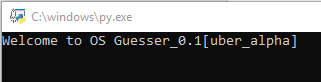
OS\_Guesser0.1[uber-alpha]

Technical and User Documentation

Author: Mario Pereira

**Contents**

[**Introduction** 1](#_Toc10195724)

[**Terminology, Abbreviations and Acronyms** 1](#_Toc10195725)

[**Executive Summary** 1](#_Toc10195726)

[**Technical Requirements** 1](#_Toc10195727)

[**Design and Code** 1](#_Toc10195728)

[**User Documentation** 1](#_Toc10195729)

[**Conclusions** 1](#_Toc10195730)

[**References** 1](#_Toc10195731)

[**Appendices** 1](#_Toc10195732)

## **Introduction**

The script this document pertains to - “OS\_Guesser0.1 [uber-alpha].py” was designed and coded by Mario Pereira, it was created originally to be a windows 7 host detector but eventually took a direction toward becoming a broader utility that guesses any host’s operating system the user decides to scan. With this document one should have the technical information necessary to properly operate the python script and modify it if necessary, in terms of copyright this script is fully open-source and free of proprietary ideals.

## **Terminology, Abbreviations and Acronyms**

1. Nmap

* Short for Network Mapper, this is a network security auditing tool.

1. Pseudocode

* Not real working code, this code serves the purpose of design communication between humans, by using appropriate English for replacing coding syntax.

1. IDE

* Stands for Integrate Development Environment, this is a software application that facilitates coding by providing the coder with source code editors, build automation, debuggers and other relevant coding features.

1. Docstring

* Simply put this is a multi-line string that is not assigned to any code but is specified beside coding modules or definitions to explain the process within said modules/definitions running inside the script file.

1. Readme

* This is a txt file included within the folder that holds the script. Readme files usually provide a summary of the program and its purpose, they also include special requirements, changes between versions and links to other documentation.

## **Executive Summary**

This Document serves as a guide for understanding the purpose, design and code of the current script version 0.1 of OS Guesser as well as understanding its functionality, how to run it, operational requirements and current limitations. The technical requirements detail dependencies and best practice for running the script and the User Documentation is provided in the form of video for understanding how to use it.

## **Technical Requirements**

1. Network

* The script scans the network subnet you are connected to, without a connection to a network the script will not function.
* For best performance it is best to ensure that host you are scanning have at least one port one and one closed on the network.

1. Firewalls

* Turning off Firewalls can vastly improve the performance of OS Guess.
* For best results turn off all firewalls on every host connected to the network.
* For security reasons remove the internet connection and isolate the SOEs to your internal network this is to prevent intrusions from the internet.

1. Python 3+ Interpreter

* Available for download from <https://www.python.org/downloads/release/python-373/>
* Python is an interpreted scripting language that is object-oriented, to run a file with the .py extension python is necessary.

1. Nmap

* This is a open-source tool for security auditing and network discovery. It offers a broad range of commands for testing a network or individual hosts connections, ports and more. It is required by the script to audit the network for hosts and then individually scan each one for a “finger print” which can then be used for identifying Operating Systems.

## **Design and Code**

1. Flow Chart

The Flowchart above lays out a simple format of the processes involved, and the path intended for the user to take.

1. Pseudocode

defining sub\_scanner function

subnet = user input (ask for subnet with slash notation)

print "beginning scan for connected hosts"

IP = addresses found by nmap process

file = create a file named "IP4Hosts" and write IP inside

for each line inside file:

DataVariable = format all data to one IP address per line

remove irrelevant characters from DataVariable

DataList = save DataVariable in form of list

CleanList = convert DataList to string

file2 = create a file named "hosts.txt" and write CleanList

TextForUserInFile = for each line in file 2 append ':' #this is so the code can differentiate positions when placing host info

create a file named "hostsprocessed.txt" and write TextForUserInFile

Return User to Main Menu

defining os\_scanner function

IPtable = open file "hosts.txt" and read as lines

index each line in IPtable

HostLineNo = user input (ask the user for integers corresponding to IPtable lines)

