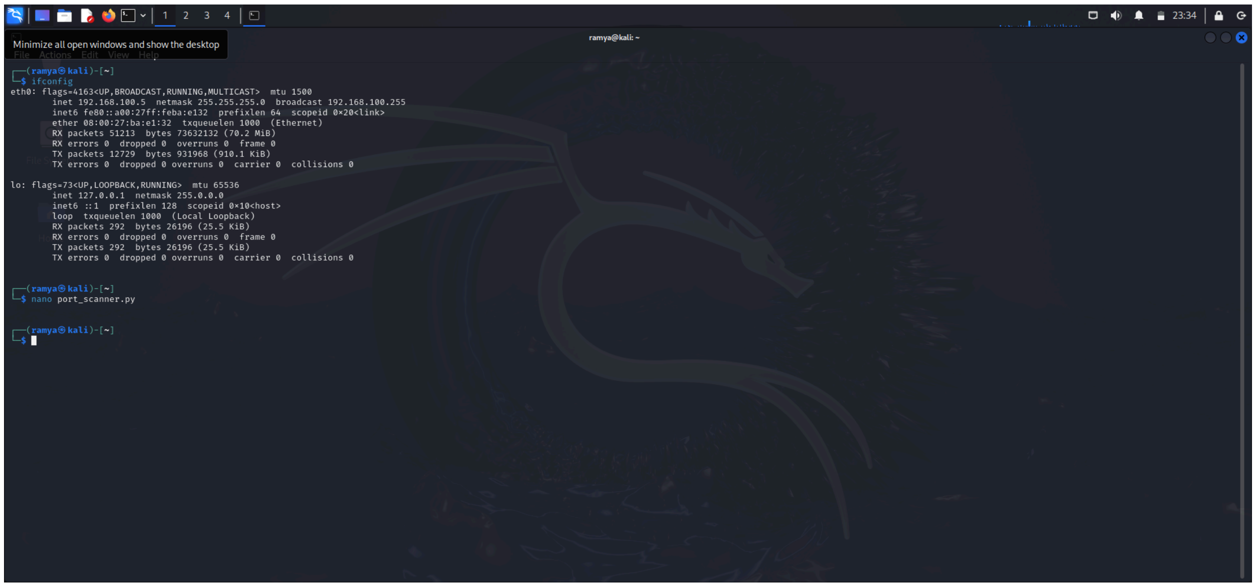
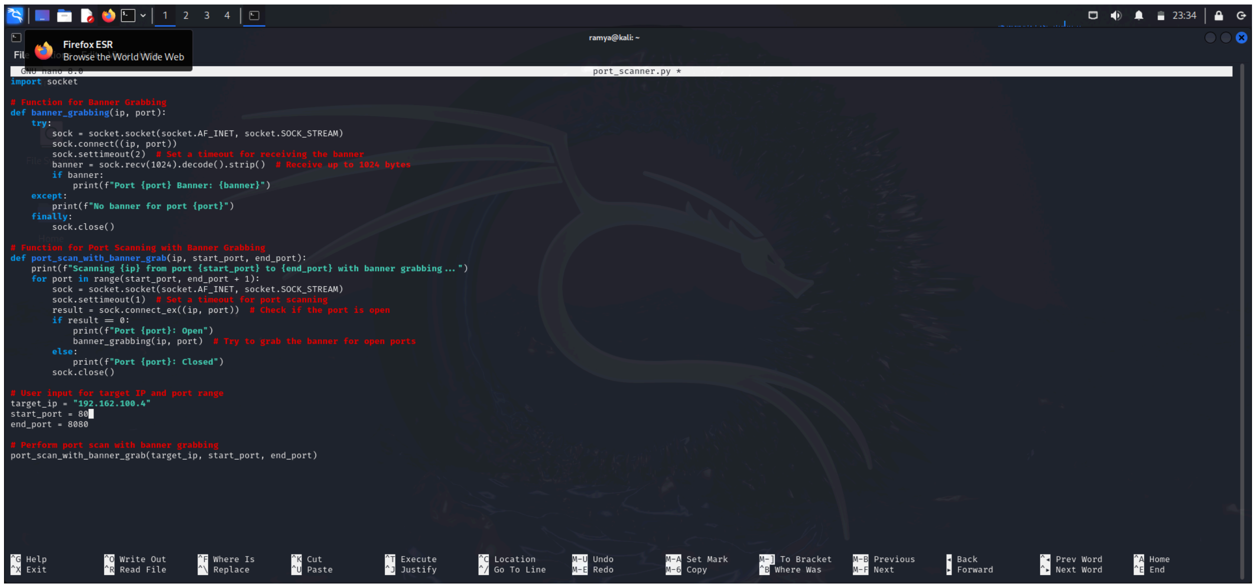
Banner Grabbing

a.) Determines Versions and Services: The vulnerability assessment procedure is aided by knowing which services are running on a host and their version numbers.
Gathers Configuration Details: By examining a service's configuration, vulnerabilities might be found.

