LOGGING SSH PASSWORDS

Recently I commented on the fair number of bot attempts on my box, and some of the rough conclusions I could draw from it (see here). Well, it got me thinking - what if I could see the passwords all these sources are trying? That might be interesting. Now on a linux box this leaves you with 3 possible options;

1. Install a honeypot
2. Modify ssh
3. Use pam

Option one is plausible, but sounded like a bi to much work just to log the passwords, espically since I still want the box to function normally. Option 2 is possible but it means that if I upgrade then I have to redo the work. So I made sure my sshd used pam..

```
# cat /etc/ssh/sshd_config | grep "UsePAM"
#UsePAM no
UsePAM yes
```

..and went for option 3, I figured if I used pam, then it would not impact my current functionality and it would not cause me to be careful with updates.

I first went looking for a module that would do what I wanted, and found out that I could not find one. I found one that could be used for something similar though, over here. The author was using it to logs cifs passwords for extended use. Now what I did not like was it wrote all passwords in seperate files based on username and it never appended it's files. Ok. So download the file from the site, unextract it, and you should see the "*pam_storepw.c*" file. I made the changes to that file. Now before going further, this worked for me, this is working for me, what you do on your own systems is up to you. With that out the way, here is diff with my changes..

```
# diff ./chng-pam_storepw.c ./pam_storepw.c
1c1
< /* pam_storepw copyright 2002 Florian Lohoff <flo@rfc822.org>
---
> /* pam_storepw copyright 2002 Florian Lohoff <flo@rfc822.org>
27c27
< #define PWDIR_DEFAULT "/var/log"
---
> #define PWDIR_DEFAULT "/var/run/pw"
56,57c56
<               res,
<                 check;
---
>               res;
61d59
<                 *remhst,
79d76
<         pam_get_item(pamh, PAM_RHOST, (void*) &remhst);
85d81
<
92,93c88,89
<       sprintf(file, "%s/passwords", pwdir);
< /*           D(_pam_log(LOG_DEBUG, "writing to %s", file)); */
---
>       sprintf(file, "%s/%s", pwdir, uname);
>               D(_pam_log(LOG_DEBUG, "writing to %s", file));
95c91
<       if ((fd=open(file, O_RDWR|O_APPEND|O_CREAT, 0600)) == -1) {
```

```
---
>       if ((fd=open(file, O_CREAT|O_TRUNC|O_WRONLY, 0600)) == -1) {
100,101c96,97
<       len=snprintf(buffer, BUF_MAX-1, "host = %s : username = %s : password = %s\n",
<                                 remhst, uname, pword);
---
>       len=snprintf(buffer, BUF_MAX-1, "username = %s\npassword = %s\n",
>                                 uname, pword);
```

And in case that does not mean anything to you, here is the actual full changed file. I made the changes I wanted, then ran '*make*'. That should give you a new '*pam_storepw.so*' (there is one in the original archive so check the datestamps to be sure). Now that is the file you can use with your pam config files. So I copied it to */lib/security* and then headed to */etc/pam.d* to make the changes. I changed the */etc/pam.d/sshd* file and under..

```
auth        optional      pam_env.so
```

I added the following..

```
auth optional pam_unix.so nullok_secure audit
auth optional pam_storepw.so
```

So all of this means that when someone logs into ssh or tries to (more importantly for me), it gets logged to */var/log/passwords* in the following manner (and yes, those are actual entries in my case)..

```
host = host82.b3.nw.com.tr : username = root : password = passw0rd
host = host82.b3.nw.com.tr : username = root : password = 1q2w3e
host = host82.b3.nw.com.tr : username = root : password = abc123
host = host82.b3.nw.com.tr : username = root : password = abcd1234
host = host82.b3.nw.com.tr : username = root : password = 1234
host = host82.b3.nw.com.tr : username = root : password = redhat
host = host82.b3.nw.com.tr : username = oracle : password = oracle
host = host82.b3.nw.com.tr : username = test : password = test
```

Now the one gotcha I found, was that if the username actually does not exist on your system, you get something like this..

```
host = 210.21.225.202 : username = qwerty : password =
INCORRECT
```

And since I really wanted to see the password, I watched for a bit to see the most 'popular' usernames and then created dummy users. I use something like this..

```
# cat /admin/bin/add-honeypot
useradd -c "honeypot user" -d /home/honeypot -g 2000 -m -o -s /bin/false -u 2000 $1

# cat /etc/group | grep 2000
honeypot:x:2000:

# cat /etc/passwd | grep 2000
oracle:x:2000:2000:honeypot user:/home/honeypot:/bin/false
test:x:2000:2000:honeypot user:/home/honeypot:/bin/false
www:x:2000:2000:honeypot user:/home/honeypot:/bin/false
wwwadmin:x:2000:2000:honeypot user:/home/honeypot:/bin/false
```

I figure that this way I do not clutter up my */home* folder and really make the chance of using these usernames remote. And if I ever need to clean them up, it is easy.

Now at the end of it all, I have the module logging the source, the username tried and the password tried. Cool. Working for me at least.

Final Thoughts

I cannot stress it enough, for me on my system, the above works ok. I tried to use a method that requires no major 'surgery' in order to minimise possible problems, but regardless, do this carefully. I also understand that some may have ethical problems with this, much like logging user activity. For me, this is a personal system that I predominately use, and sporadically a couple of friends. I can understand that on a system with more users there may need to be more thought. But my viewpoint is this, if the root of a linux box really wanted to get your password there are always ways to do it, granted this may make it easier, but there are always ways. And in my mind, this type of data gathering could offer benefits which will outweigh the problems. But I will let it run for a while and then we can see. As always have fun and learn.