rpm -Uvh /usr/local/waf_test/sbin/wswaf-${1}.el6.x86_64.rpm
else
    echo "correct waf version!"
fi
# 再次检测版本
current_version=`/usr/local/nginx/sbin/nginx -v 2>&1 |grep 'wswaf'|awk -
F"[ /]" '{print $4}'|awk -F ".el6|.standalone" '{print $1}'`
if [ $current_version != ${1} ]
then
    dateStamp=$(date '+%Y-%m-%d:%H:%M:%S')
    echo "{\"error_type\":\"wswaf-${1} install failed \",
\"time\":\"$dateStamp\"}" >>/dev/stderr
    exit 1
fi

# 更改squid配置文件
sed -i '/domain_replace_dst_ip/c\domain_replace_dst_ip 117.23.51.196'
/usr/local/squid/etc/channel/ws_${2}.conf
service squid reload
sleep 60

# 更改shark配置文件
# s是替换，-i表示在原文件上操作，^表示句首，&表示在匹配到的字符前加入&前的字符#，g表
示global全部替换，不加g表示只替换每行第一个
sed  -i 's/^[ ]*server 121.46.247.124:9101 max_fails=0;/#&/g'
/usr/local/shark/etc/shark.conf
service shark reload
sleep 60

# 关闭打包程序
sh /usr/local/waf_test/sbin/rmcrontab.sh

# 如果tmp下原有signal文件，则删除
if test -e /usr/local/waf_test/tmp/${6}
then
    rm -f /usr/local/waf_test/tmp/${6}
fi

# 清空access.log
echo -n > /usr/local/nginx/logs/access/access.log

echo "ready to analyze data!"
sleep 60
```

3. start.sh

```
    1.创建对应的文件目录
    2.调用gainCpuMem.sh脚本
```

```
   #!/bin/bash

   # waf_version host        concurrent_thread  repeat_time
data_version     guid       mode
   # waf版本          压测的域名      并发数                    重复发送数据次数   测试数据
        唯一标识      模式


   mkdir /usr/local/waf_test/analysis/${6}-${7}
   mkdir /usr/local/waf_test/analysis/${6}-${7}/nginx
   mkdir /usr/local/waf_test/analysis/${6}-${7}/shark
   mkdir /usr/local/waf_test/analysis/${6}-${7}/squid

   sh /usr/local/waf_test/sbin/gainCpuMem.sh ${1} ${2} ${3} ${4} ${5} ${6}
${7}
```

4. gainCpuMem.sh

```
   获取nginx、shark、squid的CPU和MEM均值
```

```
   #!/bin/bash

   # waf_version host        concurrent_thread  repeat_time
data_version    guid       mode
   # waf版本          压测的域名      并发数                    重复发送数据次数   测试数据
       唯一标识      模式


   avg_nginxC=0
   avg_nginxM=0
   avg_sharkC=0
   avg_sharkM=0
   avg_squidC=0
   avg_squidM=0
   nginx_arrayC=()
   nginx_arrayM=()
   shark_arrayC=()
   shark_arrayM=()
   squid_arrayC=()
   squid_arrayM=()

   while true
   do
    if test -e /usr/local/waf_test/tmp/${6}
    then
        break
    fi

    echo 'one time'
    total_nginxC=0
    total_nginxM=0
    PID_nginx_array=`ps -ef|grep nginx|grep worker|grep -v nobody|awk
'{print $2}'`
    nginx_len=0
    for PID in $PID_nginx_array
    do
```

