



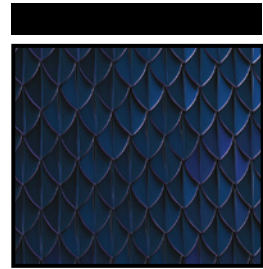
# PYTHON<sup>®</sup> FOR CYBERSECURITY

USING PYTHON FOR  
CYBER OFFENSE AND DEFENSE

Howard E. Poston III

WILEY





# Python<sup>®</sup> for Cybersecurity

Using Python for Cyber Offense  
and Defense

Howard E. Poston III

WILEY

Copyright © 2022 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.  
Published simultaneously in Canada.

978-1-119-85064-9  
978-1-119-85070-0 (ebk.)  
978-1-119-85065-6 (ebk.)

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware the Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at [www.wiley.com](http://www.wiley.com).

**Library of Congress Control Number:** 2021951037

**Trademarks:** WILEY and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Python is a registered trademark of Python Software Foundation. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Cover image: © Alexander/Adobe Stock

Cover design: Wiley/Michael E. Trent

*To Rachel*



## About the Author

**Howard E. Poston III** is a freelance consultant and content creator with a focus on blockchain and cybersecurity. He has developed and taught more than a dozen courses exploring and explaining various aspects of cybersecurity and has written hundreds of articles on the subject on different outlets. Howard Poston is also the author of several academic articles on security topics, and has spoken on blockchain and cybersecurity at international security conferences.



## Acknowledgments

Thanks to my technical editor, Ben Heruska, and the amazing team at Wiley without whom this book would not have been possible.



## About the Technical Editor

**Benjamin Heruska** is a military officer and computer engineer in the United States Air Force, which he joined in 2008. He has diverse military engineering experience across a broad range of computing disciplines, including embedded RF systems development, IT and cybersecurity tool development, software development, vulnerability analysis, cybersecurity incident response, big data engineering and analytics, ICAM development, and technical leadership.







# Contents at a Glance

<b>Introduction</b>	<b>xvii</b>
<b>Chapter 1 Fulfilling Pre-ATT&amp;CK Objectives</b>	<b>1</b>
<b>Chapter 2 Gaining Initial Access</b>	<b>19</b>
<b>Chapter 3 Achieving Code Execution</b>	<b>39</b>
<b>Chapter 4 Maintaining Persistence</b>	<b>55</b>
<b>Chapter 5 Performing Privilege Escalation</b>	<b>77</b>
<b>Chapter 6 Evading Defenses</b>	<b>89</b>
<b>Chapter 7 Accessing Credentials</b>	<b>105</b>
<b>Chapter 8 Performing Discovery</b>	<b>125</b>
<b>Chapter 9 Moving Laterally</b>	<b>141</b>
<b>Chapter 10 Collecting Intelligence</b>	<b>157</b>
<b>Chapter 11 Implementing Command and Control</b>	<b>169</b>
<b>Chapter 12 Exfiltrating Data</b>	<b>183</b>
<b>Chapter 13 Achieving Impact</b>	<b>199</b>
<b>Index</b>	<b>213</b>





# Contents

<b>Introduction</b>	<b>xvii</b>
<b>Chapter 1 Fulfilling Pre-ATT&amp;CK Objectives</b>	<b>1</b>
Active Scanning	2
Scanning Networks with scapy	2
Implementing a SYN Scan in scapy	4
Performing a DNS Scan in scapy	5
Running the Code	5
Network Scanning for Defenders	6
Monitoring Traffic with scapy	7
Building Deceptive Responses	8
Running the Code	9
Search Open Technical Databases	9
Offensive DNS Exploration	10
Searching DNS Records	11
Performing a DNS Lookup	12
Reverse DNS Lookup	12
Running the Code	13
DNS Exploration for Defenders	13
Handling DNS Requests	15
Building a DNS Response	15
Running the Code	16
Summary	17
Suggested Exercises	17
<b>Chapter 2 Gaining Initial Access</b>	<b>19</b>
Valid Accounts	20
Discovering Default Accounts	20
Accessing a List of Default Credentials	21
Starting SSH Connections in Python	22

