**0x01：背景**

  直奔主题，先来看看网络层是如何进行入侵检测的：

  入侵检测重点关注的，是GetShell这个动作，以及GetShell成功之后的恶意行为（为了扩大战果，黑客多半会利用Shell进行探测、翻找窃取、横向移动攻击其它内部目标）。包括自己以往的真实的工作中，更多的是分析了GetShell之前的一些“外部扫描、攻击尝试”行为，基本上是没有意义的。外部的扫描和尝试攻击无时无刻不在持续发生的，而类似于SQL注入、XSS等一些不直接GetSHell的Web攻击，暂时不在狭义的“入侵检测”考虑范围，当然，利用SQL注入、XSS等入口，进行了GetShell操作的，我们仍抓GetShell这个关键点，就如sql入住进行GETshell，常见的使用 into outfile写函数，那么最简单的就是我们把流量镜像一份，孵化成日志，从uri/post/cookie等可能出现注入的地方检测是否是否了into outfile，和常用webshell形式以及状态码是否是200

常见的可以getshell的web攻击

  直接上传获取webshell、SQL注入、远程文件包含(RFI)、FTP，甚至使用跨站点脚本(XSS)作为攻击的一部分，甚至一些比较老旧的方法利用后台数据库备份及恢复获取webshell、数据库压缩等。通用功能包括但不限于shell命令执行、代码执行、数据库枚举和文件管理。

**0x02：webshell的检测**

  可以通过监控指定目录下的所有文件的创建、修改、重命名等操作，再比如说    <?php eval($\_POST[1])?>，当代码中出现eval与$\_POST时，判断为webshell，但如果只出现eval的函数，就判断为敏感函数。也可以通过webshell hash、文件名等进行检测，在GitHub上就有这么一个项目：[webshell\_特征](https://github.com/Neo23x0/signature-base/blob/e264d66a8ea3be93db8482ab3d639a2ed3e9c949/yara/thor-webshells.yar)、某大佬之前写过通过机器学习的方法去检测webshell： [初探机器学习检测 PHP Webshell](https://paper.seebug.org/526/)、webshell入侵检测总结：[webshell入侵检测](https://www.freebuf.com/articles/web/183520.html)

  其实可以看到，从网络层检测，基于关键字等检测方式，有被绕过的风险，上面还介绍了基于目录下文件的监控到达检测风险的作用，就由此引出了HIDS（主机型入侵检测系统）

**0x03：简介**

HIDS(Host-based Intrusion Detection System)作为传统攻防视角的重要一环，有着不可替代的作用，可以有效的检测到从网络层面难以发现的安全问题，如：后门，反弹shell，恶意操作，主机组件安全漏洞，系统用户管理安全问题，主机基线安全风险等。

**需求分析**

下面引用“[点融”](https://mp.weixin.qq.com/s?__biz=MzI5MjE4MDc4OQ==&mid=2247483961&idx=1&sn=2736aad509f08c20d82cfc08b62da27a&chksm=ec040463db738d754cce84506c098caca891b58740e1b38b3621f752f805eb1c02ebf0e2ac73&mpshare=1&scene=1&srcid=1226ZgmHAUTfeYMuLFPWyuHS#rd)和“ [美团](https://mp.weixin.qq.com/s?__biz=MjM5NjQ5MTI5OA==&mid=2651750220&idx=2&sn=26e1ae8056e4fd7db5e953e946a00b78&chksm=bd12a6018a652f17643ccf86264ea226a5d881c4dd2911772893e0c5d848f95f1f8de74b786d&mpshare=1&scene=1&srcid=0117j91c1pFiorQDDMN0XQka#rd)”对HIDS的需求：

1.灵活且轻量级：对系统资源需求少，系统负载占用小；

2. 有丰富的API和强大的联动能力：兼容性好，可横向扩展，支持和CMDB联动，支持Docker，可以梳理建立“白名单”(如某应用的对外连接名单，数据库名单等)。；