```
        if test -n "$PID"
        then
            CPU=`top -b -p $PID -n 1|grep nginx|awk '{print strtonum($(NF-
4))}'`
            MEM=`top -b -p $PID -n 1|grep nginx|awk '{print strtonum($(NF-
3))}'`
            total_nginxC=$(echo "scale=2; $total_nginxC+$CPU" | bc)
            total_nginxM=$(echo "scale=2; $total_nginxM+$MEM" | bc)
            nginx_len=$[$nginx_len+1]
            echo $CPU >>
/usr/local/waf_test/analysis/${6}-${7}/nginx/nginx-$PID.CPU
            echo $MEM >>
/usr/local/waf_test/analysis/${6}-${7}/nginx/nginx-$PID.MEM
        fi
    done
    echo "nginx_len:$nginx_len"
    nginx_arrayC+=($(echo "scale=2; $total_nginxC/$nginx_len" | bc))
    nginx_arrayM+=($(echo "scale=2; $total_nginxM/$nginx_len" | bc))

    total_sharkC=0
    total_sharkM=0
    shark_len=0
    PID_shark_array=`ps -ef|grep shark|grep worker|awk '{print $2}'`
    for PID in $PID_shark_array
    do
        if test -n "$PID"
        then
            CPU=`top -b -p $PID -n 1|grep shark|awk '{print strtonum($(NF-
4))}'`
            MEM=`top -b -p $PID -n 1|grep shark|awk '{print strtonum($(NF-
3))}'`
            if test -z "$CPU" -o -z "$MEM"
            then
                continue
            fi
            total_sharkC=$(echo "scale=2; $total_sharkC+$CPU" | bc)
            total_sharkM=$(echo "scale=2; $total_sharkM+$MEM" | bc)
            shark_len=$[$shark_len+1]
            echo $CPU >>
/usr/local/waf_test/analysis/${6}-${7}/shark/shark-$PID.CPU
            echo $MEM >>
/usr/local/waf_test/analysis/${6}-${7}/shark/shark-$PID.MEM
        fi
    done
    echo "shark_len:$shark_len"
    shark_arrayC+=($(echo "scale=2; $total_sharkC/$shark_len" | bc))
    shark_arrayM+=($(echo "scale=2; $total_sharkM/$shark_len" | bc))


    total_squidC=0
    total_squidM=0
    squid_len=0
    PID_squid_array=`ps -ef|grep '(squid)'|grep -v root|awk '{print $2}'`
    for PID in $PID_squid_array
    do
        if test -n "$PID"
        then
```

```
            CPU=`top -b -p $PID -n 1|grep squid|awk '{print strtonum($(NF-
4))}'`
            MEM=`top -b -p $PID -n 1|grep squid|awk '{print strtonum($(NF-
3))}'`
            if test -z "$CPU" -o -z "$MEM"
            then
                continue
            fi
            total_squidC=$(echo "scale=2; $total_squidC+$CPU" | bc)
            total_squidM=$(echo "scale=2; $total_squidM+$MEM" | bc)
            squid_len=$[$squid_len+1]
            echo $CPU >>
/usr/local/waf_test/analysis/${6}-${7}/squid/squid-$PID.CPU
            echo $MEM >>
/usr/local/waf_test/analysis/${6}-${7}/squid/squid-$PID.MEM
        fi
    done
    echo "squid_len:$squid_len"
    squid_arrayC+=($(echo "scale=2; $total_squidC/$squid_len" | bc))
    squid_arrayM+=($(echo "scale=2; $total_squidM/$squid_len" | bc))

    sleep 5
   done

   len=${#nginx_arrayC[*]}
   for((i=1;i<$[$len-3];i++))
   do
    avg_nginxC=$(echo "scale=2; $avg_nginxC+${nginx_arrayC[i]}" | bc)
    avg_nginxM=$(echo "scale=2; $avg_nginxM+${nginx_arrayM[i]}" | bc)
    avg_sharkC=$(echo "scale=2; $avg_sharkC+${shark_arrayC[i]}" | bc)
    avg_sharkM=$(echo "scale=2; $avg_sharkM+${shark_arrayM[i]}" | bc)
    avg_squidC=$(echo "scale=2; $avg_squidC+${squid_arrayC[i]}" | bc)
    avg_squidM=$(echo "scale=2; $avg_squidM+${squid_arrayM[i]}" | bc)
   done

