PYTHON® FOR CYBERSECURITY

USING PYTHON FOR CYBER OFFENSE AND DEFENSE

Howard E. Poston III

WILEY

Python® for Cybersecurity

Using Python for Cyber Offense and Defense

Howard E. Poston III

WILEY

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.)

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.

Library of Congress Control Number: 2021951037

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

Contents

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.

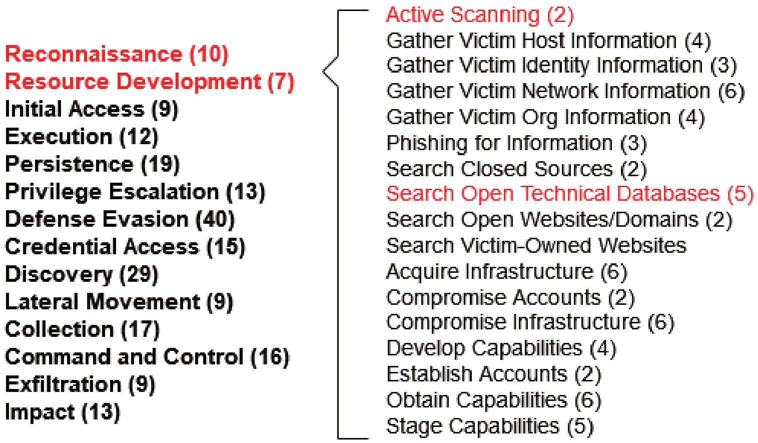
This book is designed to be interactive with code samples included for each chapter. Before moving on to the next chapter, be sure to install Python and the required Python libraries on your computer.

CHAPTER 1

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.



Reconnaissance (10)
Resource Development (7)
Initial Access (9)
Execution (12)
Persistence (19)
Privilege Escalation (13)
Defense Evasion (40)
Credential Access (15)
Discovery (29)
Lateral Movement (9)
Collection (17)
Command and Control (16)
Exfiltration (9)
Impact (13)

Active Scanning (2)
Gather Victim Host Information (4)
Gather Victim Identity Information (3)
Gather Victim Network Information (6)
Gather Victim Org Information (4)
Phishing for Information (3)
Search Closed Sources (2)
Search Open Technical Databases (5)
Search Open Websites/Domains (2)
Search Victim-Owned Websites
Acquire Infrastructure (6)
Compromise Accounts (2)
Compromise Infrastructure (6)
Develop Capabilities (4)
Establish Accounts (2)
Obtain Capabilities (6)
Stage Capabilities (5)

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)
```