3.支持基线检查/系统完整性检测；

4. 支持规则引擎，能够灵活的进行规则配置(规则引擎最好可以和AgentSmith-NIDS复用)；

**架构**

目前大部分的HIDS平台建设主要分三大部分：终端Agent监控组件，Dashboard控制面板和与SIEM、运维数据等其他平台对接的接口集合

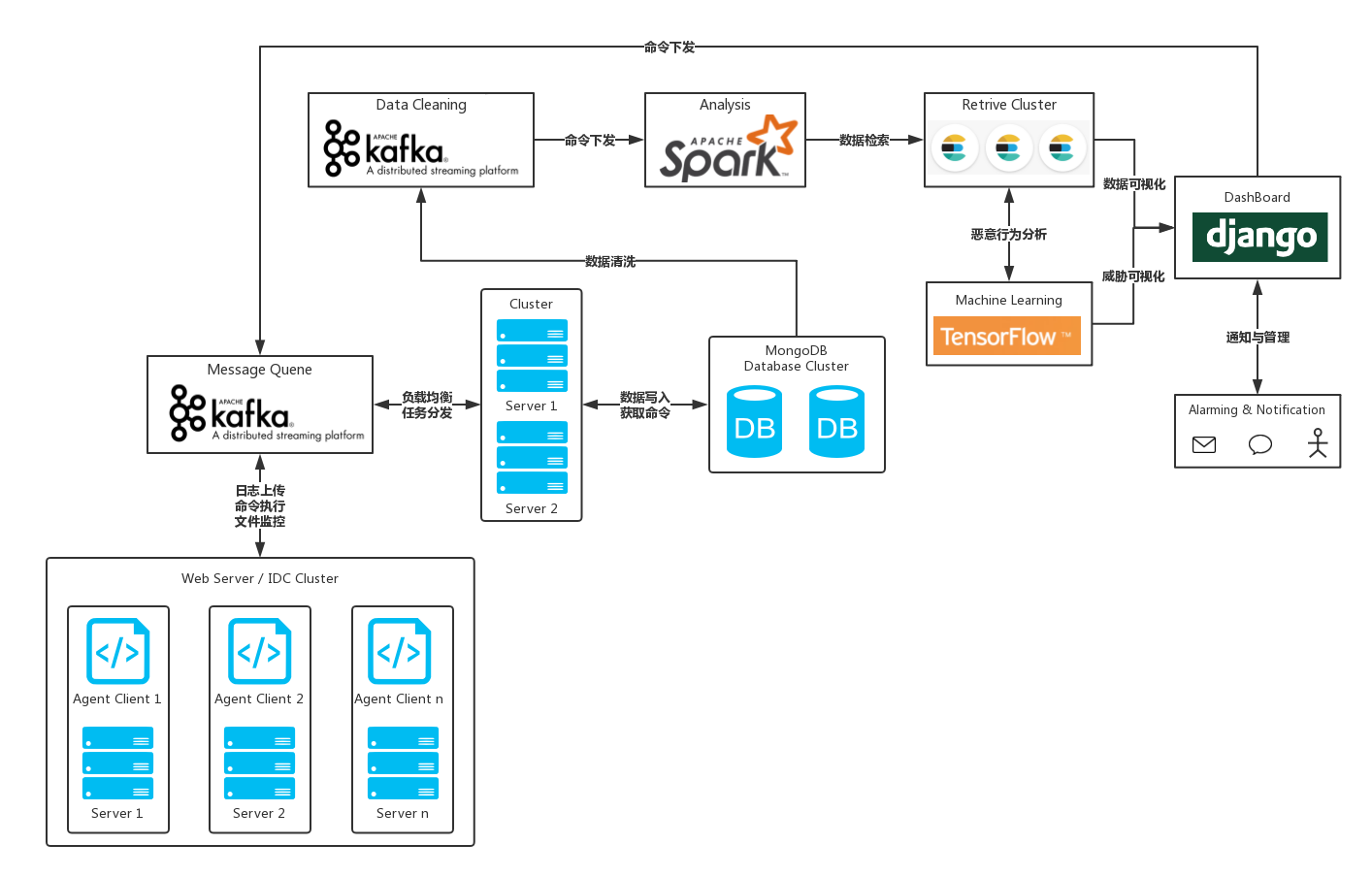
终端Agent组件：主要作用包括：监控文件变更、监控服务器状态、下发一些操作指令等（如果控制了一台HIDS服务器相当于拥有了百万台肉鸡，HIDS的研发大佬，只能帮到这了）。