   avg_nginxC=$(echo "scale=2; $avg_nginxC/($len-4)" | bc)
   avg_nginxM=$(echo "scale=2; $avg_nginxM/($len-4)" | bc)
   avg_sharkC=$(echo "scale=2; $avg_sharkC/($len-4)" | bc)
   avg_sharkM=$(echo "scale=2; $avg_sharkM/($len-4)" | bc)
   avg_squidC=$(echo "scale=2; $avg_squidC/($len-4)" | bc)
   avg_squidM=$(echo "scale=2; $avg_squidM/($len-4)" | bc)

   echo "nginx: ${nginx_arrayC[*]}" >>
/usr/local/waf_test/analysis/${6}-${7}/avg
   echo "shark: ${shark_arrayC[*]}" >>
/usr/local/waf_test/analysis/${6}-${7}/avg
   echo "squid: ${squid_arrayC[*]}" >>
/usr/local/waf_test/analysis/${6}-${7}/avg

   echo "avg_nginxC=$avg_nginxC" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
   echo "avg_nginxM=$avg_nginxM" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
   echo "avg_sharkC=$avg_sharkC" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
   echo "avg_sharkM=$avg_sharkM" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
```

```
    echo "avg_squidC=$avg_squidC" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
    echo "avg_squidM=$avg_squidM" >>
/usr/local/waf_test/analysis/${6}-${7}/report.txt
```

5. rmcrontab.sh

> 关闭打包程序

```
    #! /bin/sh
    function remove_crontab()
    {
     CMD="$1"
     is_restart=0
     awk '$0 != MATCHSTR' MATCHSTR="$CMD" /var/spool/cron/root >
/var/spool/cron/root.bak
     if diff /var/spool/cron/root /var/spool/cron/root.bak 1>/dev/null 2>&1
     then
         is_restart=1
     fi
     mv -f /var/spool/cron/root.bak /var/spool/cron/root
     chmod 600 /var/spool/cron/root
     if [ $is_restart -eq 0 ]
     then
         echo "Remove crontab: $CMD"
         return 0
     else
         echo "crontab already removed: $CMD"
         return 1
     fi
    }

    line1="*/1 * * * * /bin/sh /usr/local/nginx/bin/package_access_log.sh
1>/dev/null 2>&1"
    remove_crontab "$line1"
    line2="*/1 * * * * /bin/sh /usr/local/nginx/bin/package_log.sh
1>/dev/null 2>&1"
    remove_crontab "$line2"
    line3="*/5 * * * * /usr/bin/python
/usr/local/nginx/bin/check_conf_to_reload.py 1>/dev/null 2>&1"
    remove_crontab "$line3"
```

6. finish.sh

> 测试完成后，打包测试结果，向测试中心发送信息

```
    #!/bin/bash

    # waf_version host         concurrent_thread   repeat_time
data_version     guid        mode
    # waf版本           压测的域名      并发数                    重复发送数据次数   测试数据
        唯一标识      模式
```

```
   touch /usr/local/waf_test/tmp/${6}
   sleep 10

   echo -n > /usr/local/waf_test/tmp/info.json
   tar zcPf /usr/local/waf_test/upload.tar
/usr/local/waf_test/analysis/${6}-${7}
   echo "{\"result\":\"success\",
\"file_path\":\"/usr/local/waf_test/upload.tar\"}" >
/usr/local/waf_test/tmp/info.json

   curl -X POST http://127.0.0.1:60001/finished_bench -H "content-
type:application/json" -T /usr/local/waf_test/tmp/info.json
   echo "finished"
```

7. stop.sh

```
   中止waf server的所有测试进程
```

```
   #!/bin/bash

   # waf_version host        concurrent_thread  repeat_time
data_version    guid       mode
   # waf版本          压测的域名      并发数                   重复发送数据次数   测试数据
       唯一标识      模式

   kill `ps -ef|grep ${6}|grep -v grep|awk '{print $2}'`
   echo "测试已中止"
```

bench_server

1. 实现内容：向单域名发送请求数据，计算qps
2. 相关脚本
   1. prepare.sh

