**Casino**

1. **信息收集**
   1. **主机发现**

利用arp-scan对网络内主机进行探测，这里除192.168.6.158外均是该网络中的已知服务器，因此192.168.6.158即是casino靶机。

电脑萤幕画面

描述已自动生成

注：为什么直接锁定到了靶机？

解释：这里我们已知casino是有漏洞的需要进行攻击，因此实际上arp-scan在此只起到了查找casino网络ip的作用。在实际攻击场景中，我们并不清除哪个主机有漏洞，因此每个探测到的主机都要尝试攻击。

* 1. **端口扫描**

利用nmap对casino

1. **排查思路**

鉴于挖矿病毒上述特征，因此对挖矿病毒的排查主要从1）大量占用CPU资源的异常进程，和2）与外部通信的异常端口入手，辅以其它排查方式共同对挖矿病毒进行全面排查。

1. **排查过程**
   1. **进程排查**

输入top命令查看当前CPU占用率排名，发现存在一个pid为1201的未知可疑进程占用了大量CPU。

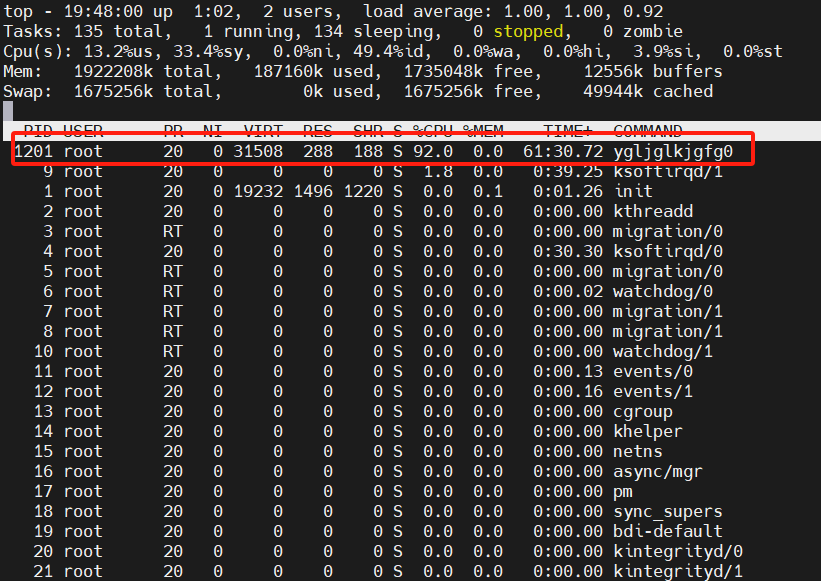


图- 1

利用ll /proc/1201/exe查看该进程的可执行文件路径，发现指向/bin/ygljglkjgfg0，查看该文件的详细信息发现其创建于Feb 24 2002。



图- 2

执行命令mkdir /tmp/virus以在/tmp目录下创建一个临时的目录virus，执行命令cp /bin/ygljglkjgfg0 /tmp/virus/bin-ygljglkjgfg0存放该样本。接下来对样本进行分析。

***注：为什么要复制存放该样本？为什么要分析该样本文件而不是原文件？***

***解释：一是分析原文件可能造成病毒进一步执行；二是原文件正在执行，其可能会不断修改自身或环境，从而干扰分析；三是易于后期取走该样本以留证。***

输入strings /tmp/virus/bin-ygljglkjgfg0 > /tmp/virus/bin-ygljglkjgfg0.log收集文件的部分源文件内容，并利用grep “crontab” /tmp/virus/bin-ygljglkjgfg0.log对定时任务关键字“crontab”和“cron”进行过滤，发现其相关路径/etc/cron.hourly/gcc.sh和/etc/crontab（该路径为存放系统定时任务脚本路径），收集该信息并记录。



图- 3



图- 4

* 1. **端口排查**

输入netstat -atunlp命令查看当前端口及网络连接情况，发现存在pid为1190的可疑进程gam\_server异常打开了43272和53535端口。利用ll /proc/1190/exe查看该进程的可执行文件路径发现为/bin/gontwkvdypwt，同样将其复制到/tmp/virus/下作为样本，利用strings和grep命令对定时任务关键字“crontab”、 “cron”过滤，发现其相关路径/etc/cron.hourly，收集该信息并记录。

