**Project 1: Python**

Jessica, Dogan, Asiye & Micah

BeCode

2 June 2025

**Python Project: 1**

**Port Scanner – Design Report**

**1. Overview**

This Python-based tool scans a host for open TCP ports in a given range. It has been improved to:

* Log activity to a file
* Include rate limiting (to avoid abuse or detection)
* Validate user input
* Follow safer programming practices

**2. Why This Matters**

Port scanning helps identify vulnerable systems, but it's also commonly used by attackers. The paper *“Detection and Characterization of Port Scan Attacks”* (Lee et al., UCSD) shows:

* Most scans are vertical (many ports on one host) or horizontal (one port across many hosts)
* IDS systems detect scans if too many probes are made too quickly

So, we built a scanner that's safe, controlled, and respectful of these insights.

**3. Key Features**

* **User Input**: Enter host, start port, and end port.
* **Validation**: Ports must be between 1 and 65535.
* **Rate Limiting**: Adds a short delay between scans (0.3–0.5 seconds).
* **Random Scan Order**: Helps avoid being flagged by simple detection systems.
* **Logging**: Every result is logged.

**5. Security Tips**

* Only scan systems you own or have permission to scan.
* Keep delays between scans to avoid detection (and being blocked).
* Don’t use scanning tools irresponsibly—they can be illegal.

**6. Summary**

This port scanner is designed for safe use with built-in protection against fast scanning. The enhancements are based on real-world scan behavior research by Lee et al., which showed that overly aggressive scanning is easy to detect.

**Ethical Considerations Report**

The port scanner tool is educational and technically useful, however, its ethical implications must be examined closely to ensure responsible use within legal and professional boundaries.

Objective and Justification  
The primary purpose of this project was to enhance our understanding of network scanning techniques and TCP/IP communication protocols. This tool is meant for educational and ethical penetration testing only. It enables us to simulate common security assessments used by professionals to test system vulnerabilities in a legal and controlled environment.

Potential Ethical Risks  
Port scanning, even though technically simple, can have serious ethical consequences if misused:  
- Unauthorized scanning of systems may be interpreted as preparatory action for cyberattacks, such as intrusion or exploitation.  
- It may violate privacy laws and organizational policies, especially when done without the consent of the system owner.  
- Even if no damage is done, repeated scanning can be considered a form of harassment or can cause disruptions in network performance.

Mitigation Strategies and Legal Compliance  
To ensure that our project aligns with ethical standards:  
- All tests were conducted within our local machines or virtual environments, on systems explicitly designed for training.  
- We respected the principle of informed consent: no scan was executed on real-world systems without authorization.  
- The project complies with ethical hacking guidelines as defined by the EC-Council and OWASP frameworks.  
- The code includes warnings and validation prompts to discourage misuse and to raise awareness about ethical constraints.

Commitment to Responsible Use  
We, as future cybersecurity professionals, recognize that tools are neutral, but their usage defines their ethical value. It is our duty to ensure that tools developed during training are never used for malicious purposes. We advocate for transparency, responsibility, and respect of the law in all practical applications.

In conclusion, this project allowed us not only to acquire technical skills but also to reflect on the responsibilities linked to cybersecurity practices. Ethics must always guide technical knowledge, and our training, is grounded in the belief that security should protect, not harm.

**Testing and Issues Report**

When migrating and putting all the code together, in the validation string we had unicode emojis who caused syntax errors that did not allow the code to run properly. Validation and port scanning strings were correct, but due to the syntax problem, code needed to be modified, otherwise we would have to add unicode libraries that we have no knowledge about. To find about this issue we had to confirm with chatGPT even though the special characters were pretty obvious.  
  
From the beta code the main variable was removed due to the reason that it caused confusion.  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Traceback (most recent call last): File "C:\Users\Zstaigah\Desktop\VSCODE\VSCODE REPOS\python send\port\_scanner.py", line 73, in <module> if not validate\_ip(host): ~~~~~~~~~~~^^^^^^ File "C:\Users\Zstaigah\Desktop\VSCODE\VSCODE REPOS\python send\port\_scanner.py", line 15, in validate\_ip log(msg) ~~~^^^^^ File "C:\Users\Zstaigah\Desktop\VSCODE\VSCODE REPOS\python send\port\_scanner.py", line 8, in log f.write(message + "\n") ~~~~~~~^^^^^^^^^^^^^^^^ File "C:\Python313\Lib\encodings\cp1252.py", line 19, in encode return codecs.charmap\_encode(input,self.errors,encoding\_table)[0] ~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ UnicodeEncodeError: 'charmap' codec can't encode character '\u2713' in position 1: character maps to <undefined>

