Study Guide

Developing Ethical Hacking Tools with Python

Created By: Javier Garcia A, Teaching Assistant

Module 1: Introduction

Lesson 1.1: Introduction

Skills learned from this lesson: Know your Instructor, Course Structure, Why Python?

- Cybersecurity Professional Balance
  - Not everything needs to be studied. If you want to be a good cybersecurity professional, you only need a passion for it. You can balance your life, your family and your hobbies with cybersecurity practice.
- The Machine learning and Cybersecurity integration is a milestone in the field. When you integrate this into your analyst activities, you expand your scope to get a better result. For this course the structure is:
  - Module 1: The need for python in Cybersecurity
  - Module 2: Automating Information Gathering
  - Module 3: Writing a Keylogger in Python
  - Module 4: Zip Password Bruteforcing in Python
  - Module 5: Going Forward

Lesson 1.2: Why Cybersecurity Analysts and Penetration Testers Need Python

Skills learned from this lesson: Cybersecurity Automation, Python in the practice, time optimization.

- Common Tasks in Cybersecurity
  - As in other areas in Cybersecurity, there are common tasks that need to be performed daily. Several times, this becomes a boring activity and can cause a significant loss of time. Here is where automation is significant to the cybersecurity professional. With automation, it is possible to optimize the

repetitive and tedious activities such as vulnerability assessment, vulnerability analysis and some types of exploitation in a timely manner. But what could be an easy way to implement automation?

- Python as a Friend
  - Python is a single language easy to understand and practice. Additionally one must have many libraries developed and be ready to integrate with other tools such as Nmap, Nikto, etc. It is possible to adjust Python to the current needs of any project. This is the reason that python is considered the good friend of Cybersecurity professionals. The tool is ideal for implementing automation in cybersecurity projects.

Module 2: Automating Information Gathering

Lesson 2.1: A review of the typical pentesting process

Skills learned from this lesson: Penetration testing process, Penetration testing methodologies, Penetration testing standards

- What is Penetration Testing?
  - This a single question but a little difficult to respond. Some novices in cybersecurity confuse this with ethical hacking and, in the most extreme cases, include these activities as a single-phase on the cyber-kill chain. These two are topics out of the scope of this course, however we mention these for contextualizing the meaning of penetration testing.
  - Penetration testing could be defined as the art of testing the security strategy of an organization. This could be physical or logical. More accurately the definition is: “a penetration test is the practice of assessing an organization’s security strategy’s ability to protect critical data from the actions of a malicious actor” (C. Duffy, Python penetration testing for Developers)
  - We could mention some methodologies and standards commonly recognized for the penetration testing professional, these are:
    - OSSTMM
    - OWASP
    - NIST 800-115
    - PTES
    - And others that we encourage you to investigate.

- Each has his properly structure and order but usually is the same and only change the name regarding the standard, for example:
- PTES: Penetration Testing Execution Standard
- This has seven phases, as follows:
    1. Pre-engagements interactions
    2. Intelligence gathering
    3. OSINT
    4. Threat modeling
    5. STRIDE
    6. Vulnerability analysis
    7. Exploitation
    8. Post-Exploitation
    9. Reporting
- It Is important to mention that the first phase in all the methodologies and standards is to get the authorization and proper documentation for dimension, planning and executing the penetration test. This is the most important phase. If you don't get that authorization you will be committing a crime and this is penalized in several jurisdictions in and out of the USA.
- The next phases are commonly mapped between the standards and become repetitive as intelligence gathering or recognition of target, vulnerability scanning, vulnerability analysis, exploitation and make the report. One after the one is the same and you could imagine repeating this activity for different targets on the day.

Lesson 2.2: Combining Python Modules for Active Info Gathering Part 1
Skills learned from this lesson: Coding, Python packages, Python IDEs

- Scripting or coding
    - Both terms are very similar - too many people use one of these for reference as the same activity. This activity is related to the writing of some specific commands or codes in one machine pretending to execute one task.
- IDEs
    - There are many development environments, each one comfortable to the necessity of the developer. For python you can use any, but in this course, will

    be used Visual Studio Code. Sure, exist any others as charm, and you could investigate it.

- Packages
    - The key with python is single and easy to understand. You could use sentences very familiar with C# or C++, and you could integrate any package from the internet. There is even a popular site where you can download and index some of the most important packages for Ethical Hacking with Python and write your scripts or complex programs with a few code lines as the popular "Hello word." This could be found at https://pypi.org/ . One of the packages that you could integrate with your script is the popular tool Nmap. Please feel free to explore the many Python packages that exist on this site.

Lesson 2.3: Combining Python Modules for Active Info Gathering Part 2
Skills learned from this lesson: Information gathering, scanners, lists & dictionaries.

- Lists
    - As one type of variable, lists in python store values and elements and could be accessed using their index, ex: print (prime numbers[0]) . This access to the first element in one list called prime numbers.

index	0	1	2	3	4
Value	1	3	5	7	11

print(prime_numbers[0])
print 1

- Dictionaries
    - Dictionaries are like list but use keys instead of index

Key	IP	os	version	user	Name
Value	10.10.10.1	Windows	10 pro	Admin	Ana

```
print(systemproperty[“os”])

print Windows
```

In python, you could set a list of dictionaries.

- Scanning
    - Little script for information gathering.

```
#! usr/bin/env python

import nmap
nmap1 = nmap.PortScanner()
i = input("ingresa la direccion ip: ")
a = nmap1.nmap_banner(i)
b = nmap1.nmap_version(i)
print(a, b)
 c = nmap1.scaninfo(i)
print (c)
```

This little script could give you the info of banner and version of the target system, so it is the single and powerful of Python in Ethical hacking. Can you imagine how you could expand it if you mix this with automation?

