前言

\$ 1s -lisah diskimage/ total 107G 0 drwxr-xr-x 2 lrocha lrocha 65956 0 Apr 21 12:41 . 0 drwxr-xr-x 2 lrocha lrocha 65932 0 Apr 21 12:41 ... 65958 288K -rwxr-xr-x 1 lrocha lrocha 287K Apr 08 16:38 vmware-2.log

0 Apr 08 16:38 server.vmsd

安全客(bobao.360.cn)

安全客 (bobao.360.cn)

安全客 (bobao.360.cn)

Create a folder structure that will

be used to mount the different

partitions

65960 148K -rwxr-xr-x 1 Irocha Irocha 148K Apr 08 16:38 vmwa Vmware disk files obtained from 65961 164K -rwxr-xr-x 1 Irocha Irocha 164K Apr 08 16:38 vmwa a ESX host. The VMDK files are 65962 56K -rwxr-xr-x 1 lrocha lrocha 53K Apr 08 16:38 vmwa the ones we will be analyzing.

65959 144K -rwxr-xr-x 1 lrocha lrocha 144K Apr 08 16:38 vmwq

65969 512 VMDK into RAW format la Trocha

Volume ID: c9ac5bd80b198ab2eb44a428a5d165d3

Last Written at: 2017-03-31 15:20:06 (UTC)

Last Checked at: 2013-07-04 12:09:22 (UTC)

Last Mounted at: 2016-11-04 13:26:31 (UTC)

Unmounted properly Last mounted on: /boot

\$ cd ..

于查找事件发生的特定时间和顺序。

改。

步骤。

\$ mkdir disk_mount

\$ mkdir disk1_mount/home/

\$ mkdir disk1_mount/opt

\$ mkdir disk1_mount/swap \$ mkdir disk1_mount/tmp \$ mkdir disk1_mount/var

\$ fsstat -o 1050624 Server.raw Cannot determine file system type

S mkdir rawimage

(1.00/100%)

\$ cd rawimage \$ mmls server.raw DOS Partition Table Offset Sector: 0

65963 148K -rwxr-xr-x 1 lrocha lrocha 148K Apr 08 16:38 vmware-7. rog 65964 644K -rwxr-xr-x 1 lrocha lrocha 642K Apr 08 16:38 vmware.log 65966 106G armyray 1 lrocha lrocha 106G Apr 08 17:40 server-flat.vmdk 65967 12K Use the qemu-img ha Trocha 8.5K Apr 08 16:38 server.nvram 65968 4.0K utility to convert the ha lrocha 526 Apr 08 16:42 server. vmdk

65971 4.0K -rwxr-xr-x 1 lrocha lrocha 269 Apr 08 16:38 server.vmxf

\$ qemu-img convert -f vmdk -0 raw diskimage/server.vmdk rawimage/server.raw

本文将介绍Linux取证技术并予以实践。我将执行一系列步骤,以分析从运行红帽操作系统的受损系统获取的磁盘。我

首先是识别文件系统、挂载不同的分区、创建一个超级时间轴和一个文件系统时间轴。我还快速查看了工件,然后解

从ESX主机获取不同的磁盘文件时,你需要VMDK文件。然后,你将其移到你的实验室,这可能很简单,,因为你的笔

记本电脑使用SIFT工作站运行虚拟机。要分析VMDK文件,你可以使用"libvmdk-utils"软件包,该软件包包含用于访问。

存储在VMDK文件中的数据的工具。不过另一种方法是将VMDK文件格式转换为RAW格式。如果采用后一种方法,运行

挂了不同的分区。我将跳过如何获取磁盘的过程,不过可点击此处了解如何从VMware ESX主机获取磁盘镜像。

不同工具将会更容易,比如Sleuth Kit中的工具(将针对镜像大量使用)。要执行转换,可以使用QEMU磁盘镜像实用 程序。步骤如下图所示。

Linux取证技术实践

Units are in 512-byte sectors Description slot Start Length 0000000000 0000000000 000: Meta 0000000001 Primary Table (#0) 001: -----0000000000 0000002047 0000002048 Unallocated 002: 000:000 0001050623 0001048576 Linux (0x83) 0000002048 003: 000:001 0001050624 0075497471 0074446848 Linux Logical Volume Manager (0x8e) Linux Logical Volume Manager (0x8e) 004: 000:002 0146793933 0075497472 0222291404 0000006707 005: ----0222291405 0222298111 Unallocated \$ fsstat -o 2048 server.raw FILE SYSTEM INFORMATION File System Type: Ext4 First partition starting on sector Volume Name: 2048 is the /boot partition.