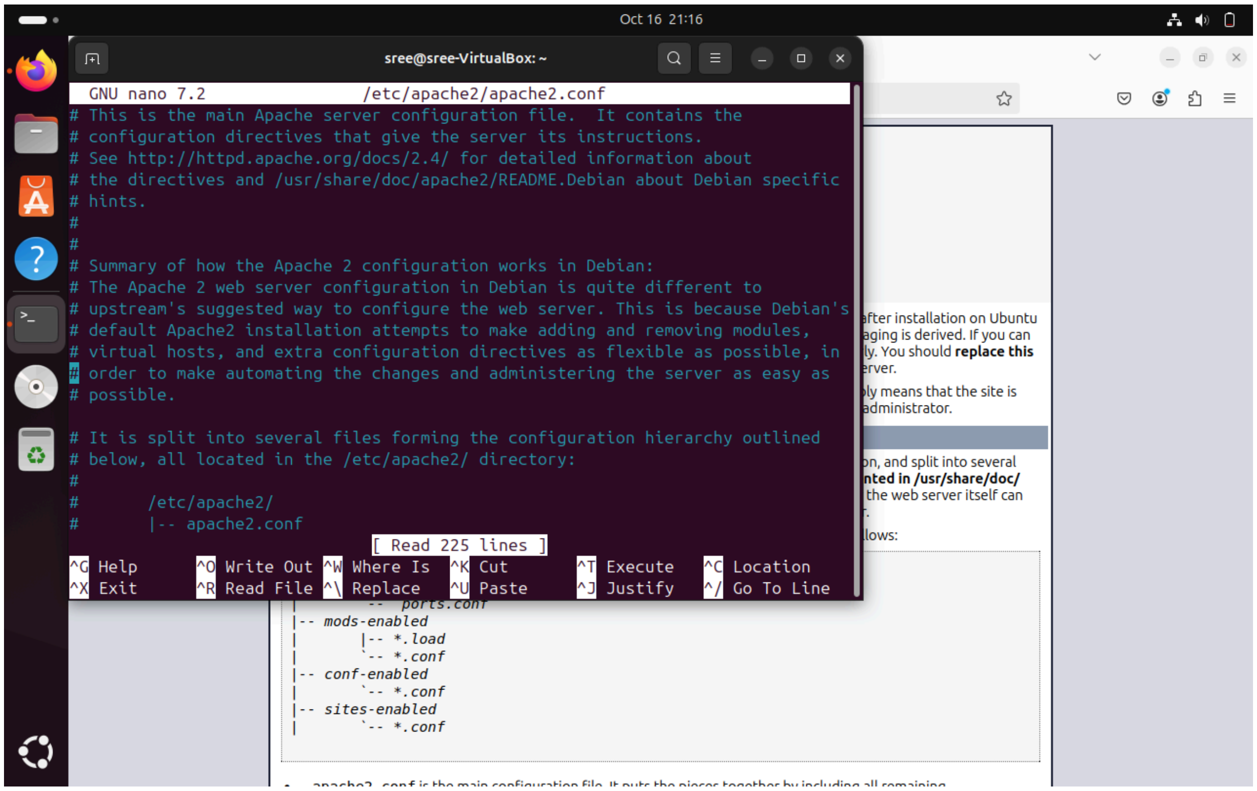
b.) Sock.Recv (1024) may read up to 1024 bytes from the other end of the connection. Because it captures the service's initial response (banner) once the connection is established, it is crucial for banner grabbing. This provides useful information regarding the service.
c.) Banner Information Examples:
• HTTP: "HTTP/1.1 200 OK" (which shows the version and status).
FTP: "220 Welcome to FTP service" (a welcoming message for the service).
• SSH: The version of SSH is indicated by "SSH-2.0-OpenSSH_7.4".


Minimize all open windows and show the desktop
File Actions Edit View Help
ramya@kali: ~
23:34

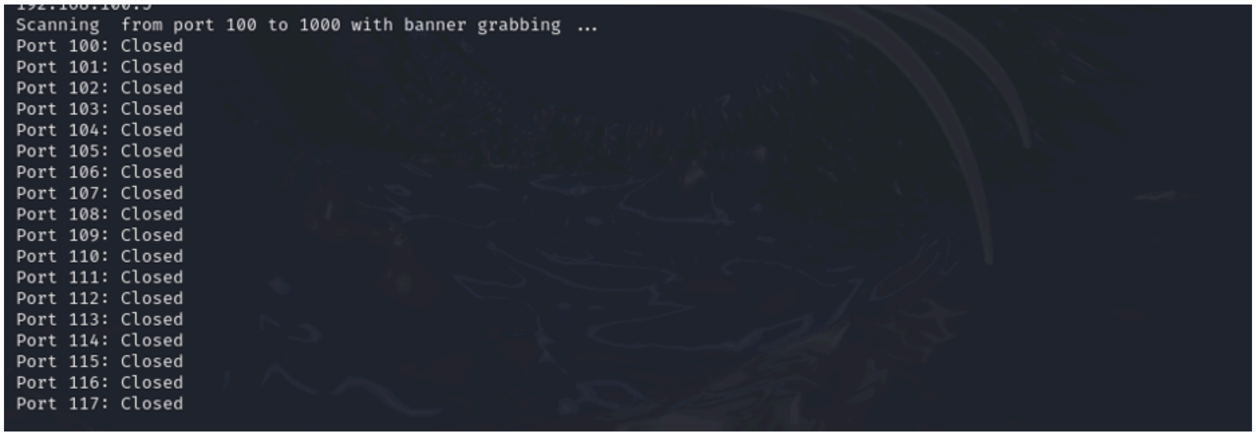
```
┌──(ramya㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.5  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:feba:e132  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:ba:e1:32  txqueuelen 1000  (Ethernet)
        RX packets 51213  bytes 73632132 (70.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12729  bytes 931968 (910.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 292  bytes 26196 (25.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 292  bytes 26196 (25.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(ramya㉿kali)-[~]
└─$ nano port_scanner.py


┌──(ramya㉿kali)-[~]
└─$ 
```