Performing Telnet Queries in Python	23
Running the Code	24
Account Monitoring for Defenders	24
Introduction to Windows Event Logs	25
Accessing Event Logs in Python	28
Detecting Failed Logon Attempts	28
Identifying Unauthorized Access to Default Accounts	30
Running the Code	30
Replication Through Removable Media	31
Exploiting Autorun	31
Converting Python Scripts to Windows Executables	32
Generating an Autorun File	33
Setting Up the Removable Media	34
Running the Code	34
Detecting Autorun Scripts	34
Identifying Removable Drives	35
Finding Autorun Scripts	36
Detecting Autorun Processes	36
Running the Code	36
Summary	37
Suggested Exercises	37
<b>Chapter 3 Achieving Code Execution</b>	<b>39</b>
Windows Management Instrumentation	40
Executing Code with WMI	40
Creating Processes with WMI	41
Launching Processes with PowerShell	41
Running the Code	42
WMI Event Monitoring for Defenders	42
WMI in Windows Event Logs	43
Accessing WMI Event Logs in Python	45
Processing Event Log XML Data	45
Running the Code	46
Scheduled Task/Job	47
Scheduling Malicious Tasks	47
Checking for Scheduled Tasks	48
Scheduling a Malicious Task	48
Running the Code	49
Task Scheduling for Defenders	50
Querying Scheduled Tasks	51
Identifying Suspicious Tasks	52
Running the Code	52
Summary	53
Suggested Exercises	53

<b>Chapter 4</b>	<b>Maintaining Persistence</b>	<b>55</b>
	Boot or Logon Autostart Execution	56
	Exploiting Registry Autorun	56
	The Windows Registry and Autorun Keys	57
	Modifying Autorun Keys with Python	60
	Running the Code	61
	Registry Monitoring for Defenders	62
	Querying Windows Registry Keys	63
	Searching the HKU Hive	64
	Running the Code	64
	Hijack Execution Flow	65
	Modifying the Windows Path	65
	Accessing the Windows Path	66
	Modifying the Path	67
	Running the Code	68
	Path Management for Defenders	69
	Detecting Path Modification via Timestamps	69
	Enabling Audit Events	71
	Monitoring Audit Logs	73
	Running the Code	75
	Summary	76
	Suggested Exercises	76
 <b>Chapter 5</b>	 <b>Performing Privilege Escalation</b>	 <b>77</b>
	Boot or Logon Initialization Scripts	78
	Creating Malicious Logon Scripts	78
	Achieving Privilege Escalation with Logon Scripts	79
	Creating a Logon Script	79
	Running the Code	79
	Searching for Logon Scripts	80
	Identifying Autorun Keys	81
	Running the Code	81
	Hijack Execution Flow	81
	Injecting Malicious Python Libraries	82
	How Python Finds Libraries	82
	Creating a Python Library	83
	Running the Code	83
	Detecting Suspicious Python Libraries	83
	Identifying Imports	85
	Detecting Duplicates	85
	Running the Code	86
	Summary	86
	Suggested Exercises	87

<b>Chapter 6</b>	<b>Evading Defenses</b>	<b>89</b>
	Impair Defenses	90
	Disabling Antivirus	90
	Disabling Antivirus Autorun	90
	Terminating Processes	93
	Creating Decoy Antivirus Processes	94
	Catching Signals	95
	Running the Code	95
	Hide Artifacts	95
	Concealing Files in Alternate Data Streams	96
	Exploring Alternate Data Streams	96
	Alternate Data Streams in Python	97
	Running the Code	98
	Detecting Alternate Data Streams	98
	Walking a Directory with Python	99
	Using PowerShell to Detect ADS	100
	Parsing PowerShell Output	101
	Running the Code	102
	Summary	102
	Suggested Exercises	103
<b>Chapter 7</b>	<b>Accessing Credentials</b>	<b>105</b>
	Credentials from Password Stores	106
	Dumping Credentials from Web Browsers	106
	Accessing the Chrome Master Key	108
	Querying the Chrome Login Data Database	108
	Parsing Output and Decrypting Passwords	109
	Running the Code	109
	Monitoring Chrome Passwords	110
	Enabling File Auditing	110
	Detecting Local State Access Attempts	111
	Running the Code	113
	Network Sniffing	114
	Sniffing Passwords with scapy	114
	Port-Based Protocol Identification	116
	Sniffing FTP Passwords	116
	Extracting SMTP Passwords	117
	Tracking Telnet Authentication State	119
	Running the Code	121
	Creating Deceptive Network Connections	121
	Creating Decoy Connections	122
	Running the Code	122
	Summary	123
	Suggested Exercises	123