IPforNmap = in IPtable return line number integer(HostLineNo)

perform a scan for IPforNmap with Nmap and return results to "host\_temp(HostLineNo).txt" file

Prompt user for more scans or the main Menu

defining commitscan function

IPtable = open file "hosts.txt" and read as lines

index each line in IPtable

HostLineNo = user input (ask the user for integers corresponding to IPtable lines)

x =

y =

z =

w =

Read each line in "host\_temp(HostLineNo).txt":

if "running" in any line:

append that line to x

else if "Aggressive OS guesses:" in any line:

append that line to y

else if "OS CPE" in any line:

append that line to z

else "Warning" in any line:

append that line to w

open to write "hostsprocessed.txt" and append x+y+z+w to line number HostLineNo

Prompt user for more checks or the main menu

defining main\_menu function

print Welcome to OS Guesser 0.1 [uber\_alpha]

print Description: This script will collate OS specific details about hosts in a particular subnet.

print To begin - start with producing your host files by selecting SUB

Selection = prompt for user input with ([1/SUB] | [2/SCAN] | [3/CHECK] | {4/DETAILS] type number or word - )

if Selection = 1/SUB:

run subnet\_scanner function

else if Selection = 2/SCAN:

run os\_scanner function

else if Selection = 3/CHECK:

run commitscan function

else if Selection = 4/DETAILS:

print GUIDE:

print SUB - Scan your subnet first.

print SCAN - Secondly scan each host individually.

print CHECK - Thirdly creates Operating System Guess summary in hostprocessed.txt file for each

host (this is where we can compare hosts)

else:

print Invalid input

return the main\_menu function

1. IDE

* Jetbrain’s Pycharm Community Edition 2018.3.4

1. Docstrings

def subnet\_scanner():

"""subnet\_scanner is the first step to using this script tool, this function prompts the user for a subnet which it will

then scan for all connected IP addresses and write the data to a file(variable IP4host). This file is cleaned up as a

string and reprinted for future use as a reference point to host IP addresses across other defined functions, it also

also takes that clean file and adds a ":" to the end of each line and saves that as a new file name hostsprocessed.txt

this new file is the subnet host OS documentation file in which the 3/CHECK (commitscan function) will append OS guesses

and other OS related information pertaining to corresponding host IP Address."""

# asking user for subnet

# using cmd prompt in python to get nmap K>I>S>S(my rump) keep it simple stupid

# textfile produced by nmap below

# title

# recompiling txt into string with only IP Addresses

# turning into list to remove whitespaces

# turning back into string so later can output to file

# writing string with IPs to file named below

def os\_scanner():

"""os\_scanner is a function that will present hosts.txt(created by subnet\_scanner) and index each line for user input,

the user will be given a prompt asking for the index corresponding to the host they would like to scan, user can now

append that host OS on the main menu by using the 3/CHECK input"""

def commitscan():

"""commitscan is a function that is supposed to be run after the user has run a 2/SCAN (os\_scanner function) so that

the host scanned can be appended in a nice format to a txt file named "hostsprocessed.txt" this file can be used for

guessing network subnet hosts corresponding Operating Systems, this function can be edited easily to return other details

besides Operating System guesses."""

def main\_menu():

"""This is the main menu which serves as a guide to understanding and properly using this tool, it provides a step by step

view of the process as well as script 3/ DETAILS which include a guide on running the script. There are defined functions

within thin one such as CheckMore which are used to prompt the user to do repeated actions but always give the option to

also return to the main menu."""

1. Readme file

The following was extracted from the README.txt in the OS Guesser Script Folder.