```
探测waf server服务器是否成功响应
```

```
#!/bin/bash

# waf_version host        concurrent_thread  repeat_time        测试数据
          guid          mode
# waf版本         压测的域名      并发数                   重复发送数据次数
data_version        唯一标识       模式

status=`curl http://115.54.16.68/testcdn.htm -H "Host:botsec.haplat.net"
-i|sed -n 1p|awk '{print $2}'`
if [ $status = '200' ]
then
    echo 'connect successfully'
else
    dateStamp=$(date '+%Y-%m-%d:%H:%M:%S')
    echo "{\"error_type\":\"connect wafserver failed \",
\"time\":\"$dateStamp\"}" >>/dev/stderr
```

```
    exit 1
fi

echo 'ready to send data!'
```

2. start.sh

```
1.创建响应的文件目录
2.调用play_back.py
3.调用finish.sh
```

```
#!/bin/bash

# waf_version host        concurrent_thread  repeat_time        测试数据
          guid        mode
# waf版本        压测的域名     并发数                  重复发送数据次数
data_version        唯一标识    模式

mkdir /usr/local/waf_test/analysis/${6}-${7}

python /usr/local/waf_test/sbin/play_back.py ${1} ${2} ${3} ${4} ${5}
${6} ${7}

sh /usr/local/waf_test/sbin/finish.sh ${1} ${2} ${3} ${4} ${5} ${6} ${7}
```

3. play_back.py

```
多线程向waf server发送数据，统计qps
```

```
#!/usr/bin/python
#coding=utf-8

from __future__ import division
import logging
import
os,sys,time,socket,threading,copy,StringIO,cPickle,subprocess,fcntl,stru
ct,signal,shutil,commands,logging,datetime,json,platform,re
import re,ConfigParser
import time
#import redis
import Queue
import thread


# 传参：waf版本 域名 并发数 重复次数 测试数据 唯一id

work_dir = "/usr/local/waf_test/"
nginx_work_dir = "/usr/local/nginx/conf/servers/"
logging.basicConfig(filename = work_dir + "logs/waf_test.log",level =
logging.INFO, filemode = 'a', format = '%(asctime)s - %(levelname)s: %
(message)s')
data_dir = work_dir + "datas/"
worker_num = int(sys.argv[3])
exit_worker = 0
```

```
status_list={}
total_num=0
ok_num1=0
err_num=0
use_time=0
q = Queue.Queue(maxsize = 0)
lock=threading.Lock()
host = sys.argv[2]
repeat_time = int(sys.argv[4])
guid = sys.argv[6]
mode = sys.argv[7]
count=0
qps=[]
requests_info = {}
data_len = 0
for i in range(0, worker_num):
    requests_info[i] = Queue.Queue(maxsize = 0)

def main():
    global use_time
    global start
    #file_path=sys.argv[1]
    ip='115.54.16.68'
    init_data_list() # 初始化数据
    #sock=connect_web(ip)
    start_time=time.time()
    start = time.time()
    init_work_list(ip)
    #do_replay(sock,file_path,ip)
    while 1:
        if exit_worker == worker_num:
            break
        time.sleep(1)
    end_time=time.time()
    use_time= end_time - start_time
    play_summary(use_time)


def init_data_list():
    all_request = []
    try:
        dir_list = os.listdir(data_dir)
        if not dir_list:
            return
        else:
            for file_name in dir_list:
                split_num = 0;
                request_data=""
                #print file_path
                file_path = data_dir + file_name
                with open(file_path,"r") as f:
                    for line in f.readlines():
                        if line=="====****\n":
                            #print re.sub("Host:.*\r\n","Host:
www.linsd.com\r\n",request_data)
                            #send_http_request(request_data,ip)
                            #for i in conf_list:
                            i = "botsec.haplat.net"
```