<b>Chapter 8</b>	<b>Performing Discovery</b>	<b>125</b>
	Account Discovery	126
	Collecting User Account Data	126
	Identifying Administrator Accounts	127
	Collecting User Account Information	128
	Accessing Windows Password Policies	128
	Running the Code	129
	Monitoring User Accounts	130
	Monitoring Last Login Times	130
	Monitoring Administrator Login Attempts	131
	Running the Code	132
	File and Directory Discovery	133
	Identifying Valuable Files and Folders	133
	Regular Expressions for Data Discovery	135
	Parsing Different File Formats	135
	Running the Code	136
	Creating Honeypot Files and Folders	136
	Monitoring Decoy Content	136
	Creating the Decoy Content	137
	Running the Code	138
	Summary	138
	Suggested Exercises	139
<b>Chapter 9</b>	<b>Moving Laterally</b>	<b>141</b>
	Remote Services	142
	Exploiting Windows Admin Shares	142
	Enabling Full Access to Administrative Shares	143
	Transferring Files via Administrative Shares	144
	Executing Commands on Administrative Shares	144
	Running the Code	144
	Admin Share Management for Defenders	145
	Monitoring File Operations	146
	Detecting Authentication Attempts	147
	Running the Code	148
	Use Alternative Authentication Material	148
	Collecting Web Session Cookies	149
	Accessing Web Session Cookies	150
	Running the Code	150
	Creating Deceptive Web Session Cookies	151
	Creating Decoy Cookies	151
	Monitoring Decoy Cookie Usage	153
	Running the Code	153
	Summary	154
	Suggested Exercises	155

<b>Chapter 10</b>	<b>Collecting Intelligence</b>	<b>157</b>
	Clipboard Data	158
	Collecting Data from the Clipboard	158
	Accessing the Windows Clipboard	159
	Replacing Clipboard Data	159
	Running the Code	160
	Clipboard Management for Defenders	160
	Monitoring the Clipboard	161
	Processing Clipboard Messages	161
	Identifying the Clipboard Owner	161
	Running the Code	162
	Email Collection	162
	Collecting Local Email Data	162
	Accessing Local Email Caches	163
	Running the Code	163
	Protecting Against Email Collection	164
	Identifying Email Caches	165
	Searching Archive Files	165
	Running the Code	166
	Summary	166
	Suggested Exercises	166
<b>Chapter 11</b>	<b>Implementing Command and Control</b>	<b>169</b>
	Encrypted Channel	170
	Command and Control Over Encrypted Channels	170
	Encrypted Channel Client	171
	Encrypted Channel Server	172
	Running the Code	173
	Detecting Encrypted C2 Channels	174
	Performing Entropy Calculations	175
	Detecting Encrypted Traffic	175
	Running the Code	176
	Protocol Tunneling	176
	Command and Control via Protocol Tunneling	176
	Protocol Tunneling Client	177
	Protocol Tunneling Server	177
	Running the Code	179
	Detecting Protocol Tunneling	179
	Extracting Field Data	181
	Identifying Encoded Data	181
	Running the Code	181
	Summary	182
	Suggested Exercises	182
<b>Chapter 12</b>	<b>Exfiltrating Data</b>	<b>183</b>
	Alternative Protocols	184
	Data Exfiltration Over Alternative Protocols	184
	Alternative Protocol Client	185



