Logon Scripts

# .bashrc based persistence

Text

Description automatically generated

The first step I took was to download the persistence script from https://thepcn3rd.blogspot.com/2021/11/t1546-unix-shell-configuration.html . I then replaced the default SSH key data with my own.

Text

Description automatically generated

Because the original script relies on the ~/.ssh directory already existing I added the above code to the script. This code is supposed to check if the .ssh directory exists, then create it if it doesn’t. It then is supposed to create the authorized\_keys file if it doesn’t exist. However, when I run the code it always takes path 1, regardless of whether .ssh exists or not. I don’t know what the problem is. Due to the issues with the code, I decided to use the original script instead.

Text

Description automatically generated

I then encoded the original script (with my SSH key) into base64.

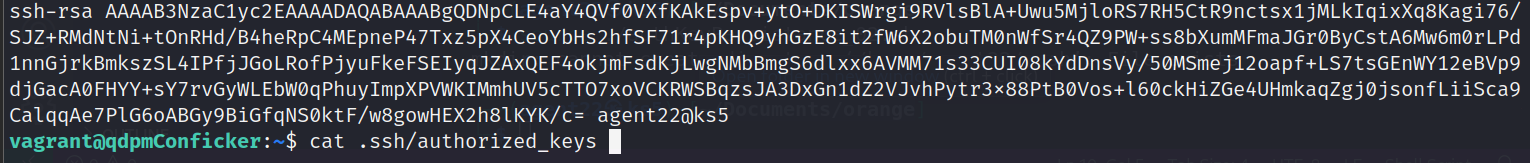
Text

Description automatically generated

Next, I copied the base64 version of the script to the ~/.bashrc file for the vagrant user. I then created the above short script. The script echoes the base64 data into a base64 decoder, then runs the code in bash.



I then ran the .bashrc file.



Next, I checked to see if my SSH key had been added. It had.

Text

Description automatically generated

I then confirmed that I could access the qdpm webserver over SSH using my SSH key. After entering the password for my SSH key (required by my kali box) I was allowed into the qdpm server.

In the future I would probably move the script placed in the .bashrc file to a separate file named “backup” or something innocuous, hidden in an obscure directory. Then call that script file from .bashrc.

# Crontab Based Persistence

Text

Description automatically generatedThe first step I took to establish persistence with crontab was to locate my payload. I then copied that payload to my working directory. This payload is a simple web shell backdoor.

Text

Description automatically generated

I then renamed the payload for simplicity sake.

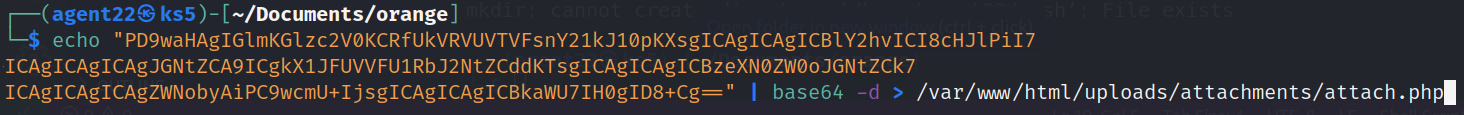
A screenshot of a computer

Description automatically generated with medium confidence

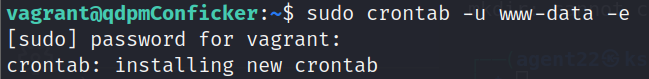
Using the tr (translate) command I moved all of the php code to one line. This was accomplished by replacing all the hidden \n (new line) characters with spaces.Text

Description automatically generated

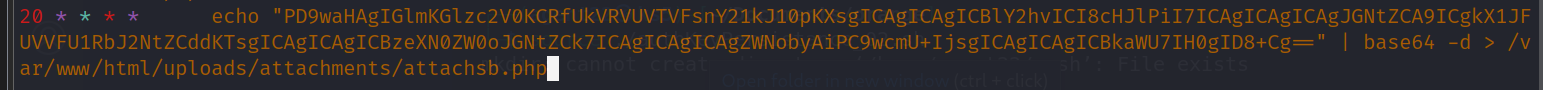
I then encoded this one line of php code into base64.



I then formed the script I needed to add to the crontab on my own box. I then copied that script to my clipboard. This script decodes the base64 encoded backdoor then copies it to the attach.php file. This file is located in the /var/www/html/ directory which means it can be accessed over http.



Next, I opened the crontab file for the www-data user.



Next, I configured crontab to run my command at 20 minutes past the hour 24/7. I then copied in my script and saved the file.

Graphical user interface, text, website

Description automatically generated

After waiting for a few minutes, I confirmed that the backdoor was available and working.

Again, I’d probably place the code to be run by crontab in a different file, then call that file from crontab. This would look less suspicious then a long base64 string.