DashBoard：用来执行一些策略推送、资源管理方面的操作

MQ && Servers：用来做负载均衡并吞吐数据到数据库

Database：数据库

SIEM APIs：用来将HIDS的数据和SIEM做整合



终端Agent通过对业务IT资产文件的监控可以发现一些潜在威胁的文件或者是被人中的webshell后门，也可以记录和发现文件改动。同时终端Agent肩负着把日志摆渡到数据库的工作，方便运维人员对日志进行检索，进行终端日志的统一化收集管理。这里直接使用了公司先已有的安全架构，直接将日志发送到kibana统一分析

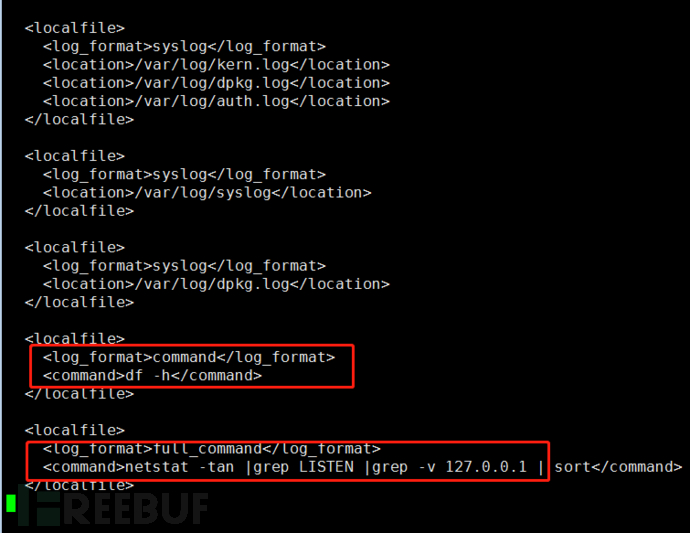
**0x04 开源HIDS**

 知名的HIDS项目如OSSEC、Osquery。OSSEC的主要功能包括日志监控、文件完整性检测、Rootkit检测以及联动配置，另外你也可以将自己的其他监控项集成到OSSEC中。可以参照OSSEC官网，如果你可以一样，不想参考英文文档~[OSSEC\_安装](https://www.cnblogs.com/zlslch/p/8512757.html)、[OSSEC\_扩展使用](https://xz.aliyun.com/t/2003)

**日志监控**

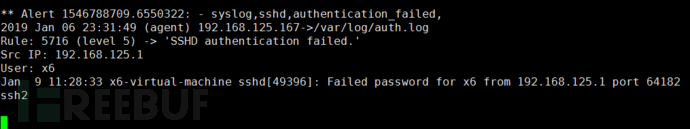
  日志是平常安全运维中很重要的一项，OSSEC日志检测为实时检测，OSSEC的客户端本身没有解码文件和规则，所监控的日志会通过1514端口发送到服务端。

  在agent端ossec.conf/agent.conf配置，需要注意的是command和full\_command不能配置在agent.conf中，需要配置在ossec.conf中