Also from the part where we tried to add the password section with the main variable priority so it would boot or proceed in order. We did not manage to provide the proper code to make it work this way thus we removed the main part of the code and we went in with cleaner version without parts. This helped us to reach the end result with no issues.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**if \_\_name\_\_ == "\_\_main\_\_":**

**host = input("Enter host (e.g., 127.0.0.1): ").strip()**

**try:**

**start = int(input("Enter start port: "))**

**end = int(input("Enter end port: "))**

**scan\_host(host, start, end)**

**except:**

**print("Please enter valid numbers for ports.")**

**log("User entered invalid port numbers.")**

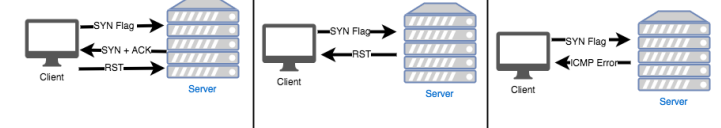
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How can detection of port scanning be achieved?**

Port scanning begins by first specifying the IP address of the target computer, which identifies the computer we want to examine. Once the target is defined, the next step is to set the range of ports that the scanner will probe. For instance, we might configure the scanner to check ports from 1 to 1060 like in this case.



Port scanner sends a SYN request to connect to a specific TCP or UDP port on a computer and asks their current status. This request is a fundamental part of the TCP (Transmission Control Protocol) handshake, or for UDP (User Datagram Protocol) ports, a specific UDP probe. There are three different type of results:



1- Open: If the port is open this means listening. It responds with a SYN-ACK (Synchronize-Acknowledgement) packet.

2- Closed : If the port is not available for the usage in that time this means closed. The computer responds with an RST (reset) packet indicating the connection is cancelled.

3- Timeout: If the port scanner doesn't receive a response, it indicates a timeout, which can mean the port is filtered or unreachable.

By analyzing these distinct responses, a port scanner can effectively map out the accessibility of ports on a target computer.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Principal Code (beta):**

**import socket**

**def log(message):**

**with open("simple\_port\_scanner.log", "a") as f:**

**f.write(message + "\n")**

**def scan\_port(host, port):**

**s = socket.socket()**

**s.settimeout(1)**

**try:**

**s.connect((host, port))**

**result = f"Port {port} is OPEN on {host}"**

**print(result)**

**log(result)**

**s.close()**

**return True**

**except:**

**return False**

**def scan\_host(host, start\_port, end\_port):**

**header = f"Scanning {host} from port {start\_port} to {end\_port}"**

**print(header)**

**log(header)**

**for port in range(start\_port, end\_port + 1):**

**scan\_port(host, port)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**host = input("Enter host (e.g., 127.0.0.1): ").strip()**

**try:**

**start = int(input("Enter start port: "))**

**end = int(input("Enter end port: "))**

**scan\_host(host, start, end)**

**except:**

**print("Please enter valid numbers for ports.")**

**log("User entered invalid port numbers.")**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**#----------------LOG FILE----------------**

**LOG\_FILE = "enhanced\_port\_scanner.log"**

**def log(message):**

**with open(LOG\_FILE, "a") as f:**

**f.write(message + "\n")**

**def validate\_ip(ip\_str):**

**try:**

**socket.inet\_aton(ip\_str)**

**msg = Fore.GREEN + f"[OK] Valid IP address: {ip\_str}"**

**print(msg)**

**log(msg)**

**return True**

**except socket.error:**

**msg = Fore.RED + f"[ERROR] Invalid IP address: {ip\_str}"**

**print(msg)**

**log(msg)**

**return False**

**#----------------VALIDATION----------------**

**def validate\_ports(start\_port, end\_port):**

**if not (1 <= start\_port <= 65535 and 1 <= end\_port <= 65535):**

**print(Fore.RED + "[ERROR] Ports must be between 1 and 65535.")**

**log(Fore.RED + "[ERROR] Invalid port range.")**

**sys.exit(1)**

**if start\_port > end\_port:**

**print(Fore.RED + "[ERROR] Start port must be less than or equal to end port.")**

**log(Fore.RED + "[ERROR] Start port is greater than end port.")**

**sys.exit(1)**

**msg = f"[OK] Valid port range: {start\_port}-{end\_port}"**

**print(msg)**

**log(msg)**

**Full Code - Final Version**import socket

import sys

# BANNER LIB

from colorama import Fore, Style, init

init(autoreset=True)

# BANNER CALL CODE PARAMETERS

colors = [Fore.RED, Fore.YELLOW, Fore.GREEN, Fore.CYAN, Fore.BLUE, Fore.MAGENTA]

def print\_rainbow\_banner():

banner\_lines = [

" ██▓███ ▒█████ ██▀███ ▄▄▄█████▓ ██████ ▄████▄ ▄▄▄ ███▄ █ ███▄ █ ▓█████ ██▀███ ▓█████ ███▄ █ ██░ ██ ▄▄▄ ███▄ █ ▄████▄ ▓█████ ▓█████▄ ",