```
                              url = "http://" + i + "/"
                              request_data =
re.sub("http[s]?://(.+?)/",url,request_data)
                              #request_data =
re.sub("http://(.+?)/",url,request_data)
                              host_name = "Host: "+ i + "\r\n"
                              #print host_name
                              request_data =
re.sub("Host:.*\r\n",host_name,request_data)
                              #print request_data
                #print request_data
                              all_request.append(request_data)
                              #request_data = request_data + "###@@@"
                              #total_num=total_num+1
                              #split_num = split_num + 1
                              #if split_num == 100:
                              #   thread.start_new_thread(replay_work,
(ip,request_data))
                              #   split_num = 0
                              request_data = ""
                          else:
                              request_data = request_data + line
    except Exception, e:
        logging.error(e)

    for i in range(0, repeat_time):
        request_num = len(all_request)
        for j in range(request_num):
            for k in range(0, worker_num):
                requests_info[k].put(all_request[j])

    time.sleep(1)


def init_work_list(ip):
    for i in range(worker_num):
        thread.start_new_thread(replay_work, (ip,"", i))

def replay_work(ip,data, worker_id):
    global exit_worker
    global lock
    global ok_num1
    global qps

    while requests_info[worker_id].empty() == False:
        try:
            request_data = requests_info[worker_id].get()
            send_request(request_data,ip,worker_id)
        except Exception, e:
            logging.error(e)
    if requests_info[worker_id].empty():
        print "no requests"
    if lock.acquire():
        exit_worker=exit_worker+1
        lock.release()


def fPopen(aCmd):
    p=subprocess.Popen(aCmd, shell=True,
bufsize=4096,stdin=subprocess.PIPE, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, close_fds=True)
    sOut = p.stdout.read()
    sErr = p.stderr.read()
    return (sOut,sErr)

def send_request(data,ip,worker_id):
    global ok_num1
    global err_num
    global status_list
    global lock
    global end
    global start
    global data_len
    try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((ip, 80))
            http_request = data
            sock.send(http_request)
            rev_data = sock.recv(1024)
            status = rev_data[9:12]
            #print status
            if lock.acquire():
                ok_num1 += 1
        data_len += len(data)
        #print ok_num1
                if ok_num1 == 100000:
                    end = time.time()
                    Qps = 100000/(end - start)
                    s = 'data_len:' + str(data_len) + 'start:' +
str(start) + ' end:' + str(end) + ' qps:' + str(Qps)
            print s
                    qps.append(s)
                    ok_num1 = 0
                    start = time.time()
            data_len = 0
                lock.release()
            sock.close()
    except socket.error, msg:
            err_num=err_num+1
            logging.error(msg)
    except Exception, e:
            err_num=err_num+1
            logging.error(e)


def play_summary(use_time):
    with open(work_dir + 'analysis/' + guid + '-' + mode +
'/report.txt', 'w') as f:
        for element in qps:
            f.write(str(element) + '\n')
    # print "total_num: " + str(total_num)
    print "ok_num1: " + str(ok_num1)
    print "err_num: " + str(err_num)
    print "use_time: " + str(use_time)

if __name__ == "__main__":
    main()
    sys.exit(0)
```

4. finish.sh

```
完成发送数据，向测试中心发送消息
```

```
#!/bin/bash

# waf_version host        concurrent_thread  repeat_time        测试数据
          guid        mode
# waf版本       压测的域名     并发数                重复发送数据次数
data_version       唯一标识    模式


curl -X POST http://localhost:60002/finished_bench -H "content-
type:application/json" -d "{\"result\": \"success\"}"
echo "finished"
```

5. stop.sh

```
中止bench server上的所有测试进程
```

```
#!/bin/bash

# waf_version host        concurrent_thread  repeat_time        测试数据
          guid          mode
# waf版本       压测的域名     并发数                重复发送数据次数
data_version       唯一标识       模式


kill `ps -ef|grep ${6}|grep -v grep|awk '{print $2}'`
echo '测试已中止'
```