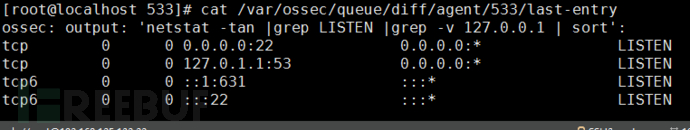


图：客户端监控

  一个以时间日志文件，ossec/logs/alerts/2019/Jan/ossec-alerts-06.log（本次06为agent端的主机名）

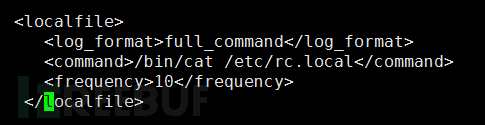


图：auth认证日志

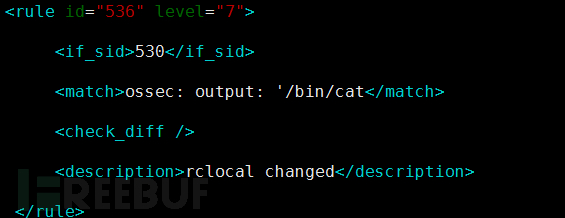


图：端口监听日志

  再已启动项为例，Linux下开机启动项是应急响应中很重要的检测项，Redhat中的运行模式2、3、5都把/etc/rc.d/rc.local做为初始化脚本中的最后一个。这里我在agent的ossec.conf中新加一个监控，检测当rc.local发生改变的时候告警。

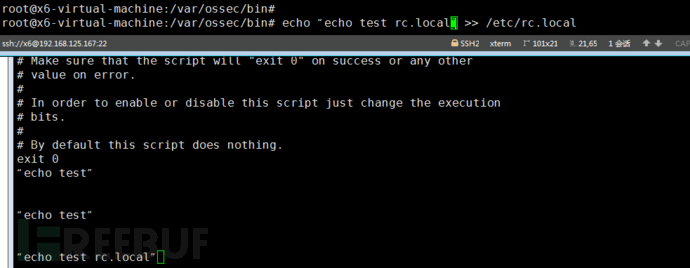


图：客户端监控规则



图：服务器监控规则

  会在queue/diff/agent生成和规则id相同的文件夹，报警如下

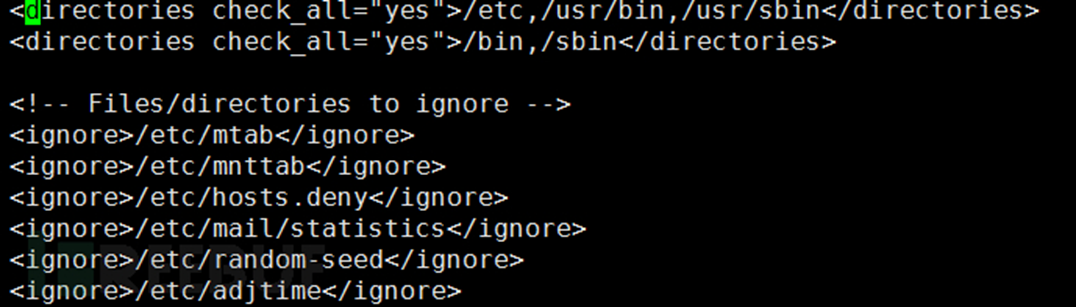


图：监控结果

这里包括做了一些端口反弹如bash、nc、python就不一一贴图了

**完整性检测**

  有多种类型的攻击和许多攻击向量，但所有这些都有一个独特之处：它们留下痕迹并始终以某种方式改变系统。从修改一些文件的病毒到改变内核的内核级rootkit，系统的完整性总会有一些变化。完整性检查是入侵检测的重要组成部分，可检测系统完整性的变化，通过可以通过md5检验来实现，而命令替换在应急响应中很常见，经常被替换掉的命令例如ps、netstat、ss、lsof等等。另外还有SSH后门。完整性检测的工作方式是Agent周期性的扫描系统文件，并将检验和发送给Server端。Server端存储并进行比对，发现修改是发出告警。



图：客户端监控配置

本文使用的属性值为check\_all，即使用全部为yes，会根据周期性进行检测

checkall：以下的属性全部为yes

realtime：实时监控

report\_changes：报告文件变化，文件类型只能是文本

check\_sum：监测MD5和SHA1 HASH的变化，相当于设置check\_sha1sum=”yes”和check\_md5sum=”yes”