Alternative Protocol Server	186
Running the Code	188
Detecting Alternative Protocols	189
Detecting Embedded Data	190
Running the Code	191
Non-Application Layer Protocols	191
Data Exfiltration via Non-Application Layer Protocols	192
Non-Application Layer Client	193
Non-Application Layer Server	193
Running the Code	194
Detecting Non-Application Layer Exfiltration	195
Identifying Anomalous Type and Code Values	196
Running the Code	196
Summary	197
Suggested Exercises	197
<b>Chapter 13 Achieving Impact</b>	<b>199</b>
Data Encrypted for Impact	200
Encrypting Data for Impact	200
Identifying Files to Encrypt	201
Encrypting and Decrypting Files	202
Running the Code	202
Detecting File Encryption	203
Finding Files of Interest	204
Calculating File Entropies	204
Running the Code	205
Account Access Removal	205
Removing Access to User Accounts	205
Changing Windows Passwords	207
Changing Linux Passwords	207
Running the Code	207
Detecting Account Access Removal	208
Detecting Password Changes in Windows	209
Detecting Password Changes in Linux	210
Running the Code	211
Summary	211
Suggested Exercises	212
<b>Index</b>	<b>213</b>





# Introduction

This book is all about how to use Python for cybersecurity. Before we dive into that, let's take a moment to talk about the "why" of Python for cybersecurity.

A good starting point is answering the question "Why use automation?" If you're already in the cybersecurity field, you probably know that automation is your friend.

If you're just entering the field, consider how hard it is to keep one of your less tech-savvy relatives or friends from installing malware on their phone or falling for a phishing email. Now, scale that up to hundreds or thousands of people. Add in the fact that attackers are actually motivated to target your organization, and a single successful attack could cost the company millions of dollars. Managing cyber risk includes preventing malware infections, detecting and remediating ongoing attacks, ensuring compliance with corporate security policies, and more. By helping to handle some of this for you, automation is your friend.

So, given that automation is necessary in cybersecurity, why use Python? Python has a few features that make it a good choice, including the following:

- **It's popular:** There's a decent chance that you already know some Python. It's a lot easier to learn new ways to use a language that you know than to learn a new language from scratch. In 2021, Python was the second most popular language on the TIOBE index (<https://www.tiobe.com/tiobe-index/>) and was quickly overtaking C.
- **It's easy:** For those of you who don't know Python, it's pretty quick and easy to pick up. This is helpful for both learning and dashing out a program quickly.

- **It's powerful:** Python has many powerful libraries that can be easily imported into your code. If you want to do anything with network traffic, it's a lot easier to use `scapy` than to try to do it from scratch.

## How This Book Is Organized

---

This book is organized based on the MITRE ATT&CK framework. The MITRE ATT&CK framework is a tool produced by the MITRE Corporation to build understanding of how a cyberattack works. It takes the lifecycle of a cyberattack and breaks it into objectives that the attacker may need to achieve on the way to their final goal. For each of these objectives, MITRE ATT&CK describes various ways in which they can be accomplished.

## Tactics and Techniques

The MITRE ATT&CK framework is organized as a hierarchy. At the top level of this hierarchy are the MITRE *tactics*, which describe the goals that an attacker may want to achieve during a cyberattack. These tactics include the following:

- Reconnaissance
- Resource Development
- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact

For each of these tactics, MITRE ATT&CK outlines several techniques and subtechniques that describe specific methods of achieving these goals. For example, an attacker could use Brute Force (<https://attack.mitre.org/tactics/TA0006/>) or Network Sniffing (<https://attack.mitre.org/techniques/T1110/>) to achieve Credential Access (<https://attack.mitre.org/>

techniques/T1040/). Each of these techniques and subtechniques has its own page describing how the attack is performed, how it can be detected, and more.

This book is structured around the MITRE ATT&CK framework. Each tactic will have its own chapter (except for the first two, which are combined into MITRE Pre-ATT&CK).

