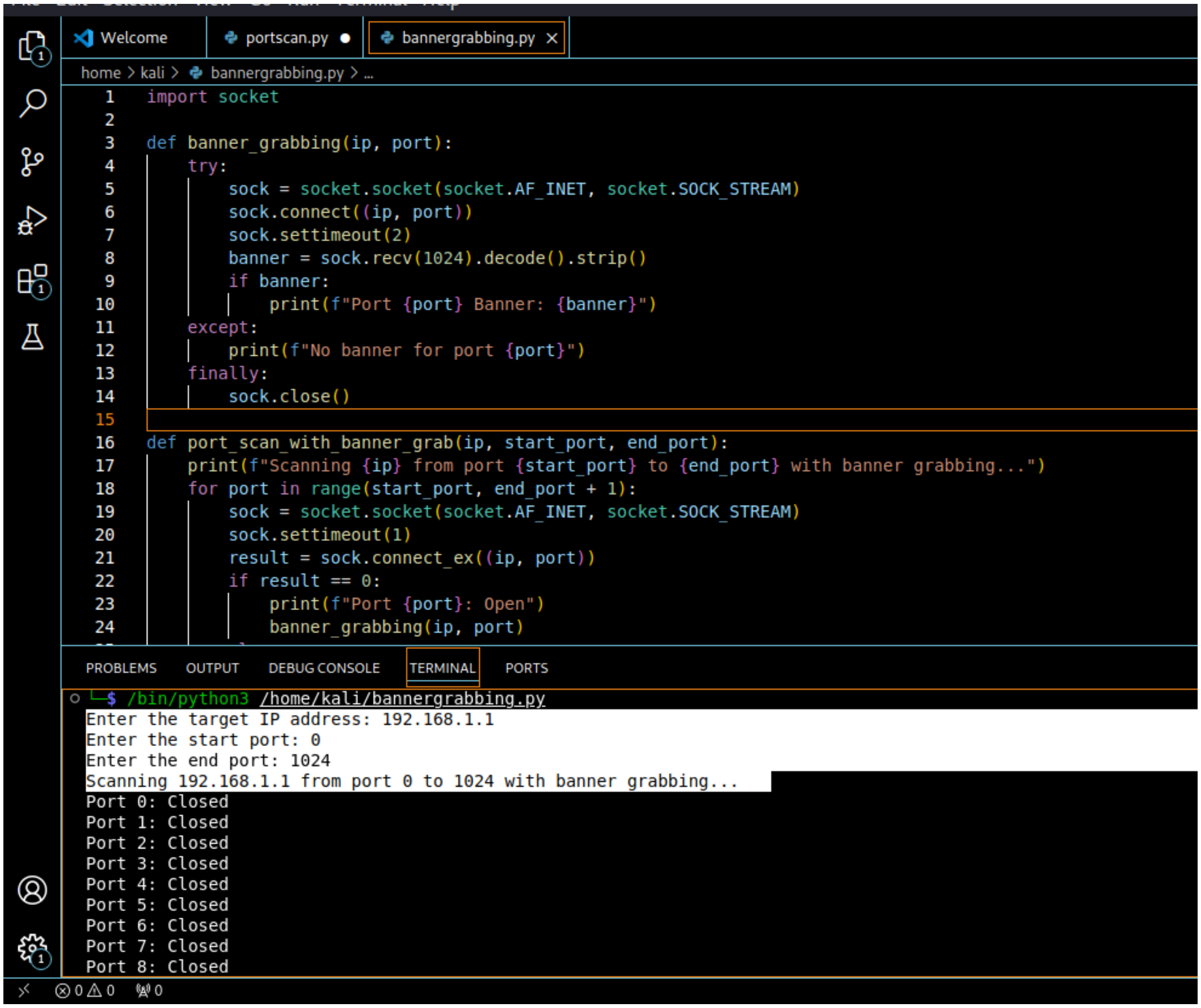
Banner Grabbing



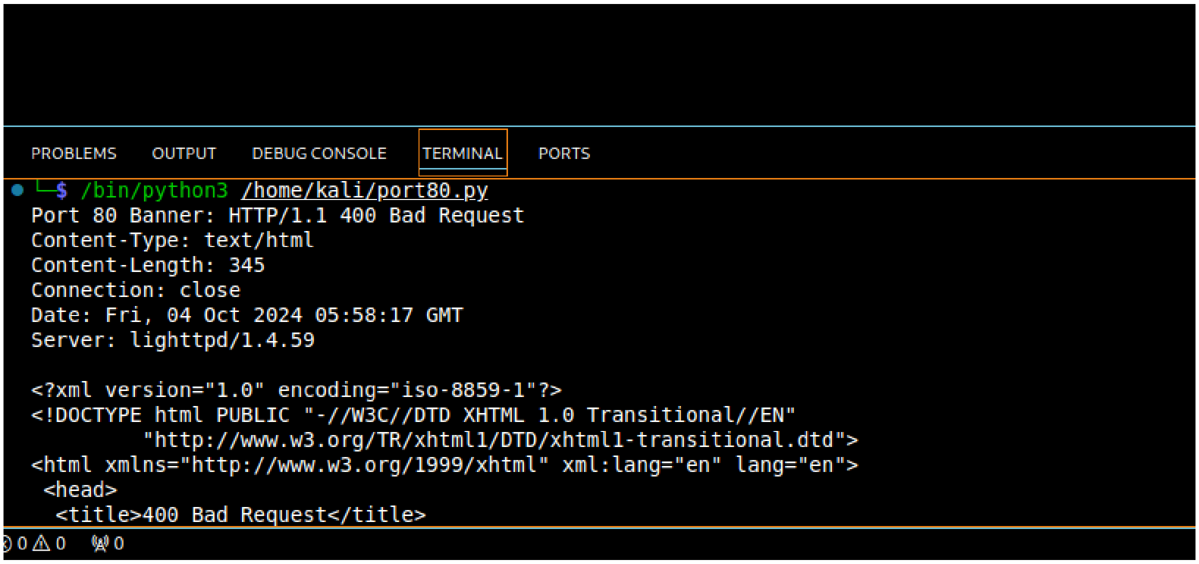
```python
import socket

def banner_grabbing(ip, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((ip, port))
        sock.settimeout(2)
        banner = sock.recv(1024).decode().strip()
        if banner:
            print(f"Port {port} Banner: {banner}")
    except:
        print(f"No banner for port {port}")
    finally:
        sock.close()

def port_scan_with_banner_grab(ip, start_port, end_port):
    print(f"Scanning {ip} from port {start_port} to {end_port} with banner grabbing...")
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((ip, port))
        if result == 0:
            print(f"Port {port}: Open")
            banner_grabbing(ip, port)
```

```
└─$ /bin/python3 /home/kali/bannergrabbing.py
Enter the target IP address: 192.168.1.1
Enter the start port: 0
Enter the end port: 1024
Scanning 192.168.1.1 from port 0 to 1024 with banner grabbing...
Port 0: Closed
Port 1: Closed
Port 2: Closed
Port 3: Closed
Port 4: Closed
Port 5: Closed
Port 6: Closed
Port 7: Closed
Port 8: Closed
```

Attempting for banner grabbing for the ports that are open, 53, 80,443 .



```
● └─$ /bin/python3 /home/kali/port80.py
  Port 80 Banner: HTTP/1.1 400 Bad Request
  Content-Type: text/html
  Content-Length: 345
  Connection: close
  Date: Fri, 04 Oct 2024 05:58:17 GMT
  Server: lighttpd/1.4.59

  <?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
           "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
    <title>400 Bad Request</title>
```