check\_sha1sum：监测SHA1 HASH的变化

check\_md5sum：监测MD5 HASH的变化

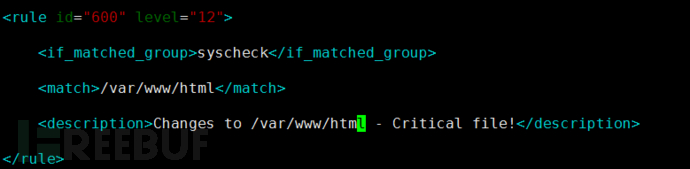
check\_size：监测文件大小

check\_owner：监测属主

check\_group：监测属组

check\_perm：监测文件权限

restrict：限制对包含该字符串的文件监测

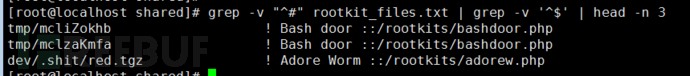


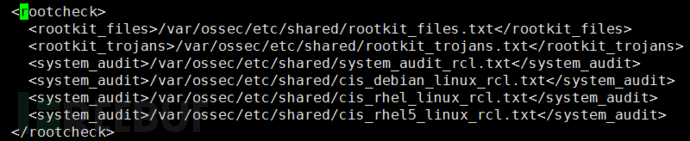
图：服务端监控规则

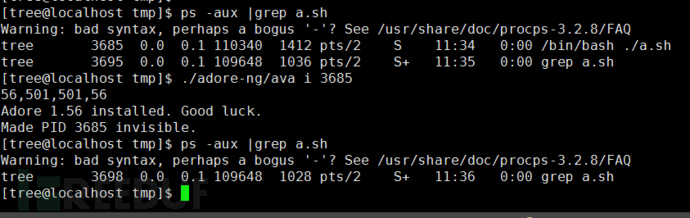
  服务端数据存放到服务端的/var/ossec/queue/syscheck目录下，这里健康的是常被攻击的web路径，如当修改/var/www/html时，告警级别为12

**rootkit检测**

  ossec检测rootkit的原理是对比/var/ossec/etc/shared/rootkit\_fikes.txt文件，该文件包含了rootkit常用的文件，和病毒库一样。



  会扫描整个文件系统，检测异常文件和异常的权限设置，文件属主是root，但是其他用户可写是非常危险的，rootkit将会扫描这些文件。同时还会检测具有suid权限的文件、隐藏的文件和目录。另外还会检测隐藏端口、隐藏进程、/dev目录、网卡混杂模式等。   
这里找了个经典的rootkit测试，[adore-ng](https://github.com/trimpsyw/adore-ng)



**0x05：个人HIDS实践**

HIDS，从开源项目从回走。OSSEC的主要功能包括日志分析、文件完整性检测、Rootkit检测以及联动配置，先从这几个入手。

**个人需求**

基础信息：

系统版本信息 帐号信息 用户组信息 系统应用信息 系统ARP信息 CPU信息 系统环境变量 文件监控信息 硬盘信息 内核信息 内存信息 本地防火墙信息 网络连接 文件共享信息 端口监听信息 进程信息 本地路由表信息 系统服务 系统ARP信息 rpm包信息

检测信息：

用户登录信息（包括可登录shell，ssh公私钥，可以用户，root\_id，用户登录的源ip地址等） sudo授权信息 /etc/rc.local脚本内容 计划任务/etc/cron.d 用户定义的自启动项 系统内核模块运行时的文件信息（基于白名单的md5计算内核模块文件检测） 运行进程数（基于白名单的异常进程检测） 端口网络连接信息（基于白名单的异常连接检测或基于威胁情报的异常网络连接检测） 本地防火墙信息检测 rpm包检测（diff对比以及rpm签名） 环境变量检测

日志信息： bash\_history（snoopy）命令记录 audit日志 auth认证日志 secure安全日志 服务日志（如中间件、数据库、FTP等）

kern内核日志 自定义local5日志

**基本信息获取**

基本信息获取的方法比较简单，本文使用python来实现，业务的现学现卖的变成，这里使用使用psutil库获取系统信息：[psutil](https://www.cnblogs.com/saneri/p/7528283.html)

#!/usr/bin/python

#coding:utf-8

import psutil

import datetime

import time

# 当前时间

now\_time = time.strftime('%Y-%m-%d-%H:%M:%S', time.localtime(time.time()))

print(now\_time)