Each of these chapters explores two of the techniques from its tactic and how they can be implemented in Python. Each of these offensive sections will be paired with a defensive section demonstrating how Python can also be used to defeat these attack vectors.

## Why MITRE ATT&CK?

The goal of this book is to demonstrate how Python can be used to address cybersecurity use cases. To that end, it is helpful to have a clear framework that outlines different offensive and defensive cybersecurity tasks.

MITRE ATT&CK provides that framework with its hierarchy of tactics and techniques that describe the various objectives of a cyberattack and how to achieve them. This book draws offensive techniques from each of the MITRE ATT&CK tactics and demonstrates how they and defensive countermeasures can be implemented using Python.

Beyond this structure, MITRE ATT&CK is also useful because it provides a wealth of additional resources and room to grow. Each technique includes in-depth information about how the attack works and how to defend against it. MITRE ATT&CK also describes hundreds of techniques not covered in this book, providing numerous opportunities to apply Python to new use cases.

## Tools You Will Need

---

This book is designed to demonstrate how to use Python to solve various use cases. If you don't have Python open and aren't running the code, then you're doing it wrong.

## Setting Up Python

The code samples included with this book were written for version 3.9 of Python. If you are using an earlier version of Python or, if by the time you are reading this, Python has advanced so far as to break backwards compatibility, then the code samples may not work for you.

To download the latest version of Python, we recommend visiting <https://www.python.org/downloads/>. From there, you can download and install the appropriate version for your system. Also, install `pip` and ensure that Python 3 is the default Python on the system by removing Python 2.X, installing a package like `python-is-python3`, or creating an alias for the `python` and `pip` commands.

Most of the sample code included in this book will run on either Windows or \*nix systems. However, some examples do include platform-specific functionality, such as access to Windows log files. In these cases, we recommend using a virtual machine, such as VirtualBox (<https://www.virtualbox.org/wiki/Downloads>) or VMware Workstation (<https://www.vmware.com/products/workstation-player.html>), if you don't own a computer with the necessary OS.

## Accessing Code Samples

Each chapter of this book will include at least four Python code files. Depending on the exercise, additional code or files may be included as well.

These code samples are available at <https://www.wiley.com/go/pythonforcybersecurity> on the Download Code tab. The code samples are available in ZIP files labeled with the chapter number. Before beginning a chapter, download the appropriate file and extract its contents.

These code samples may be updated over time to maintain compatibility with current Python versions and libraries and operating system internals (such as how Windows organizes its Registry and Event logs). If this occurs, the downloadable code samples may not exactly match the sample code in the text.

## Installing Packages

One of the main benefits of Python for cybersecurity is the wide range of libraries that it provides. Many of the code samples included with this book require packages that are not shipped as part of the core Python distribution.

From the Download Code tab at <https://www.wiley.com/go/pythonforcybersecurity>, download the ZIP file for this chapter. This includes a file named `requirements.txt`, which lists the Python libraries that are used within this book.

To install these packages, run the command `python -m pip install -r requirements.txt` in the directory where you have saved this file. If the command completes successfully, then all required packages will be downloaded and installed on your computer.

## From Here

---

Python is a popular, easy-to-use, and powerful programming language, making it an ideal choice for cybersecurity automation. This book demonstrates how Python can be applied to various offensive and defensive cybersecurity use cases from the MITRE ATT&CK framework.

This book is designed to be interactive with code samples included for each chapter. Before moving on to the next chapter, be sure to install Python and the required Python libraries on your computer.

# Fulfilling Pre-ATT&CK Objectives

Originally, MITRE Pre-ATT&CK was a stand-alone matrix within the MITRE ATT&CK framework. It detailed the various steps that an attacker could take to prepare before attempting to gain initial access to a target environment.

In October 2020, MITRE restructured the ATT&CK framework and condensed MITRE Pre-ATT&CK into two *tactics* of the ATT&CK matrix. The new version breaks Pre-ATT&CK into Reconnaissance and Resource Development, as shown in Figure 1.1.