这是因为此分区的类型为0x8e(逻辑卷管理器)。如今,许多Linux发行版默认使用LVM(逻辑卷管理器)方案。LVM

The /boot partition cannot be on a logical volume group

(LVM) because the boot loader

cannot read

之后,你可以从磁盘镜像列出分区表,并使用"mmls"实用程序获取有关每个分区起始位置(扇区)的信息。然后,使

用起始扇区,并使用"fsstat"实用程序查询与文件系统相关的详细信息。从下图可以看到,"mmls"和"fsstat"实用程序能

够识别第一个分区"/boot", 类型为0x83 (ext4)。但是,"fsstat"无法识别从扇区1050624起始的第二个分区。

```
使用允许将硬盘驱动器或一组硬盘驱动器分配给物理卷的抽象层。物理卷组合成逻辑卷组,逻辑卷组可以分为具有挂
载点和ext4等文件系统类型的逻辑卷。
使用"dd"实用程序,你可以轻松看到LVM2卷的存在。为了使其可用于我们不同的取证工具,我们需要从LVM分区表创
建设备映射, 其将通过创建环回设备并进行映射来自动创建分区设备。然后, 我们使用管理LVM卷的不同实用程序,
比如"pvs"、"vgscan"及"vgchange"。下图显示了执行此操作的必要步骤。
                   $ sudo kpartx -r -a -v server.raw
                   add map loop0p1 (252:2): 0 1048576 linear /dev/loop0 2048
                                                                                Read the partition table and
                   add map loop0p2 (252:3): 0 74446848 linear /dev/loop0 1050624
                                                                                 create the devices maps.
                   add map loop0p3 (252:4): 0 146793933 linear /dev/loop0 75497472

    -a : Add partition mappings

                                                                                  -r : Read-only partition
                   $ sudo pvs
                                                                                  -v: Operate verbosely
                    PV
                                             Fmt Attr PSize PFree
                                      VG
                    /dev/mapper/loop0p2 rootVg lvm2 a--
                                                      35.50g
                                                               0
                                                                                Use "pvs" to see information
                    /dev/mapper/loop0p3 rootvg lvm2 a--
                                                      69,990
                                                                               about the LVM physical volumes.
                    /dev/sdb1
                                      mift-gm lvm2 a-- 39.76g
                                                                                Then use "vgscan" to scan all
                                                                               volume groups. A new volume
                   $ sudo vgscan -
                                                                                   group was found!
                    Reading all physical volumes. This may take a while...
                    Found volume group "rootvg" using metadata type lvm2
                                                                               Make the volume group "rootvg"
                   $ sudo vgchange -a y rootvg 
                                                                               active and then we can see the
                   6 logical volume(s) in volume group "rootvg" now active
                                                                               different devices that point to
                                                                                 the different partitions
                   $ sudo ls /dev/rootvg
                   lv_home lv_opt lv_root lv_swap lv_tmp lv_var
                   $ sudo file -sL /dev/rootvg/lv_root
                   /dev/rootvg/lv_root: Linux rev 1.0 ext4 filesystem data, UUID=d86fc54e-9bdc-4e4a-975b-
                   13fbe68cba78 (extents) (large files) (huge files)
```

\$ sudo mount -o ro /dev/rootvg/lv_opt disk1_mount/opt/ \$ sudo mount -o ro /dev/rootvg/lv_home disk1_mount/home/ \$ sudo mount -o ro /dev/rootvg/lv_tmp disk1_mount疫鹼雲(bobao.360.cn)

挂载磁盘后,通常可以通过创建时间轴来开始进行取证分析和调查。这是一个非常有用的关键步骤,因为其包含有关

以人可读格式修改、访问、更改及创建的文件的信息,称为MAC时间证据(修改的、访问的、更改的)。此活动有助

在我们创建我们的时间轴之前,值得注意的是,在Linux文件系统(如ext2和ext3)中,没有关于文件创建/生成时间的

\$ sudo mount -o ro /dev/rootvg/lv_root disk1_mount/ \$ sudo mount -o ro /dev/rootvg/lv_var disk1_mount/var/