# 查看cpu物理个数的信息

print(u"物理CPU个数: %s" % psutil.cpu\_count(logical=False))

#CPU的使用率

cpu = (str(psutil.cpu\_percent(1))) + '%'

print(u"cup使用率: %s" % cpu)

#查看内存信息,剩余内存.free 总共.total

#round()函数方法为返回浮点数x的四舍五入值。

free = str(round(psutil.virtual\_memory().free / (1024.0 \* 1024.0 \* 1024.0), 2))

total = str(round(psutil.virtual\_memory().total / (1024.0 \* 1024.0 \* 1024.0), 2))

memory = int(psutil.virtual\_memory().total - psutil.virtual\_memory().free) / float(psutil.virtual\_memory().total)

print(u"物理内存： %s G" % total)

print(u"剩余物理内存： %s G" % free)

print(u"物理内存使用率： %s %%" % int(memory \* 100))

# 系统启动时间

print(u"系统启动时间: %s" % datetime.datetime.fromtimestamp(psutil.boot\_time()).strftime("%Y-%m-%d %H:%M:%S"))

# 系统用户

users\_count = len(psutil.users())

#

# >>> for u in psutil.users():

# ... print(u)

# ...

# suser(name='root', terminal='pts/0', host='61.135.18.162', started=1505483904.0)

# suser(name='root', terminal='pts/5', host='61.135.18.162', started=1505469056.0)

# >>> u.name

# 'root'

# >>> u.terminal

# 'pts/5'

# >>> u.host

# '61.135.18.162'

# >>> u.started

# 1505469056.0

# >>>

users\_list = ",".join([u.name for u in psutil.users()])

print(u"当前有%s个用户，分别是 %s" % (users\_count, users\_list))

#网卡，可以得到网卡属性，连接数，当前流量等信息

net = psutil.net\_io\_counters()

bytes\_sent = '{0:.2f} Mb'.format(net.bytes\_recv / 1024 / 1024)

bytes\_rcvd = '{0:.2f} Mb'.format(net.bytes\_sent / 1024 / 1024)

print(u"网卡接收流量 %s 网卡发送流量 %s" % (bytes\_rcvd, bytes\_sent))

io = psutil.disk\_partitions()

# print(io)

# print("io[-1]为",io[-1])

#del io[-1]

print('-----------------------------磁盘信息---------------------------------------')

print("系统磁盘信息：" + str(io))

for i in io:

o = psutil.disk\_usage(i.device)

print("总容量：" + str(int(o.total / (1024.0 \* 1024.0 \* 1024.0))) + "G")

print("已用容量：" + str(int(o.used / (1024.0 \* 1024.0 \* 1024.0))) + "G")

print("可用容量：" + str(int(o.free / (1024.0 \* 1024.0 \* 1024.0))) + "G")

print('-----------------------------进程信息-------------------------------------')

# 查看系统全部进程

for pnum in psutil.pids():

p = psutil.Process(pnum)

print(u"进程名 %-20s 内存利用率 %-18s 进程状态 %-10s 创建时间 %-10s " \

% (p.name(), p.memory\_percent(), p.status(), p.create\_time()))

**文件监控**

inotify是一种文件系统变化通知机制，如增加文件、删除等事件可以立刻让用户得知，[inortify参考文档](https://www.ibm.com/developerworks/cn/linux/l-inotify/index.html?ca=drs-%20https://my.oschina.net/LinBigR/blog/833919)，这里监控的是常受攻击的web目录，监控了文件的创建、删除、修改属性，该机制是著名的桌面搜索引擎项目 beagle 引入的，并在 Gamin 等项目中被应用。

#!/usr/bin/python

# encoding:utf-8

import os

from pyinotify import WatchManager, Notifier,ProcessEvent,IN\_DELETE, IN\_CREATE,IN\_MODIFY

class EventHandler(ProcessEvent):