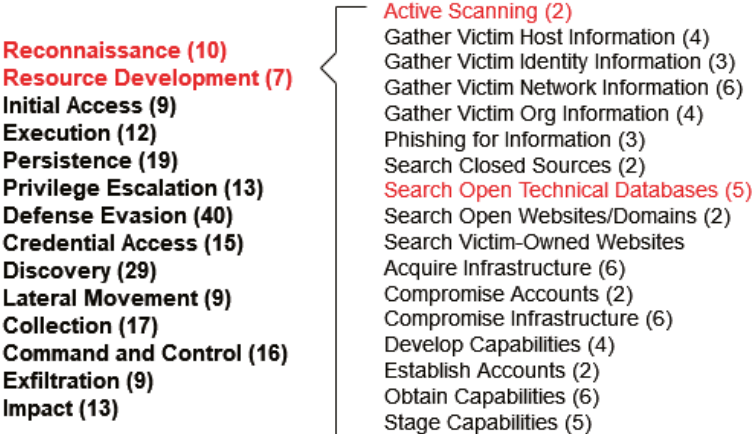


Figure 1.1: MITRE Pre-ATT&CK

In this chapter, we will focus on the Reconnaissance tactic of MITRE Pre-ATT&CK. The reason is that while Resource Development can be automated, the details can vary greatly, and this stage of the attack is not visible to the defender. For example, Python could be used for implementing a domain generation algorithm (DGA) for phishing or automating the deployment of web-based services, but these apply only in certain types of attacks and can easily be implemented in other ways.

Reconnaissance, on the other hand, can benefit significantly from automation. Also, Python includes several packages that help with automating reconnaissance, such as `scapy` and various DNS libraries.

The MITRE Pre-ATT&CK framework includes 10 techniques for Reconnaissance. Here, we will explore the use of Python for the Active Scanning and Search Open Technical Databases techniques.

The code sample archive for this chapter can be found on the Download Code tab at <https://www.wiley.com/go/pythonforcybersecurity> and contains the following sample code files:

- `PortScan.py`
- `HoneyScan.py`
- `DNSExploration.py`
- `HoneyResolver.py`

## Active Scanning

---

Network reconnaissance can be performed by either active or passive means. Active reconnaissance involves interacting with the target environment, while passive reconnaissance can involve eavesdropping on traffic or taking advantage of publicly available sources of information.

As its name suggests, the Active Scanning technique in MITRE ATT&CK is an example of Active Reconnaissance. It involves performing port or vulnerability scans against a target to determine which IP addresses are active, what services they are running, any vulnerabilities that may exist, and similar intelligence.

### Scanning Networks with `scapy`

Nmap is the most used tool for port scanning. It implements several different types of scans and can be used to detect the versions of operating systems and services and to perform custom vulnerability scans.



In this section, we'll implement a couple of simple scans:

- **SYN scan:** A SYN scan sends a TCP SYN packet to a port and looks for a SYN/ACK packet in response.
- **DNS scan:** A DNS scan tests to see whether a DNS server is running on the target system.

To implement these scans, we'll be using the `scapy` library in Python. `scapy` makes it easy to create and send custom packets over the network and to sniff network traffic for responses.

### PortScan.py

```
from scapy.all import *
import ipaddress

ports = [25,80,53,443,445,8080,8443]

def SynScan(host):
    ans,unans = sr(
        IP(dst=host)/
        TCP(sport=33333,dport=ports,flags="S")
        ,timeout=2,verbose=0)
    print("Open ports at %s:" % host)
    for (s,r,) in ans:
        if s[TCP].dport == r[TCP].sport and r[TCP].flags=="SA":
            print(s[TCP].dport)

def DNSScan(host):
    ans,unans = sr(
        IP(dst=host)/
        UDP(dport=53)/
        DNS(rd=1,qd=DNSQR(qname="google.com"))
        ,timeout=2,verbose=0)
    if ans and ans[UDP]:
        print("DNS Server at %s"%host)

host = input("Enter IP Address: ")
try:
    ipaddress.ip_address(host)
except:
    print("Invalid address")
    exit(-1)

SynScan(host)
DNSScan(host)
```