***注：为什么strings该文件能查看到其相关文件路径（定时任务）？***

***解释：恶意文件为便于隐藏，通常将所有代码集成在一个可执行文件中，再通过执行该文件统一进行病毒扩散、权限维持以及攻击目的代码（在挖矿病毒中为侵占计算资源实现挖矿）的执行等。因此通过分析该占用CPU的恶意可执行文件往往可以找到其相关恶意文件的路径。***

电脑萤幕画面

描述已自动生成

图- 5



图- 6

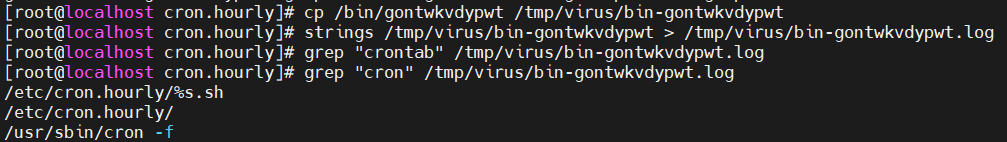


图- 7

* 1. **初始化任务排查**

利用收集到的可疑进程和可执行文件名进行搜索，发现以下初始化任务脚本，S90ygljglkjgfg0和 K90ygljglkjgfg0均指向了ygljglkjgfg0，利用cat /etc/rc.d/init.d/ygljglkjgfg0查看该初始化脚本发现其无论在任何运行级别下都可以运行，执行的文件路径为/bin/ygljglkjgfg0（图- 8表明该初始化脚本在运行级别0123456下均存在）。

文本

描述已自动生成

图- 8



图- 9



图- 10

文本

描述已自动生成

图- 11

虽然没有搜索到和gontwkvdypwt相关的文件，但是在检查初始化任务文件目录/etc/rc.d/rc\*.d/时发现在任何运行级别下均存在一个乱码文件名为twpydvkwtnog，其恰巧是gontwkvdypwt的反序拼写。利用cat /etc/rc.d/init.d/twpydvkwtnog查看该初始化脚本发现其执行的文件路径为/bin/gontwkvdypwt。