在激活LVM卷组之后,我们有六个设备映射到六个挂载点,这些挂载点生成了该磁盘的文件系统结构。下一步是将不

同的卷作为只读挂载,因为我们将挂载正常设备进行取证分析。因为创建一个匹配分区方案的文件夹结构非常重要。

```
时间戳。只有3个时间戳。ext4中引入了创建时间戳。Dan Farmer和Wietse Venema的"The Forensic Discovery 1st
Edition"一书概述了不同的时间戳:
最后修改时间。对于目录,是指最后一次添加、重命名或删除条目的时间。对于其他文件类型,是指最后一次写入文
件的时间。
最后访问(读取)时间。对于目录,是指最后一次被搜索的时间。对于其他文件类型,是指最后一次文件被读取的时
间。
最后状态更改。状态更改的例子有: 所有者的变更、访问许可的更改、硬链接计数的更改或任何MAC时间的显式更
```

删除时间。Ext2fs和Ext3fs在dtime时间戳中记录文件被删除的时间,但并不是所有的工具都支持它。

创建时间:Ext4fs在crtime时间戳中记录文件被创建的时间,但并不是所有的工具都支持它。

不同的时间戳存储在包含在inode中的元数据中。Inode(索引节点)相当于Windows中的MFT条目号。在Linux系统中 读取文件元数据的一种方法是,首先使用,比如"ls-i file"命令,获取inode号,然后针对分区设备使用"istat",并指定

inode号。这将显示不同的元数据属性,包括时间戳、文件大小、所有者组和用户标识、权限及包含实际数据的块。

好的,我们先来创建一个超级时间轴。我们将使用Plaso来创建。Plaso是基于Perl的log2timeline的基于Python的重写。

超级时间轴的创建是一个简单的过程,其适用于不同的操作系统。但是,解释很难。最后一个版本的Plaso引擎能够解

析EXT 4,还能解析不同类型的工件,比如syslog消息、审计、utmp,等等。为创建超级时间轴,我们将针对已挂载的

Create a Supertimeline using the

Linux parsers.

z: Specify the timezone p : Activate post-processing

Use psort.py to convert the

supertimeline into your favorite

format. In this case I will use CSV

安全客 (bobao.360.cn)

(deleted-realfoc) means the file "/sbin/ping." was deleted and the inode was reallocated. The

安全客 (bobao.360.cn)

/\$OrphanFiles/Or metadata belongs to another file.

/usr/include/netapi.h (deleted-realloc)

/\$OrphanFiles/OrphanFile-149743 (deleted)
/bin/ping (deleted-realloc)
/usr/bin/ping.

/usr/include/a.h /usr/include/netapi.h (deleted-realloc)

/\$OrphanFiles/OrphanFile-149732 (deleted)

/tmp/k (deleted) -/SOrphanFiles/OrphanFile-149721 (deleted)

/\$OrphanFiles/OrphanFile-149712 (deleted)

/\$OrphanFiles/OrphanFile-149698 (deleted)

/tmp/ccmaq3gB.o (deleted-realloc)

/bin/ls (deleted-realloc)

/bin/ping
/SorphanFiles/or
/SorphanFiles/or
/SorphanFiles/or
/SorphanFiles/or
/sorphanFiles/or

/usr/sbin/ping. (deleted-realloc)

/usr/include/a.h

/bin/ping

/bin/ping

/usr/sbin

/usr/sbin/

/bin/ls

/bin/ls

/usr/bin/ls.