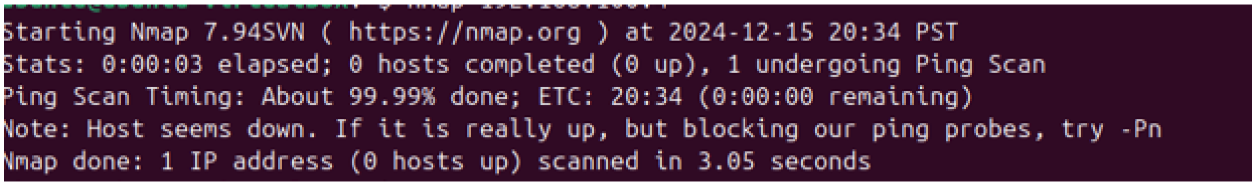
```
  GNU nano 8.0                         port_scanner.py *
import socket

# Function for Banner Grabbing
def banner_grabbing(ip, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect((ip, port))
        sock.settimeout(2)  # Set a timeout for receiving the banner
        banner = sock.recv(1024).decode().strip()  # Receive up to 1024 bytes
        if banner:
            print(f"Port {port} Banner: {banner}")
    except:
        print(f"No banner for port {port}")
    finally:
        sock.close()

# Function for Port Scanning with Banner Grabbing
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)  # Set a timeout for port scanning
        result = sock.connect_ex((ip, port))  # Check if the port is open
        if result == 0:
            print(f"Port {port}: Open")
            banner_grabbing(ip, port)  # Try to grab the banner for open ports
        else:
            print(f"Port {port}: Closed")
        sock.close()

# User input for target IP and port range
target_ip = "192.162.100.4"
start_port = 80
end_port = 8080

# Perform port scan with banner grabbing
port_scan_with_banner_grab(target_ip, start_port, end_port)
```



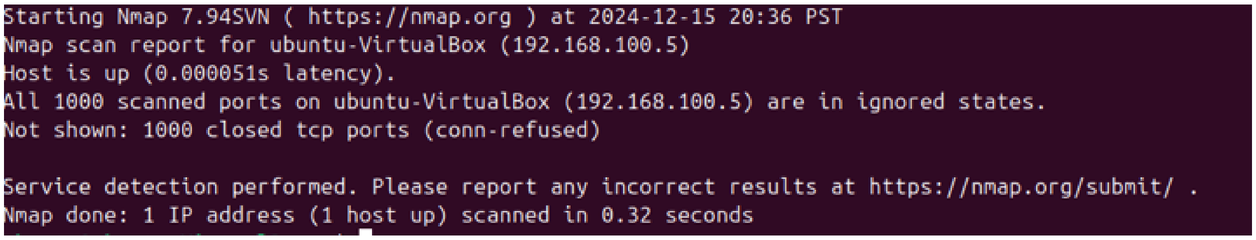
```
GNU nano 7.2                    /etc/apache2/apache2.conf
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
                          [ Read 225 lines ]
^G Help       ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit       ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```



```
Scanning  from port 100 to 1000 with banner grabbing ...
Port 100: Closed
Port 101: Closed
Port 102: Closed
Port 103: Closed
Port 104: Closed
Port 105: Closed
Port 106: Closed
Port 107: Closed
Port 108: Closed
Port 109: Closed
Port 110: Closed
Port 111: Closed
Port 112: Closed
Port 113: Closed
Port 114: Closed
Port 115: Closed
Port 116: Closed
Port 117: Closed
```



```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-15 20:34 PST
Stats: 0:00:03 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Ping Scan Timing: About 99.99% done; ETC: 20:34 (0:00:00 remaining)
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.05 seconds
```



```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-15 20:36 PST
Nmap scan report for ubuntu-VirtualBox (192.168.100.5)
Host is up (0.000051s latency).
All 1000 scanned ports on ubuntu-VirtualBox (192.168.100.5) are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```