1. 监控web服务是否正常，不低于3种监控策略。要求间隔1分钟，持续监控。

############监控web服务的脚本编写##############

$ vi web-monitor.sh

#!/bin/bash

url=www.chengbi.cn

Port=80

check\_web(){

[ -f /etc/init.d/functions ]&& . /etc/init.d/functions

if [ "`nc -w 5 $url $Port &&echo ok`" = "ok" ];then

#if [ "`echo -e '\n'|telnet $url $Port|grep Connected|wc -l`" = "1" ];then

#if [ "`nmap $url -p $Port|grep open|wc -l`" = "1" ];then

#if [ "`curl -I http://$url 2>/dev/null|head -1|egrep "200|302|301"|wc -l`" ="1" ];then

#if [ "`curl -I -s -o /dev/null -w '%{http\_code}\n' http://$url`" = "200" ];then

action "$url $Port" /bin/true

else

action "$url $Port /bin/false"

fi

}

main(){

while true

do

check\_web

sleep 1m

done

}

1. 监控db服务是否正常，不低于3种监控策略。要求间隔1分钟，持续监控。

$ vi db-monitor.sh

#!bin/bash

Port=3306

check\_db(){

[ -f /etc/init.d/functions ] && . /etc/init.d/functions

#if [ `lsof -i:$Port|wc -l` -gt 1 ];then

#if [ "`netstat -tunlp|grep 3306|wc -l`" = "1" ];then

if [ `mysql -umha -pmha -h 172.16.2.10 -P $Port -e "show databases;"|wc -l` -gt 1 ];then

action "MySQL $Port online" /bin/true

else

action "MySQL $Port down" /bin/false

fi

}

main(){

while true

do

check\_db

sleep 1m

done

}

main

1. 监控web站点目录（/var/html/www）下所有文件是否被恶意篡改（文件内容被改了），如果有就打印改动的文件名（发邮件），定时任务每3分钟执行一次(10分钟时间完成)。

############监控文件被恶意篡改的脚本编写##############

$ vi tampering-file.sh

#!/bin/bash

check\_www(){

md5sum -c /tmp/md5\_www.log > /tmp/result\_www.log

[ ! -f /tmp/result\_www.log ]&& echo "/tmp/result\_www.log not exist."exit 2

exec < /tmp/result\_www.log

while read line

do

file = `echo $line|awk -F ' ' 'printf $1'`

#echo $file

#echo $result

[ ! "$result" = "OK" ]&&action "$file" /bin/false

done

}

main(){

while true

do

LANF=en

[ -f /etc/init.d/functions ]&& . /etc/init.d/functions

[ ! -f /tmp/md5\_www.log ]&& echo "/tmp/md5\_www.log not exist."&&exit 1

check\_www

action "Alii check done." /bin/true

sleep 3m

done

}

main

# 4.[nginx实现日志按天切割](https://www.cnblogs.com/littleatp/p/4625010.html)

### **背景**

    nginx的日志文件没有rotate功能。一段时间过后，日志将越发臃肿，一个accesslog很快就突破1G，因此有必要通过脚本实现按天切割日志。

### **解决思路**

1  重命名日志文件，如更改为access\_yyyyMMdd.log，需注意的是nginx通过文件描述符定位日志文件，因此在重命名之后还是能往该文件内写入内容。

2  向nginx主进程发送USR1信号。

    nginx的master进程接到信号后：

    重新从配置文件中读取日志文件名 -> 关闭重名日志文件 -> 创建并打开日志文件(原来的名称) -> 通过worker进程作出改变

### **代码实现**

A. nginx日志按日期自动切割脚本

#!/bin/bash

#日志文件存放目录

logs\_path="/usr/local/nginx/logs/"

# pid文件

pid\_path="/usr/local/nginx/nginx.pid"

#重命名日志文件mv ${logs\_path}access.log ${logs\_path}access\_$(date -d "yesterday" +"%Y%m%d").log

#向nginx主进程发送信号以重新打开日志kill -USR1 `cat ${pid\_path}

保存为 cut-log.sh,

B. 设置定时任务 "crontab -e"

0 0 \* \* \* bash /usr/local/nginx/nginx\_log.sh

将于每天凌晨0点0分将nginx日志重命名为昨天的日期格式，并重新生成今天的新日志

5：

（生产实战案例）：监控MySQL主从同步是否异常，如果异常，则发送短信或者邮件给管理员。提示：如果没主从同步环境,可以用下面文本放到文件里读取来模拟：  
阶段1：开发一个守护进程脚本每30秒实现检测一次。  
阶段2：如果同步出现如下错误号（1158,1159,1008,1007,1062），则跳过错误。