/tmp/k (deleted)

磁盘文件夹启动log2timeline并使用Linux解析器。这个过程将需要一些时间,当完成后,你将获得plaso数据库格式的 带有不同工件的时间轴。然后,你可以使用"psort.py"实用程序将它们转换为CSV格式。下图概述了执行此操作所需的

2017-04-08 19:54:49,566 [INFO] (MainProcess) PID:2508 <frontend> Starting storage process.
2017-04-08 19:54:49,567 [INFO] (MainProcess) PID:2508 <frontend> Starting collection process.
2017-04-08 19:54:49,569 [INFO] (MainProcess) PID:2508 <frontend> Starting worker processes to ext 2017-04-08 22:16:54,939 [INFO] (MainProcess) PID:2547 <frontend> Processing is done, waiting for storage to complete.
2017-04-08 22:16:56,951 [INFO] (StorageThread) PID:2553 <storage> [Storage] Closing the storage, number of events processed: 2017-04-08 22:16:56,957 [INFO] (MainProcess) PID:2547 <frontend> Storage is done. 2017-04-08 22:16:56,957 [INFO] (MainProcess) PID:2547 <log2timeline> Processing completed.

\$ sudo log2timeline.py -z Etc/GMT -t / -p --parsers linux timeline.plaso.disk1 disk1_mount/
Source path : /data/disk_mount

2017-04-08 19:54:49,481 [INFO] (MainProcess) PID:2508 <frontend> Starting extraction in multi pro

: False

\$ psort.py -z Etc/GMT timeline.plaso.disk1 -o L2tcsv -w filter-timeline.plaso.csv

Stored Events : 540674 Events Included : 540674

Duplicate Removals : 94280

Is storage media image or device

[INFO] Output processing is done.

[INFO]

[INFO]

月的头几天可能发生了一些事。

件。目标是能够根据不同的工件重新创建所发生的事情。

S cat fls.timeline.all

Sun Apr 09 2017 00:34:09

Sun Apr 09 2017 00:34:12

Sun Apr 09 2017 00:34:27

Sun Apr 09 2017 00:34:37 Sun Apr 09 2017 00:34:38

Sun Apr 09 2017 00:34:55

Sun Apr 09 2017 00:35:00

Sun Apr 09 2017 00:35:13

Sun Apr 09 2017 00:35:39 Sun Apr 09 2017 00:35:40

Sun Apr 09 2017 00:35:43

Sun Apr 09 2017 06:35:44

Sun Apr 09 2017 00:38:11

对其内容进行排序,并以人可读格式呈现结果。您可以对使用"kpartx"创建的每个设备映射执行此操作。为简洁起见, 下图仅显示了"/"分区的这一步。你需要为每个其他映射设备执行此操作。 5 sudo fls -m / -a -r /dev/rootvg/lv_root | mactime -b - > fls.timeline.root \$ sudo fls -m / -a -r -d /dev/rootvg/lv_root | mactime -b - > fls.timeline.deleted.root r : Recursively display directories -a : Display the "." and ".." directory entries -m : The string given as mnt will be prepended to the file names as the mounting point d: Display deleted entries only 安全客 (bobao.360.cn) 在我们开始分析之前,值得一提的是,在Linux系统中存在与调查相关的大量文件和日志。可用于收集和调查的数据量

可能因配置的设置以及系统执行的功能/角色的不同而各异。另外,不同的Linux操作系统遵循一种文件系统结构—

很多,但其中要做的一件事是针对挂载的磁盘运行"chkrootkit"工具。Chrootkit是由Nelson Murilo和Klaus Steding-

现在已生成超级时间轴和时间轴,我们可以开始分析了。在这种情况下,我们直接进入时间轴分析,在此提示,在四

在分析过程中,做到细致、耐心很有帮助,拥有全面的文件系统和操作系统工件知识也有裨益。有助于分析(超级)