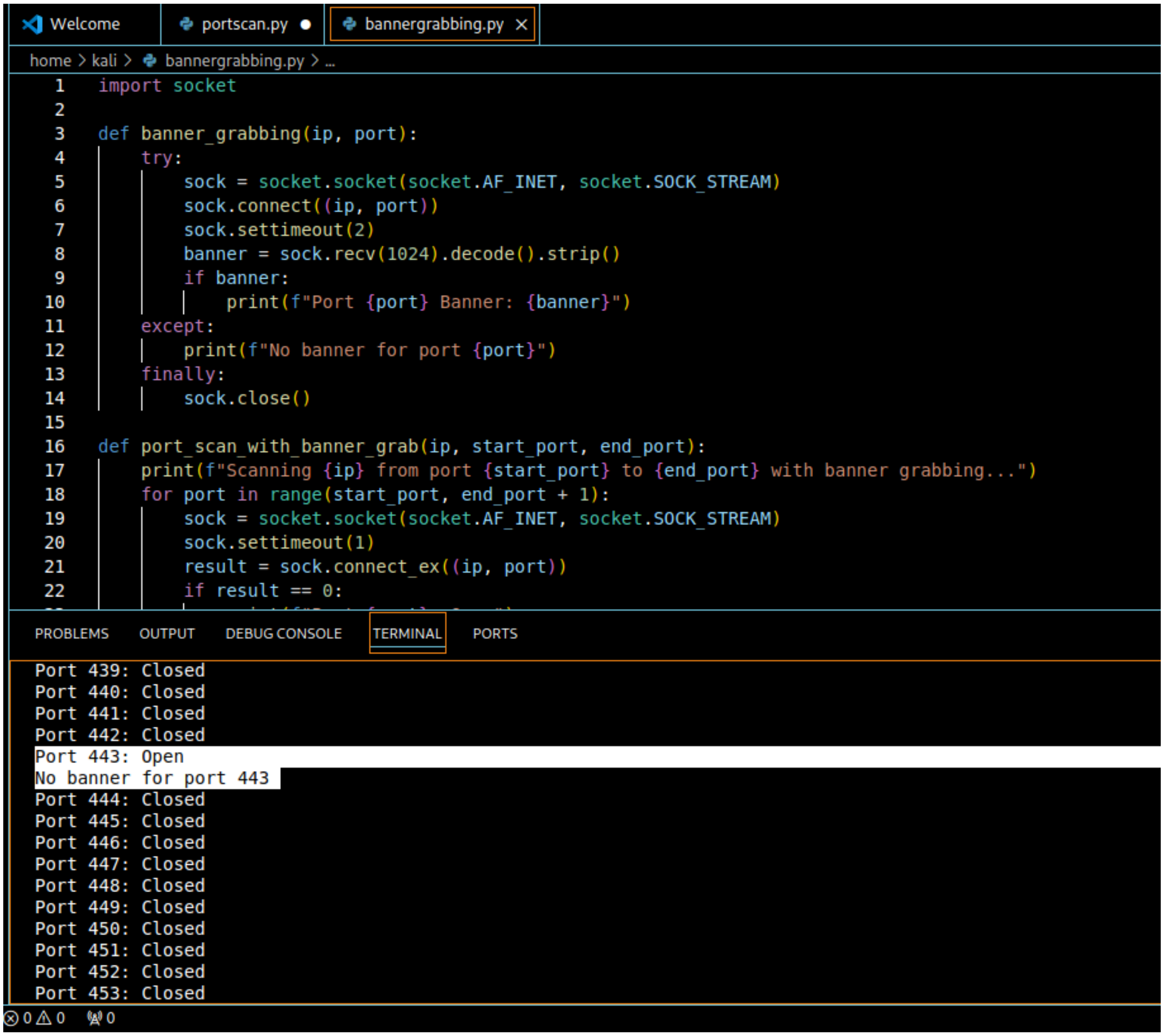
1.What is banner grabbing, and why is it useful in network security assessments? (show in your results)

Banner grabbing is a technique used to gather information about a network service by connecting to its open ports and reading the information ("banner") sent by the service. Banners often include metadata, such as the software type and version, that a server reveals to clients upon establishing a connection.

Usefulness in Network Security Assessments:

1. **Identifying Vulnerabilities**: Banner grabbing helps identify the software and versions running on a server, which is crucial for determining known vulnerabilities. For example, if a service is using an outdated version, it might have unpatched vulnerabilities that could be exploited.
2. **Service Detection**: Understanding what services are running on a given port helps assess the attack surface of a network.
3. **Network Profiling**: It is used during penetration testing to gather detailed information about an organization's infrastructure

Here in the above image we see that the server used in lighttpd/1.4.59



```python
import socket

def banner_grabbing(ip, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((ip, port))
        sock.settimeout(2)
        banner = sock.recv(1024).decode().strip()
        if banner:
            print(f"Port {port} Banner: {banner}")
    except:
        print(f"No banner for port {port}")
    finally:
        sock.close()

def port_scan_with_banner_grab(ip, start_port, end_port):
    print(f"Scanning {ip} from port {start_port} to {end_port} with banner grabbing...")
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((ip, port))
        if result == 0:
```

```
Port 439: Closed
Port 440: Closed
Port 441: Closed
Port 442: Closed
Port 443: Open
No banner for port 443
Port 444: Closed
Port 445: Closed
Port 446: Closed
Port 447: Closed
Port 448: Closed
Port 449: Closed
Port 450: Closed
Port 451: Closed
Port 452: Closed
Port 453: Closed
```