阶段3：请使用数组技术实现上述脚本（获取主从判断及错误号部分）

重点语句分析  
mysql查看状态语句

mysql>show slave status\G  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

               Slave\_IO\_State: Waiting for master to send event  
                  Master\_Host: 10.0.0.42  
                  Master\_User: backup  
                  Master\_Port: 3306  
                Connect\_Retry: 60  
              Master\_Log\_File: mysql-bin.000001  
          Read\_Master\_Log\_Pos: 538  
               Relay\_Log\_File: mysql-relay-bin.000002  
                Relay\_Log\_Pos: 701  
        Relay\_Master\_Log\_File: mysql-bin.000001  
             Slave\_IO\_Running: Yes  
            Slave\_SQL\_Running: Yes

使用MysqlCmd2 命令筛选后语句：

Yes  
Yes  
0  
0  
it

#!/bin/bash

port=3306

error=(1158 1159 1008 1007 1062)

MysqlCmd1= mysql -uroot -p192.168.51.9 -e

MysqlCmd2= mysql -uroot -p192.168.51.9 -e"show slave status\G"|egrep "\_Running|Last\_Errno|Behind\_Master"|awk '{print $NF}'

function is\_run(){

[ `lsof -i:$port|wc -l` -lt 2 ]&&{

echo"mysql is stop"

exit 1

}

}

function mysql\_status(){

array=(`$MysqlCmd2`)

}

function judge\_error(){for i in ${error[\*]}do

if [ "${array[2]}" == "$i" ];then

$MysqlCmd1"stop slave;SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER = 1;start slave;"

else

echo "mysql is failed,error id is ${array[2]}"

fidone

}

judge\_status(){

mysql\_statusecho ${array[\*]}if [ "${array[0]}" == "Yes" -a "${array[1]}" == "Yes" -a "${array[3]}" == "0" ];then

echo "Mysql slave is ok"else

judge\_error ${array[2]}fi

}

function main(){while true

do

is\_run

judge\_status

sleep 30

done

}

main

6：  
请用至少两种方法实现！  
写一个脚本解决DOS攻击生产案例  
提示：根据web日志或者或者网络连接数，监控当某个IP并发连接数或者短时内PV达到100，即调用防火墙命令封掉对应的IP，监控频率每隔3分钟。  
防火墙命令为：iptables -I INPUT -s 10.0.1.10 -j DROP。

方法1：  
设定触发条件 为并发连接数 达到100 即封掉ip  
  
重点语句分析  
先输出所有建立连接的ip地址，从大到小进行排序  
netstat -an|grep ESTABLISHED|awk -F '[: ]+' '{print $6}' |sort|uniq -c|sort -nr

如果netstat输出带有ipv6的   ::ffff:  字符串，用这个改进过的语句进行过滤  
netstat -an|grep ESTABLISHED|awk '{print $5}'|sed 's/::ffff://g'|awk -F: '{print $1}'|sort|uniq -c|sort -nr  
输出结果

210 10.11.0.78  
100 61.158.175.44  
100 42.234.11.172  
91 221.13.156.210  
81 10.11.0.79  
21 10.0.7.99

 顺序执行

#!/bin/bashwhile truedo

netstat -an|grep ESTABLISHED|awk -F '[: ]+' '{print $6}'|sort|uniq -c|sort|while read line

do

ip=`echo $line|awk '{print $2}'`

count=`echo $line|awk '{print $1}'`

if [ "$count -ge 100 ] && [ `iptables -L -n|grep "$ip"|wc -l` lt 1 ];then

iptables -I INPUT -s "$ip" -j DROP

echo $ip is DROP >> drop\_list.log

# fi

donesleep 60done

 设计为函数

#!/bin/bash

log=/tmp/ipdos.log

[ -f $log ] || touch $log

add\_iptables(){while read linedo

ip=`echo $line|awk '{print $2}'`

count=`echo $line|awk '{print $1}'`

if [ "$count" -ge 100 ] && [ `iptables -L -n|grep "$ip"|wc -l` lt 1 ];then

iptables -I INPUT -s "$ip" -j DROP

echo $ip is DROP

fidone < $log

}

main(){while truedo

netstat -an|grep ESTABLISHED|awk -F '[: ]+' '{print $6}'|sort|uniq -c|sort > $log

add\_iptables

sleep 60done

}

main

 方法2：  
设定触发条件 为过去三分钟内 访问日志中 ip达到100条记录（即pv）  即封掉ip

 重点语句分析  
web日志的格式

192.168.51.254 - - [22/Nov/2017:09:31:09 +0800] "GET /index.html HTTP/1.0" 200 612 "-" "ApacheBench/2.3"  
192.168.51.254 - - [22/Nov/2017:09:31:09 +0800] "GET /index.html HTTP/1.0" 200 612 "-" "ApacheBench/2.3"  
192.168.51.254 - - [22/Nov/2017:09:31:09 +0800] "GET /index.html HTTP/1.0" 200 612 "-" "ApacheBench/2.3"

先来处理时间，让输出时间与日志中时间格式一致  
#date "+%d/%b/%Y:%H:%M"  
22/Nov/2017:09:31

 当前时间的前一分钟和前两分钟表达方式

time1=`date "+%d/%b/%Y:%H:%M" -d "-1 minute"`  
time2=`date "+%d/%b/%Y:%H:%M" -d "-2 minute"`