情。有了这一信息,我们开始缩小(超级)时间轴的时间范围。本质上,我们将寻找与日期有时间接近的相关的工

时间轴的一件事是拥有一定的有关事件确实发生的时间的引领。在这种情况下,在此提示,在4月初可能发生了一些事

在分析时间轴后,我们发现了一些可疑活动。下图展示了使用"fls"和"mactime"生成的时间轴输出。有人删除了一个名

149706

18571

18571

149743

17583 17583

17583

18571 18571

149732

31358

31358

149745 149744

1347

149712

31359

10621

10621

31359

149697

149698

为"/tmp/k"的文件夹,并重命名了"ping"和"ls"等常用二进制文件,并将相同名称的文件放在了"/usr/bin"文件夹中。

Jessen创建的脚本集合,可让您检查磁盘是否存在任何已知的内核模式和用户模式rootkit。

0 .a.. -/rrwxr-xr-x 0

-/rrwxr-xr-x 0

-/drwxr-xr-x 0

-/rrw-r--r-- 0

332 m.c. r/rrw-r--r-- 0

332 m.c. r/rrw-r--r--

38200 .a., r/rrwsr-xr-x 0

38200 .a.. r/rrwsr-xr-x 0

38200 .a.. r/rrwsr-xr-x 0 332 .a., r/rrw-r--r-- 0 332 .a., r/rrw-r--r-- 0

7856 m.cb r/rrwxr-xr-x 0

7856 .a. r/rrwxr-xr-x 0 0 ma.b -/rrw-r--r- 0 0 macb -/rrw---- 0

12288 .a.. d/dr-xr-xr-x 0

12288 .a.. d/dr-xr-xr-x 0

12272 m.cb r/rrwxr-xr-x 0 12272 m.cb r/rrwxr-xr-x 0

109208 .a.. r/rrwxr-xr-x 0

109208 .a., r/rrwxr-xr-x 0

B: Ext&fs records the time the file was created in the crtime stamp. Not all tools are able to parse this value

Deletion time. Ext2fs and Ext3fs record the time a file was deleted in the dtime stamp.

[0x400dfc] fopen("/usr/include/a.h", "r")

[0x400e30] fopen("/tmp/filekxMDDN", "wb")

[0x400e60] fclose(0 <no return ...>

debugfs 1.42.9 (4-Feb-2014)

Group:

Blockcount: 8

Size of extra inode fields: 28

Generation: 1375904706

Fragment: Address: 0

Inode: 18571

File ACL: 0

Links: 1

EXTENTS:

名, 其有助于找到环境中更多的受损系统。

[0x400e18] srand(0x592faae9, 0x7f4e3a82d7b8, 0, 0)

[0x7f4e3a4dc8d4] --- SIGSEGV (Segmentation fault)

戳,但你可以将其与"debugfs"结合使用来获取。我们还检查了这个奇怪文件的内容是否是乱码。

[Oxfffffffffffffff] +++ killed by SIGSEGV +++

Type: regular

Extended attributes stored in inode body:

Directory ACL: 0

0

selinux = "system_u:object_r:usr_t:s0\000" (27)

[0x400e09] time(0)

12272 .a.. r/rrwxr-xr-x 0

0 m.c. d/drwx--x--x 0

0 m.c. -/drax--x--x 0

A : Last Access (read) time. For directories, the last time it was searched. For other file types, the last time the file was read.

M : Last Modification time. For directories, the last time an entry was added, renamed or removed. For other file types, the last time the file was written to.

C: Last status Change. Examples of status change are: change of owner, change of access permission, change of hard link count, or an explicit change of any of the MACtimes.

.a.. d/drax--x--x 0 .a.. -/drwx--x--x 0

0 .a.. -/rrwxr-xr-x 0

1.6.

共同标准排列不同的文件和目录。这称为文件系统层次结构标准(FHS)。熟悉这种结构有助于发现异常。要查看的东西

在开始查看结合了不同工件的超级时间轴之前,你还可以为ext文件系统层(包含有关已分配和已删除的文件及未分配

的inode的数据)创建传统时间轴。这分两步完成。首先,使用TSK中的"fls"工具生成body文件。然后,使用"mactime"

```
这需要进一步查看。查看时间轴后可以看到,"fls"的输出显示该条目已被删除。因为inode没有重新分配,所以我们可
以尝试查看文件的备份是否仍然驻留在日志中。日志概念是在ext3文件系统中引入的。在ext4中,默认情况下日志功能
为启用状态,并使用"data = ordered"模式。在这种情况下,我们也可以检查用于挂载文件系统的选项。为此,要查看
"/etc/fstab"。 我们可以看到使用的是默认值。这意味着,如果目录被删除与映像获取之间的时间间隔很短,那么我们
可能有机会从被删除的文件恢复数据。尝试恢复被删除的数据的一种方法是使用"extundelete"工具。下图显示了该步
骤。
                  $ sudo extundelete --restore-directory /tmp/k /dev/vg_root/lv_root
                  WARNING: Extended attributes are not restored.
                  Loading filesystem metadata ... 47 groups loaded.
                  Loading journal descriptors ... 21367 descriptors loaded.
                  Searching for recoverable inodes in directory /tmp/k ...
1677 recoverable inodes found.
                  Looking through the directory structure for deleted files
                  Failed to restore inode 149698 to file RECOVERED_FILES/tmp/k/du:Inode does not correspond to a regular file.
                  Restored inode 149722 to file RECOVERED_FILES/tmp/k/ping/ping.o
                 Restored inode 149723 to file RECOVERED_FILES/tmp/k/ping/ping
                  Restored inode 149701 to file RECOVERED_FILES/tmp/k/ping/ping.c
                  Restored inode 149702 to file RECOVERED_FILES/tmp/k/ping/Makefile
                  Restored inode 149703 to file RECOVERED_FILES/tmp/k/popen.c
                                                                          安全客 ( bobao.360.cn )
```