" ▓██░ ██▒▒██▒ ██▒▓██ ▒ ██▒▓ ██▒ ▓▒ ▒██ ▒ ▒██▀ ▀█ ▒████▄ ██ ▀█ █ ██ ▀█ █ ▓█ ▀ ▓██ ▒ ██▒ ▓█ ▀ ██ ▀█ █ ▓██░ ██▒▒████▄ ██ ▀█ █ ▒██▀ ▀█ ▓█ ▀ ▒██▀ ██▌",

" ▓██░ ██▓▒▒██░ ██▒▓██ ░▄█ ▒▒ ▓██░ ▒░ ░ ▓██▄ ▒▓█ ▄ ▒██ ▀█▄ ▓██ ▀█ ██▒▓██ ▀█ ██▒▒███ ▓██ ░▄█ ▒ ▒███ ▓██ ▀█ ██▒▒██▀▀██░▒██ ▀█▄ ▓██ ▀█ ██▒▒▓█ ▄ ▒███ ░██ █▌",

" ▒██▄█▓▒ ▒▒██ ██░▒██▀▀█▄ ░ ▓██▓ ░ ▒ ██▒▒▓▓▄ ▄██▒░██▄▄▄▄██ ▓██▒ ▐▌██▒▓██▒ ▐▌██▒▒▓█ ▄ ▒██▀▀█▄ ▒▓█ ▄ ▓██▒ ▐▌██▒░▓█ ░██ ░██▄▄▄▄██ ▓██▒ ▐▌██▒▒▓▓▄ ▄██▒▒▓█ ▄ ░▓█▄ ▌",

" ▒██▒ ░ ░░ ████▓▒░░██▓ ▒██▒ ▒██▒ ░ ▒██████▒▒▒ ▓███▀ ░ ▓█ ▓██▒▒██░ ▓██░▒██░ ▓██░░▒████▒░██▓ ▒██▒ ░▒████▒▒██░ ▓██░░▓█▒░██▓ ▓█ ▓██▒▒██░ ▓██░▒ ▓███▀ ░░▒████▒░▒████▓ ",

" ▒▓▒░ ░ ░░ ▒░▒░▒░ ░ ▒▓ ░▒▓░ ▒ ░░ ▒ ▒▓▒ ▒ ░░ ░▒ ▒ ░ ▒▒ ▓▒█░░ ▒░ ▒ ▒ ░ ▒░ ▒ ▒ ░░ ▒░ ░░ ▒▓ ░▒▓░ ░░ ▒░ ░░ ▒░ ▒ ▒ ▒ ░░▒░▒ ▒▒ ▓▒█░░ ▒░ ▒ ▒ ░ ░▒ ▒ ░░░ ▒░ ░ ▒▒▓ ▒ ",

" ░▒ ░ ░ ▒ ▒░ ░▒ ░ ▒░ ░ ░ ░▒ ░ ░ ░ ▒ ▒ ▒▒ ░░ ░░ ░ ▒░░ ░░ ░ ▒░ ░ ░ ░ ░▒ ░ ▒░ ░ ░ ░░ ░░ ░ ▒░ ▒ ░▒░ ░ ▒ ▒▒ ░░ ░░ ░ ▒░ ░ ▒ ░ ░ ░ ░ ▒ ▒ ",

" ░░ ░ ░ ░ ▒ ░░ ░ ░ ░ ░ ░ ░ ░ ▒ ░ ░ ░ ░ ░ ░ ░ ░░ ░ ░ ░ ░ ░ ░ ░░ ░ ░ ▒ ░ ░ ░ ░ ░ ░ ░ ░ ",

" ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ░ ",

" ░ ░ ░"

]

for i, line in enumerate(banner\_lines):

print(colors[i % len(colors)] + Style.BRIGHT + line)

# ----------------BANNER CALL----------------

print\_rainbow\_banner()

#----------------PASSWORD PARAMETERS ----------------

SECRET\_PASSWORD = "mysecretpassword"

print(Fore.RED + f"--- Access Required ---")

entered\_password = input(Fore.CYAN + "Enter password: ").strip()

if entered\_password != SECRET\_PASSWORD:

print(Fore.RED + "[ACCESS DENIED] Incorrect password. Exiting.")

sys.exit(1)

else:

print(Fore.GREEN + "[ACCESS GRANTED] Password correct. Proceeding.")