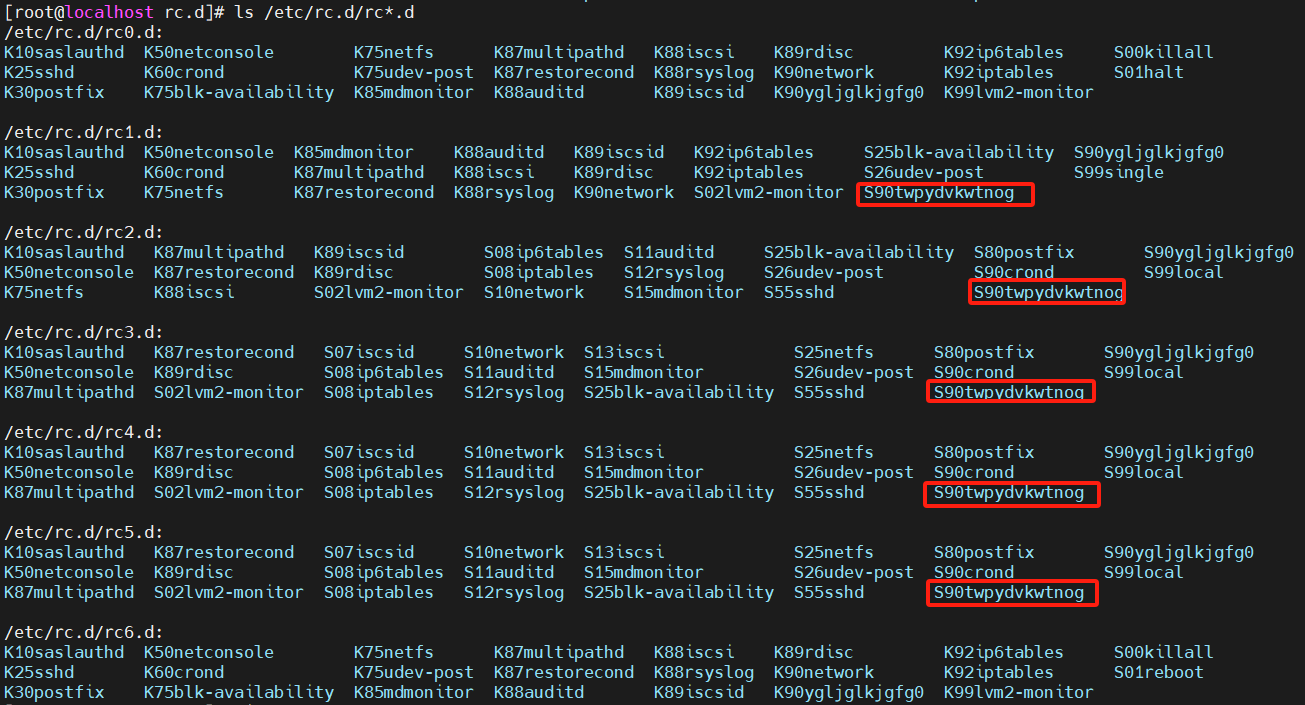


图- 12

文本

描述已自动生成

图- 13

收集以上信息并记录。

* 1. **定时任务排查**

先前通过strings查看bin-ygljglkjgfg0和bin-gontwkvdypwt已经发现其可能存在定时任务位于路径/etc/cron.d/和/etc/cron.houly/，该路径也是系统定时任务脚本的存放路径，因此检查该路径。

首先查看/etc/cron.d/，其下共有两个文件0hourly和raid-check，执行cat /etc/cron.d/0hourly和cat /etc/cron.d/raid-check后发现，0hourly为系统安排每小时运行一次的定时任务脚本，其每小时将执行/etc/cron.hourly/下的所有脚本；raid-check是系统定期检查和维护RAID 阵列的脚本。接下来分析排查/etc/cron.hourly/路径下文件。



图- 14

文本

描述已自动生成

图- 15

文本

描述已自动生成

图- 16

文本

描述已自动生成

图- 17

执行ls -a /etc/cron.hourly发现其有三个文件0anacron、gcc.sh、twpydvkwtnog.sh，鉴于**3.2**和**3.3**部分利用strings对可疑可执行文件分析所发现的信息，应着重对gcc.sh和twpydvkwtnog.sh脚本排查。执行cat /etc/cron.hourly/0anacron、cat /etc/cron.hourly/gcc.sh、cat /etc/cron.hourly/twpydvkwtnog.sh查看三个文件，0anacron为系统异步定时任务脚本，无可疑之处；gcc.sh定时启用网卡并将libudev.so文件复制并更名为libudev.so.6；twpydvkwtnog.sh定时将twpydvkwtnog文件复制并更名为whdlwwenev。

***注：为什么没有找到gcc.sh将libudev.so复制并重命名的libudev.so.6文件？***

***解释：因为该命令执行失败。***



图- 18

文本

描述已自动生成

图- 19

文本

描述已自动生成

图- 20

文本

描述已自动生成

图- 21

最后分析/etc/crontab文件，执行命令cat /etc/crontab查看该文件，发现其每3分钟将执行一次gcc.sh这个恶意脚本。

文本

描述已自动生成

图- 22

收集上述信息并记录。

1. **清理过程**

首先修改恶意程序的可执行权限，执行命令chmod 600 /bin/ygljglkjgfg0、chmod 600 /bin/gontwkvdypwt。为避免部分病毒隐藏而无法查看，执行umount -l /proc命令卸载/proc磁盘。

***注：为什么需要卸载/proc才能查看隐藏的部分病毒恶意文件？***

***解释：proc文件系统是伪文件系统，提供了关于系统进程和内核状态的信息。恶意软件可以通过修改/proc文件系统的内容来隐藏特定的文件、进程和网络连接，这通常是rootkit以及恶意内核模块常见的手法；此次XorDDoS利用此/proc文件系统来隐藏文件和进程。当卸载/proc文件系统后，这些隐藏文件才会暴露出来。***



图- 23



图- 24



图- 25

执行rm -f /etc/rc.d/rc\*.d/K90\*和rm -f /etc/rc.d/rc\*.d/S90\*命令删除初始化任务目录下所有相关文件（在初始化脚本/etc/rc.d/init.d/ ygljglkjgfg0和/etc/rc.d/init.d/twpydvkwtnog可以看到其相关初始化任务优先级全部被设置为90，因此删除所有S90和K90前缀的初始化脚本即可）。然后利用chkconfig network on恢复网络的初始化脚本。