合法文件,这些文件匹配从"/tmp/k"恢复的文件的MD5。因为某些文件是ELF二进制文件,所以我们将这些文件复制到 了一个隔离的系统中,以便执行快速分析。我们可以使用"Itrace -i"和"strace -i"轻松启动二进制文件,其将拦截和记录 不同的函数/系统调用。查看输出后可以很容易发现有错误之处。这个二进制文件看起来不是正常的"ping"命令,它调 用fopen()函数来读取文件"/usr/include/a.h",并写入/tmp文件夹中的一个文件,其中文件名使用tmpnam()生成。最 后,其产生一个分段错误。下图显示了该行为。 \$ ltrace -i ./ping [0x400b99] __libc_start_main(0x400c54, 1, 0x7fffb21316c8, 0x401060 <unfinished ...>

[0x400e20] tmpnam(0x7fffb2131280, 0x7fffb213124c, 0x7f4e3a82d0c4, 0x7f4e3a82d0d0)

提供这些信息后,我们返回查看,发现文件"/usr/include/a.h"在文件"ping"被移动/删除前一刻被修改了。所以,我们可

以检查 "a.h"文件的创建时间——ext4文件系统的新时间戳——使用"stat"命令。默认情况下,"stat"不显示crtime时间

\$ sudo debugfs -R "stat <\$(stat -c %i usr/include/a.h)>" /dev/vg_root/lv_root

Mode: 0644

Size: 332

Number: 0 ctime: 0x58e98181:00fbe91c -- Sun Apr 9 00:34:09 2017 atime: 0x58e98184:63728da8 -- Sun Apr 9 00:34:12 2017 mtime: 0x58e98181:00fbe91c -- Sun Apr 9 00:34:09 2017 crtime: 0x58e9111f:b663d8c0 -- Sat Apr 8 16:34:39 2017

Version: 0x00000000:00000001

Size: 0

A normal ping doesn't invoke those syscalls and

doesn't need to open a file /usr/include/a.h !!!

Flags: 0x80000

Going back to the mounted image,

system using the stat command.

we can get the crtime from ext4 file

安全客(bobao.360.cn)