#----------------LOG FILE----------------

LOG\_FILE = "enhanced\_port\_scanner.log"

def log(message):

with open(LOG\_FILE, "a") as f:

f.write(message + "\n")

#---------------- IP VALIDATION----------------

def validate\_ip(ip\_str):

try:

socket.inet\_aton(ip\_str)

msg = Fore.GREEN + f"[OK] Valid IP address: {ip\_str}"

print(msg)

log(msg)

return True

except socket.error:

msg = Fore.RED + f"[ERROR] Invalid IP address: {ip\_str}"

print(msg)

log(msg)

return False

#----------------PORT VALIDATION----------------

def validate\_ports(start\_port, end\_port):

if not (1 <= start\_port <= 65535 and 1 <= end\_port <= 65535):

print(Fore.RED + "[ERROR] Ports must be between 1 and 65535.")

log(Fore.RED + "[ERROR] Invalid port range.")

sys.exit(1)

if start\_port > end\_port:

print(Fore.RED + "[ERROR] Start port must be less than or equal to end port.")

log(Fore.RED + "[ERROR] Start port is greater than end port.")

sys.exit(1)

msg = f"[OK] Valid port range: {start\_port}-{end\_port}"

print(msg)

log(msg)

#----------------PORT SCANNING ----------------

def scan\_port(host, port, timeout=1):

s = socket.socket()

s.settimeout(timeout)

try:

s.connect((host, port))

msg = Fore.GREEN + f"[OPEN] Port {port} is open on {host}"

print(msg)

log(msg)

s.close()

return True

except socket.gaierror:

msg = Fore.RED + f"[ERROR] Failed to resolve host: {host}"

print(msg)

log(msg)

except socket.timeout:

msg = Fore.YELLOW + f"[TIMEOUT] Connection to port {port} timed out."

print(msg)

log(msg)

except socket.error as err:

msg = Fore.RED + f"[SOCKET ERROR] Port {port}: {err}"

print(msg)

log(msg)

except Exception as e:

msg = f"[EXCEPTION] Port {port}: {e}"

print(msg)

log(msg)

return False

def scan\_host(host, start\_port, end\_port):

header = Fore.GREEN + f"\n--- Scanning {host} from port {start\_port} to {end\_port} ---"

print(header)

log(header)

for port in range(start\_port, end\_port + 1):

scan\_port(host, port)

host = input(Fore.CYAN + f"Enter IP address to scan: ").strip()

if not validate\_ip(host):

sys.exit(1)

try:

start = int(input(Fore.CYAN + f"Enter start port: "))

end = int(input(Fore.CYAN + f"Enter end port: "))

validate\_ports(start, end)

scan\_host(host, start, end)

except ValueError:

msg = Fore.RED + f"[ERROR] Please enter valid numbers for ports."

print(msg)

log(msg)

sys.exit(1)

**Security Code Block**

**import sys**

**# --- Configuration ---**

**SECRET\_PASSWORD = "mysecretpassword" # This is the plain-text password!**

**# --- Simplest Password Check ---**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print("--- Access Required ---")**

**entered\_password = input("Enter password: ").strip()**

**# --- COMPARISON---**

**if entered\_password == SECRET\_PASSWORD:**

**print("[ACCESS GRANTED] Password correct. Proceeding.")**

**# Your port scanner's main code would go here**

**print(" (Port scanner logic starts here)")**

**else:**

**print("[ACCESS DENIED] Incorrect password. Exiting.")**

**sys.exit(1) # Stop the program immediately**

**#--- SEPARATION FROM MAIN CODE ---**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References**

EC-Council. (2021). *Certified Ethical Hacker (CEH) v11: Certified Ethical Hacker Study Guide*. EC-Council Press.

Grimes, R. A. (2019). *Cybersecurity Basics: A Hands-on Approach*. Apress.

Internet Assigned Numbers Authority. (n.d.). Service Name and Transport Protocol Port Number Registry. Retrieved from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Landoll, D. J. (2021). *The Security Risk Assessment Handbook: A Complete Guide for Performing Security Risk Assessments*. CRC Press.

Lee, C. B., Roedel, C., & Silenok, E. (2001). *Detection and characterization of port scan attacks*. Department of Computer Science & Engineering, University of California, San Diego.

OWASP Foundation. (2023). *OWASP Testing Guide v4*. https://owasp.org/www-project-web-security-testing-guide/

N. G. Pandey and R. S. Thakur, "PORT SCANNING AND ITS DETECTION IN PHYSICAL NETWORK," Dept. of CSE, DBACER, Nagpur.

Palo Alto Networks. (n.d.). *What is a port scan?* Cyberpedia. Retrieved from https://www.paloaltonetworks.com/cyberpedia/what-is-a-port-scan

S. Coyle, "Port Scanning: Techniques, Tools and Detection," University of Bradford, Bradford, United Kingdom.

Whitman, M. E., & Mattord, H. J. (2022). *Principles of Information Security (7th ed.)*. Cengage Learning.