#!/bin/bash

log=/tmp/dosip.log

access\_log="/usr/local/nginx/logs/access.log"

[ -f $log ] || touch $log

time=`date "+%d/%b/%Y:%H:%M"`

time1=`date "+%d/%b/%Y:%H:%M" -d "-1 minute"`

time2=`date "+%d/%b/%Y:%H:%M" -d "-2 minute"`

add\_iptables(){while read linedo

ip=`echo $line|awk '{print $2}'`

count=`echo $line|awk '{print $1}'`

if [ "$count" -ge 100 ] && [ `iptables -L -n|grep "$ip"|wc -l` -lt 1];then

iptable -I INPUT -s "$ip" -j DROP

echo "$ip" is DROP

fidone < $log

}

main(){while truedocat $access\_log|egrep "$time|$time1|$time2"|awk '{print $1}'|sort|uniq -c > $log

add\_iptablessleep 10done

}

main

7.如何实现对MySQL数据库进行分库备份，请用脚本实现

重点语句分析  
备份语句 ：

mysqldump -u$MYUSER -p$MYPASS -S$SOCKET -x -B -F -R databaseName  
-x 提交请求锁定所有数据库中的所有表，以保证数据的一致性。这是一个全局读锁，  
 并且自动关闭--single-transaction 和--lock-tables 选项。  
-B 备份指定的库  
-F 开始导出之前刷新日志。  
 请注意：假如一次导出多个数据库(使用选项--databases或者--all-databases)，将会逐个数据库刷新日志。  
 除使用--lock-all-tables或者--master-data外。在这种情况下，日志将会被刷新一次，相应的所以表同时被锁定。  
 因此，如果打算同时导出和刷新日志应该使用--lock-all-tables 或者--master-data 和--flush-logs。

-R 导出存储过程以及自定义函数

提取数据库名称到数组：

dataArray=(`mysql -uroot -pxxx -e"show databases;" |egrep -v "Database|schema"`)

#!/bin/bash

user=rootpwd=192.168.51.9

socket="/var/lib/mysql/mysql.sock"

mysqldump="mysqldump -u$user -p$pwd -S$socket -x -B -F -R"

backup=/var/sqlbackup

[ -f $backup ] || mkdir -p $backup

dataArray=(`mysql -u$user -p$pwd -e"show databases;" |egrep -v "Database|schema"`)

for i in ${dataArray[\*]}do

$mysqldump $i | gzip > ${backup}/"$i"\_$(date +"%F").sql.gz done

8:

如何实现对MySQL数据库进行分库加分表备份，请用脚本实现

重点语句分析  
备份语句 ：

mysqldump -u$MYUSER -p$MYPASS -S$SOCKET -x  -F -R databaseName tableName

用两个for循环来备份每一张表

这里将各数据库名称存到数组里

将各个表名存在字符串中

#!/bin/bash

user=rootpwd=192.168.51.9

socket="/var/lib/mysql/mysql.sock"

mysqldump="mysqldump -u$user -p$pwd -S$socket -x -F -R"

backup=/var/sqlbackup2

[ -f $backup ] || mkdir -p $backup

dataArray=(`mysql -u$user -p$pwd -e"show databases;" |egrep -v "Database|schema"`)

for i in ${dataArray[\*]}do

Table=`mysql -u$user -p$pwd -e"show tables from $i" | egrep -v "Tables\_in\_"`

for t in $Table

do

$mysqldump $i $t | gzip > ${backup}/"$i"\_"$t"\_`date +"%F"`.sql.gz

done

done

9：

监控memcache服务是否正常，模拟用户（web客户端）检测。

使用nc命令加上set/get来模拟检测，以及监控响应时间及命中率。

重点语句分析：

使用nc  
设置值  printf "set test 0 0 4\r\ngood\r\n"|nc 10.0.0.45 12000  
取值      printf "get test\n"|nc 10.0.0.45 12000

监控响应时间   
time echo stats |nc 10.0.0.45 12000 &>/dev/null

监控命中率a    
hit=`echo stats |nc 10.0.0.45 12000|grep get\_hits|awk '{print $3}'`  
miss=`echo stats |nc 10.0.0.45 12000|grep get\_misses|awk '{print $3}'`  
let a=$hit/($hit+$miss)

#!/bin/bash

printf "set test 0 0 4\r\ngood\r\n"|nc 10.0.0.45 12000

a=`printf "get test\n"|nc 10.0.0.45 12000|awk 'NR==2 {print $0}'`

[ $a == "good"] || echo "memcache is down" && time echo stats |nc 10.0.0.45 12000 &>/dev/null

hit=`echo stats |nc 10.0.0.45 12000|grep get\_hits|awk '{print $3}'`

miss=`echo stats |nc 10.0.0.45 12000|grep get\_misses|awk '{print $3}'`

let b=$hit/'($hit+$miss)'

echo "the memcache hit is $b"