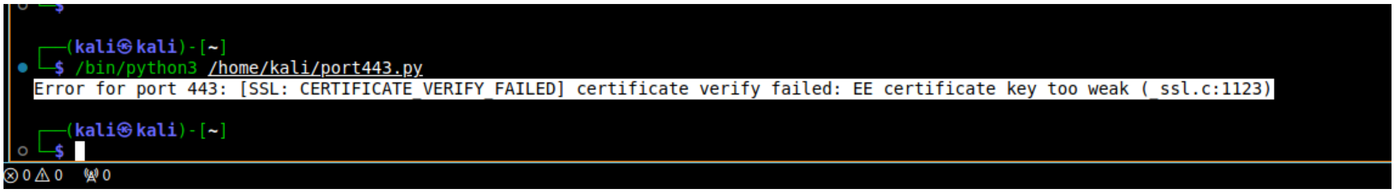
In the context of this program, what does sock.recv(1024) do, and why is it important for banner grabbing? (show in your results)

The line sock.recv(1024) reads up to 1024 bytes of data sent from the server once the connection is established. This data is what we call the "banner." It typically contains information such as: Service type ,Version number, Software details

- **Importance for Banner Grabbing**: **Data Collection**: This command is crucial because it collects the banner data sent by the server. Without this line, the program would connect to the port but not read the response, which is the actual data that reveals what service is running.

- **Limit to Read Size**: The number 1024 specifies the maximum amount of data to read. If the response from the server is larger, only the first 1024 bytes will be collected. This is typically sufficient to capture banner information.

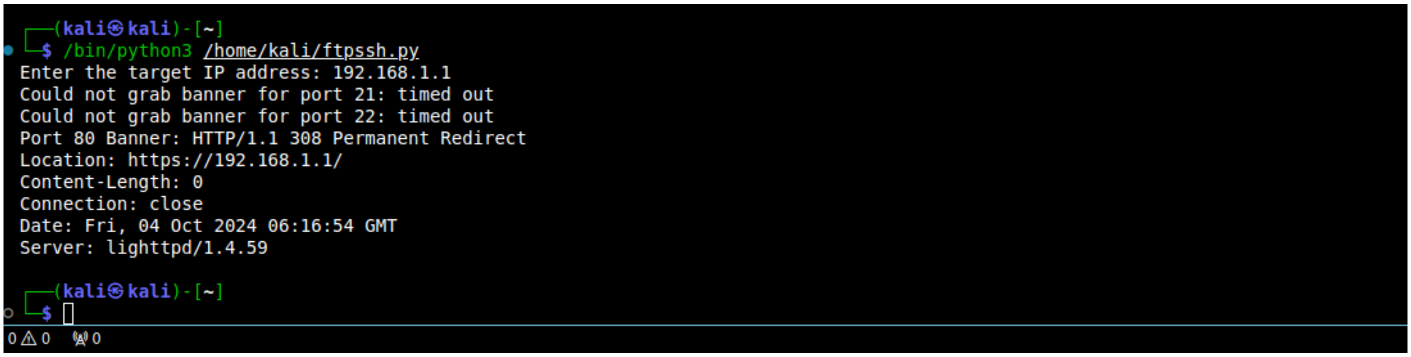
1. **What are some examples of banner information that you might retrieve from common services like HTTP, FTP, and SSH? (show in your results)**



```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/port443.py
Error for port 443: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: EE certificate key too weak (_ssl.c:1123)

┌──(kali㉿kali)-[~]
└─$ 
```

From the above image we see that the EE certificate key too weak, one of the way an attacker can exploit this weakness.



```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/ftpssh.py
Enter the target IP address: 192.168.1.1
Could not grab banner for port 21: timed out
Could not grab banner for port 22: timed out
Port 80 Banner: HTTP/1.1 308 Permanent Redirect
Location: https://192.168.1.1/
Content-Length: 0
Connection: close
Date: Fri, 04 Oct 2024 06:16:54 GMT
Server: lighttpd/1.4.59

┌──(kali㉿kali)-[~]
└─$ 
```

In this above banner grab we see the server information lighttpd/1.4.59
The older versions of lighttpd are vulnerable with remote access privileges and DOS attacks which are patched.