恢复的文件对更多了解发生了什么非常有用,可进一步帮助调查。我们可以计算文件MD5,验证其内容以及其是否是

NSLR数据库或Virustotal已知的。如果其是一个二进制文件,那我们可以使用"objdump"和"readelf"等工具针对该二进

MD5,我们发现其是与Red Hat一起分发的合法操作系统文件。但是,"/bin"文件夹中的文件,比如"ping"和"ls",不是

制文件算出字符串并推导出功能。我们还获取并查看了在时间轴中看到的在"/usr/sbin"上创建的不同文件。检查其

```
(0):680449
                                                                  The contents of this are not a
                                                                  normal header file. Very suspicious!
                  $ xxd usr/include/a.h | head
                  0000000: c3a4 a1a0 a2f5 9996 919b c2d0 8a8c 8dd0
                  0000010: 9d96 91d0 9996 919b d1f5 9b8a c2d0 8a8c
                  0000020: 8dd0 9d96 91d0 9b8a d1f5 938c c2d0 8a8c
                  0000030: 8dd0 9d96 91d0 938c d1f5 b8b3 b6bd bca0
                  0000040: cdd1 cdd1 ccc2 9ed1 97d3 938c d1d3 9b8a
                  0000050: d1d3 9996 919b d1d3 8f8c d1d3 919a 8b8c
                  0000060: 8b9e 8bdl d38f 9691 98dl f5f5 a4al a0a2
                  0000070: f58f 8cc2 d08a 8c8d d09d 9691 d08f 8cd1
                  0000080: f58f 8d90 9c8f 8cc2 9e8b 9092 d38f 8cd1
                  0000090: d391 9a8b 8c8b 9e8b d1d3 938c d1d3 9b8a
                                                                         受全害 (bobao.360.cn)
现在我们知道有人在2017年4月8日16:34创建了这个"a.h"文件,我们能够恢复被删除的其他几个文件。此外,我们发现
一些系统二进制文件似乎放错了,至少"ping"命令预期从"a.h"文件中读取了一些东西。有了这些信息,我们回头看看超
级时间轴,以便找到这个时候可能发生的其他事件。正如我提到的,超级时间轴能够解析来自Linux操作系统的不同工
件。在这种情况下,经过一些清理,我们可以看到,在相关时间内我们有来自audit.log和WTMP的工件。Linux
audit.log跟踪有关红帽系统的安全相关信息。基于预先配置的规则,Audit守护进程生成日志条目,以尽可能多地记录
有关系统上正在发生的事件的信息。WTMP记录有关登录和登出系统的信息。
这些日志显示,在文件"a.h"被创建及"ping"和"ls" 二进制文件被错放前一刻,某人使用根证书从从IP 213.30.114.42(假
IP) 登录了系统。
                  04/08/2017,16:33:07,Etc/GMT,M....CG_Audit log File Content Modification Time,-,-,[audit_type: USER_START pid: 2242] user pid=2242 uid=0 auid=0 ses=5 subj=sys...,[audit_type: USER_START pid: 2242] user pid=2242 uid=0 auid=0 ses=5 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg="op=login id=0 exe="/usr/sbin/sshd" hostname=213.30.114.42 addr=213.30.114.42
                  terminal=/dev/pts/0 res=success',2,06:/mnt/disk_mount/var/log/audit/audit.log,38,-,selinux,text_prepend:
```

04/08/2017,16:33:07,Etc/GMT,M..._EOG_Audit log File Content Modification Time,-,-, [audit_type: USER_LOGIN] pid: 2242] user pid=2242 uid=0 auid=0 ses=5 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login id=0 exe="/usr/sbin/f Supertimeline shows artifacts from audit.log and WTMP, tou could use the PID to correlate the different events. 04/08/2017,16:33:07,Etc/GMT,.A.. LOG.UTMP session Start Time,-,-,User: root,User: root Computer Name: 213.30.114.42 Terminal: pts/0 PID: 2242 Terminal_ID: 808416116 Status: USER_PROCESS IP Address: 213.30.114.42 Exit: 0,2,05:/mnt/disk_mount/var/log/wtmp,755,-,utmp,text_prepend: / 安全客(bobao.360.cn)

现在我们有了网络指标。下一步,我们应开始查看我们的代理和防火墙日志,以获取关于IP地址的痕迹。同时,我们可

以继续我们的时间轴分析,以查找有关的其他工件,并对找到的文件进行深入的二进制分析,创建IOC,比如Yara签

To tear down the image, unmount \$ sudo umount disk1_mount/opt/ the different partitions. Then \$ sudo umount disk1_mount/tmp/ deactivate the volume group and \$ sudo umount disk1_mount/ finally delete the device mappers \$ sudo vgchange -a n rootvg

安全客 (bobao.360.cn)

完成分析和取证工作后,你可以解挂分区,取消激活卷组并删除设备映射。下图显示了这些步骤。

O logical volume(s) in volume group "rootvg" now active

\$ sudo umount disk1_mount/home/ \$ sudo umount disk1_mount/var/

\$ sudo kpartx -d rawimage/server.raw

loop deleted : /dev/loop0

```
后记
Linux取证与微软Windows取证相比有很大不同,也很有趣。有趣的部分(调查)是熟悉Linux系统工件。安装原始的
Linux系统,获取磁盘并查看不同的工件。然后利用一些工具/方法攻破机器,获得磁盘并再次进行分析。你可以通过这
种方式反复练习自己的Linux取证技能。
```