"""事件处理"""

def process\_IN\_CREATE(self, event):

print("Create file: %s " % os.path.join(event.path,event.name))

def process\_IN\_DELETE(self, event):

print("Delete file: %s " % os.path.join(event.path,event.name))

def process\_IN\_MODIFY(self, event):

print("Modify file: %s " % os.path.join(event.path,event.name))

def FSMonitor(path='/var/www/html'):

wm = WatchManager()

mask = IN\_DELETE | IN\_CREATE |IN\_MODIFY

notifier = Notifier(wm, EventHandler())

wm.add\_watch(path, mask,rec=True)

print('now starting monitor %s'%(path))

while True:

try:

notifier.process\_events()

if notifier.check\_events():

notifier.read\_events()

except KeyboardInterrupt:

notifier.stop()

break

if \_\_name\_\_ == "\_\_main\_\_":

FSMonitor()

**进程监控**

1.可使用inotify实时监控/proc下面的文件变动即可（a.有可能进程启动、结束太快来不及读取文件内容从而没捕捉到必要信息）

2.使用psutil获取进程信息，diff下即可，基本信息中已获取进程信息

3.直接调用os模块执行命名ps -ef（嗯，我知道上面的思路好low）

**文件完整性**

文件完整性，本文监控了命令替换，监控的目录为/usr/bin/，获取文件路径，计算md5直接diff。

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import os

import hashlib

import datetime

import difflib

global srt1

def gci(filepath):

#遍历filepath下所有文件，包括子目录

files = os.listdir(filepath)

#print(files)

for fi in files:

fi\_d = os.path.join(filepath,fi)

if os.path.isdir(fi\_d):#判断是否是一个目录，是则继续执行循环执行gci函数

gci(fi\_d)

else:

file=os.path.join(filepath,fi\_d)

#计算文件的md5值

hash = hashlib.md5()

f=open(file,'rb')

while True:

b=f.read(1000)#每次以特定字节读取，避免内容过大，占用内存

if not b:

break

hash.update(b)#更新哈希对象以字符串参数

f.close()

str=(file,hash.hexdigest())#返回摘要，作为十六进制数据字符串值，并定义为字典

str1=(str[0]+":"+str[1])

str2=open('/root/md5.txt')#md5的文件为md5文件样本

str3=str2.read()

d = difflib.HtmlDiff()

f=open('/root/md6.txt','w')

print (d.make\_file(str1,str3),file=f)#将文件名和md5值对应，并初始md5值对比发现问题邮件发送

f.close()

if \_\_name\_\_=="\_\_main\_\_":

gci('/usr/bin/')

# 输出文件的md5值以及记录运行时间

starttime = datetime.datetime.now()

endtime= datetime.datetime.now()

print('运行时间：%ds'%((endtime-starttime).seconds))

**rootkit检测**

  rootkit检测还没有好的思路，按照ossec是调用rootkit常用的文件，这里能想到的就是监控系统的文件变化，以及进程信息，若进程被rootkit隐藏，可以对比/proc，对比id查看是否被隐藏，若按照这种思路，以上的代码改改就能实现了。或许还可以直接调用chkrootkit、rhhunter进行检测。

参考文献：

[保障IDC安全：分布式HIDS集群架构设计](https://mp.weixin.qq.com/s?__biz=MjM5NjQ5MTI5OA==&mid=2651750220&idx=2&sn=26e1ae8056e4fd7db5e953e946a00b78&chksm=bd12a6018a652f17643ccf86264ea226a5d881c4dd2911772893e0c5d848f95f1f8de74b786d&mpshare=1&scene=1&srcid=0117j91c1pFiorQDDMN0XQka#rd)

[企业安全建设—基于Agent的HIDS系统设计的一点思路](https://xz.aliyun.com/t/1732)

[点融开源AgentSmith HIDS--- 一套轻量级的HIDS系统](https://mp.weixin.qq.com/s?__biz=MzI5MjE4MDc4OQ==&mid=2247483961&idx=1&sn=2736aad509f08c20d82cfc08b62da27a&chksm=ec040463db738d754cce84506c098caca891b58740e1b38b3621f752f805eb1c02ebf0e2ac73&mpshare=1&scene=1&srcid=1226ZgmHAUTfeYMuLFPWyuHS#rd)