***注：为什么还要执行chkconfig network on恢复网络？***

***解释：因为恶意文件中有启用网络功能的代码，将其删除后网络功能将会不正常，因此需要手动恢复一次网络。***



图- 26

图片包含 图形用户界面

描述已自动生成

图- 27

接下来删除恶意定时任务脚本。执行ls /etc/cron.hourly/发现在卸载/proc磁盘后共有6个以乱码命名的恶意定时任务脚本，执行rm -rf /etc/cron.hourly/\*.sh删除6个恶意脚本。然后利用find命令全局搜索新出现的恶意文件名，结合exec将其相关文件全部删除。



图- 28

文本

中度可信度描述已自动生成

图- 29

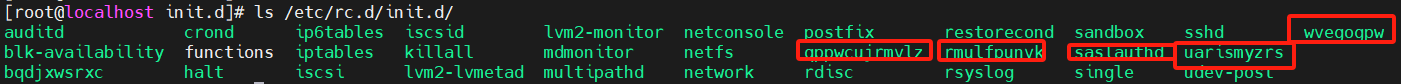


图- 30

文本

描述已自动生成

图- 31

文本

描述已自动生成

图- 32

然后执行rm -rf /lib/libudev.so删除gcc.sh中指向的恶意库文件，并删除/etc/crontab脚本中定时执行gcc.sh的命令。



图- 33

文本

描述已自动生成

图- 34

接下来查找所有在当日生成的恶意文件，找到其中的病毒文件并清除。执行find / -ctime 0 2>/dev/null，分析后发现还有文件名为乱码的恶意文件在/tmp（图- 35）和/bin（图- 37）中。

文本

描述已自动生成

图- 35

对/bin目录和/tmp目录下的剩余所有病毒进行清除，执行find /bin/ -ctime 0 | tail -n 10 | xargs -I {} rm -rf {}和find /tmp/ -ctime 0 | tail -n 6 | xargs -I {} rm -rf {}命令删除/bin和/tmp中剩余的恶意文件。

***注：为什么此时/bin/下没有ygljglkjgfg0恶意文件了？***

***解释：证明该文件并非是真正的恶意文件主体，只是一个套壳的“替罪羊”，由于/proc的卸载，套壳行为无法继续实现因此/bin/ygljglkjgfg0消失了。***



图- 36

图形用户界面, 文本, 应用程序

描述已自动生成

图- 37

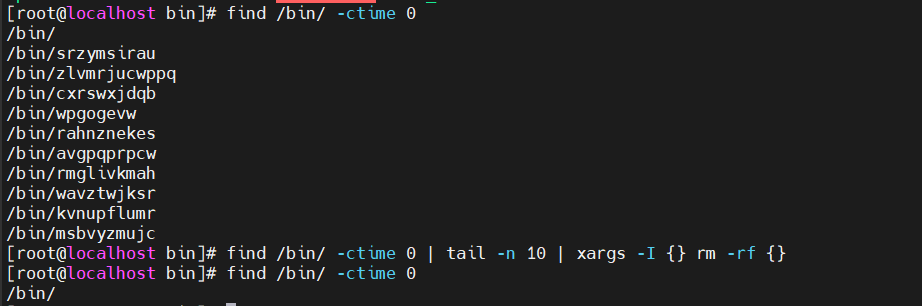


图- 38

最后执行mount proc proc /proc将/proc磁盘挂载回来。

重启后利用top查看是否还有占用CPU的异常进程，查看/bin下是否有当天生成的病毒文件，查看是否还有恶意的初始化任务和定时任务。如果均没有，则证明病毒清理完毕。