OS GUESSER0.1[uber-alpha]

---Developed by Mario Pereira

PURPOSE

---The notion of an OS Guesser was conceived (and later designed) half way through coding/testing a different script which was intended to detect Windows 7

---operating systems in an internal network. This was then put aside as the thought and the appeal of a selective OS Guesser became stronger.

---

---The OS Guesser is designed for scanning a network with an installed Nmap package to take an educated guess as to what Operating Systems

---might be running on any host IP address that is in your network.

---

---For more details including flowchart and pseudocode please check OS Guess User and Technical Documentation available in the documents folder on GITHUB

REQUIREMENTS

---Python 3.7 or later

---Nmap 7.70 or later

---Internal Network connection

---other hosts to scan

---runs on any OS platform which has the above requirements.

GUIDE

---Main menu is presented like so ---> [1/SUB] | [2/SCAN] | [3/CHECK] | {4/DETAILS]

---

---If you type 1 or SUB - Scan your subnet first.

---YOU MUST RUN THE ABOVE STEP BEFORE PROCEEDING OR THE SCRIPT WILL CRASH

---

---If you type 2 or SCAN - Secondly scan each host individually.

---If you type 3 or CHECK - Thirdly create Operating System Guess summary in hostprocessed.txt file for each

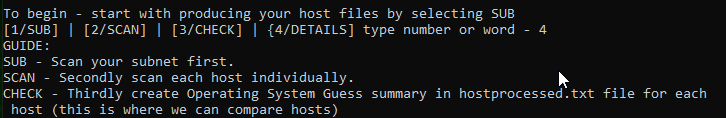
---host (this is where we can compare hosts)

---

---For a more in-depth User Guide refer to the User Documentation Video (provide link

## **User Documentation**

1. A Brief Guide is available inside the Readme file and while running the OS Guesser script from the 4/DETAILS function. This guide reads the following:

* Main menu is presented like so ---> [1/SUB] | [2/SCAN] | [3/CHECK] | {4/DETAILS]
* If you type 1 or SUB - Scan your subnet first.
* YOU MUST RUN THE ABOVE STEP BEFORE PROCEEDING OR THE SCRIPT WILL CRASH
* If you type 2 or SCAN - Secondly scan each host individually.
* If you type 3 or CHECK - Thirdly create Operating System Guess summary in hostprocessed.txt file for each host (this is where we can compare hosts)
* For a more in-depth User Guide refer to the User Documentation Video (<https://youtu.be/B4TzRxVQRdE>)

The image above demonstrates the 4/DETAILS function in the script.

1. The official documentation for this script is in the form of a video with the full utility of the OS Guesser in action <https://youtu.be/B4TzRxVQRdE> <--If this link does not work I have provided the original video in the Folder.

## **Conclusions**

The scripting program OS Guesser documented here provides the user with the ability to perform guesses from within any IPv4 network. The script does require PC configuration in the form of disabled firewalls/network security to enable accurate guessing. The Documentation including pseudo coding, flow charts, design and code are all available from the GITHUB repository <https://github.com/Monsid/OS-Guesser> with a Video User Guide available on [Youtube](https://youtu.be/B4TzRxVQRdE) .

## **References**

1. Software used to Design and Produce Script

* Creating Flowcharts - www.draw.io
* Writing Pseudocode – Notepad
* IDE for writing Code – Jetbrain’s Pycharm Community Edition 2018.3.4

1. Software used to test script.

* Python 3.7 from console.
* Jetbrain’s Pycharm in debug mode.
* Google forms – peer review testing.

1. Software used to produce Documentation.

* Microsoft Word
* Notepad
* Radeon ReLive – video capture
* DaVinci Resolve by Black Magic Design – video editing <https://www.blackmagicdesign.com/products/davinciresolve/>

## **Appendices**

1. Distribution – Github

* <https://github.com/Monsid/OS-Guesser>

1. Testing - Peer Review, see next page.

* <https://forms.gle/ncoEfUycu6knXAeJ7> currently ongoing, feel free to include a response.