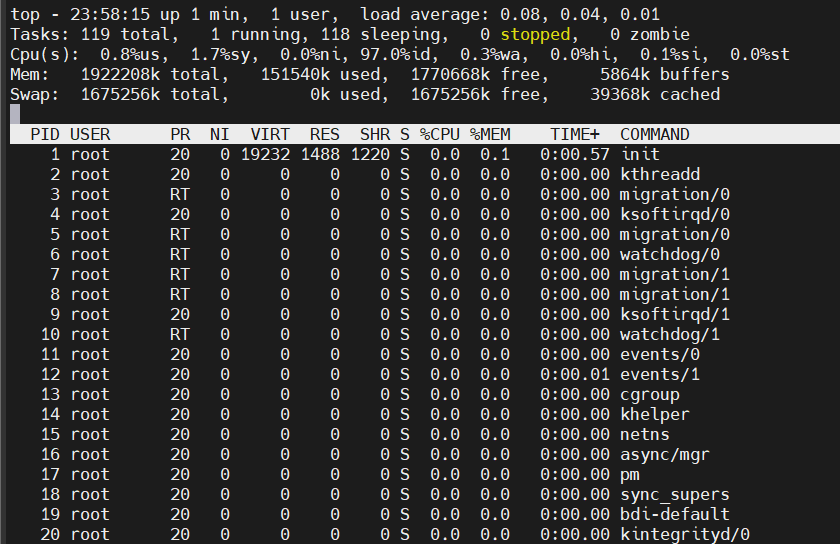


图- 39



图- 40

图片包含 文本

描述已自动生成

图- 41

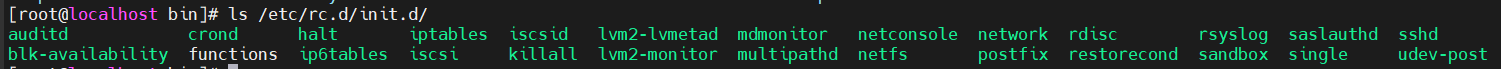


图- 42



图- 43

1. **溯源过程**

溯源的过程主要依靠排查系统日志实现，其中/var/log/messages中记录了Linux系统绝大多数重要信息，系统出现问题后首先检查的就应是该日志文件。

在本次挖矿病毒的排查清理过程中，该病毒可能会被日志记录的关键点包括1）所有恶意文件创建的最早时间；2）异常进程： ygljglkjgfg0（pid1201），gam\_server（pid1190）；3）异常端口：43272（gam\_server），53535（gam\_server）；4）异常文件名：gcc.sh、ygljglkjgfg0、gontwkvdypwt、twpydvkwtnog、libudev等。

首先执行cp /var/log/messages ~/messages.log复制一份日志以便于后续分析。首先对1）恶意文件创建时间在日志中过滤进行查找，根据图- 2中恶意文件/bin/ygljglkjgfg0的创建时间为Feb 24 2002，因此执行cat ~/messages.log |grep “Feb 24” > ~/ygljglkjgfg0.log。然后执行vi ygljglkjgfg0.log查看过滤出的日志。



图- 44



图- 45

根据分析发现，当日的日志中存在可疑ip：103.64.12.132具有多次ssh登录的异常行为，且多次运行chkpwd命令检查系统密码，并在后续创建了计划任务、恶意gcc.sh、libudev.so文件。

图形用户界面, 文本

描述已自动生成

图- 46

图形用户界面, 文本, 应用程序

描述已自动生成

图- 47

图片包含 应用程序

描述已自动生成

图- 48

综上所述确定此次实际通过ssh弱口令爆破进入到系统，并留下XorDDoS恶意文件。

1. **加固方法**
   1. **修改弱口令为强口令**
      1. **提高密码复杂度**
   * 建议密码长度至少为12个字符
   * 密码应包含大小写字母、数字和特殊字符的组合
   * 避免使用常见的词语、短语、生日或其他容易被猜到的信息
     1. **提升密码更换频率**
   * 建议每隔90天更换一次密码
   * 不得重复使用最近5次的旧密码
     1. **采用多因素认证（MFA）**
   * 启用多因素认证，提高系统的访问安全性
   1. **SSH访问控制**
      1. **基于白名单的访问控制**
   * 仅允许特定的IP地址或IP范围访问SSH服务
     1. **修改SSH默认端口**
   * 将SSH默认端口22改为一个非标准端口（如14762），以减少被自动扫描和攻击的风险
   1. **其他安全措施**
      1. **系统更新和补丁**
   * 定期更新系统和安装安全补丁，确保系统没有已知的漏洞
     1. **用户和权限管理**
   * 最小化用户权限，仅授予必要的访问权限，避免权限滥用