Module 3: Writing a KeyLogger in Python

Lesson 3.1: Writing a Keylogger in Python part 1

Skills learned from this lesson: KeyLoggers, pynput, ftplib

- Keylogger
    - Keyloggers are popular hacking tools that could be used for getting information from the victims. Keyloggers are classified as physical and logical.
    - The truth here is that as an Ethical Hacker, you need to develop your scripts and be capable of reading and understand the code of other hackers (also malicious

hackers). In this activity, it is important that you have your IDE, your logic, your knowledge and most importantly, your laboratory for testing your scripts. In this way, we can proceed to create our basic keylogger program in python

- pynput
    - This library allows you to control and monitor input devices. Currently, mouse and keyboard input and monitoring are supported.
    - See here for the full documentation: https://pypi.org/project/pynput/
    - This library is the most important module for keylogger, so I recommend that you validate the complete documentation in the official PyPI link provided.
    - Let’s code:

```
#! usr/bin/env python

From pynput import Key, Listener
Import ftplib
Import logging

        Logdir = “”
Logging.basicConfig(filename=logdir+”klog-res.txt”), level=logging.DEBUG,
format = “%(asctime)s:%(message)s ”

def pressing_key(key)
        try:
                logging.info(str(key))
        except AttributeError:
                print (“A special key {0} has been pressed.”.format(key))
```

This easy code will show to us in screen the key that has been pressed. In the next lesson, we will describe what each part of the code means.

Lesson 3.2: Writing a Keylogger in Python part 2
Skills learned from this lesson: Functions, Methods, Listeners

- One important concept in Python code is indenting. Indenting is the space that exists from the border of the code line at the left to the starting code of our application. This makes the code readable.

  For example:

  ```
  def pressing_key(key)
          try:
                  logging.info(str(key))
          except AttributeError:
                  print ("A special key {0} has been pressed.".format(key))
  ```

- Methods
  - The methods are another type of function but this depends on the class that it belongs to.
- Functions
  - One important property of functions is the independence. This means that the function doesn't depend on any way of one class or program - the function simply exists and process the data.

Lesson 3.3: Writing a Keylogger in Python part 3
Skills learned from this lesson: Dependences, Vulnerable virtual machine, Metasploitable

- In the resources of the course in episode 3.3, you will find the links for installation of virtual box and metasploitable. Please check out that before continuing with the course.
- Is important to mention here that if you are interested in improving your skills as an infosec professional, you need continuous training, and you must have your own laboratory. One of the essential elements in that laboratory will be the virtual machines where you will deploy and test your scripts and settings before executing this in a real practice. In this way, as an ethical hacker, you could calculate the necessary resources and the potential impact that will cause the execution of the program against any target.
- When you have your laboratory ready with these two tools you can continue with the course. For this practice you need to initiate metasploitable and validate the current IP address that needs to be the same in the program that you have written. The credentials of metasploitable are as follows:

User: msfadmin
Pass: msfadmin

- The architecture of the exercise establishes the existence of two machines.

  Host machine Windows 10 (ip 192.168.0.X)
  Guess machine (virtual box) Metasploitable (ip. 192.168.0.103)

- The metasploitable machine will be the FTP server where the file with the pressed keys exist. For that reason the IP address needs to be the same that we specify in the program. You need to adjust this for your laboratory so this could be different in each case.

When you have done it, you need to follow the next steps:

1. Locate in windows (host machine)
2. Open cmd console and browse to the folder where you store the program, called "keylogger-p2.py".
3. When you be there executing it with the command: “python keylogger-p2.py” and you will see that the output message noticing that our program is running:

Started listening…

1. To test the program, open one text processor tool, like notepad, and write something (for testing purposes - you don’t need to write an extensive paragraph), when you finish, press the key that you have configured as an escape.

You will see the output message:

Connecting to the FTP and sending the data…

1. In the same directory, validate that the keylogger.txt file has been created. In our exercise you will note the file klog-res.txt. Feel free to review the content of the file and this will be the same type in the text processor tool. This confirms the correct coding of our program.
2. Now you need to validate that the file has been uploaded in the FTP. We change our machine for the metasploitable virtual machine as we were logged on it.
3. In the current directory, press the command “ls -a” and review the files. You will note the file “klog-res.txt”. Type the command “cat klog-res.txt” and validate that the content is the same as what we typed in the windows machine.
4. Validate the integrity of the file. To do this, you could use the most secure method for comparing the integrity of one file, hashing. You can analyze the file with md5 hash.
    - In windows you could use the command “cerutil -hashfile klog-res.txt MD5" and it will show you the hash: **8eae753b69ea2d604554bb04a9439d57**.
    - In metasploitable, execute the command to validate the integrity “md5sum klog-res.txt” and it will show you the hash: **8eae753b69ea2d604554bb04a9439d57**

   As you can see, it is the same hash and this means the file is integrity the same.

Module 4: Zip Password Bruteforcing in Python

Lesson 4.1: Bruteforcing Passwords with Python part 1

Skills Learned from this lesson: Bruteforcing, password cracking, protected file.

- Concepts
    - Bruteforcer
        - A bruteforcer is the noun that we give to the programs designed for executing a brute force attack to one application. This means that the program attacks one static parameter with several parameters that the application is waiting to compare and match, for example, the password.
    - Password Cracking
        - Password cracking is the term used in the infosec world for the technique in which an unauthorized entity breaks the secret key of an authentication system. This means that this entity attacks the system several times until the password matches the passwords list in a source file.

    - Protected file
        - For this practice, the protected file means one file. This could be a zipped file that is protected by a password ensuring one authentication schema. No one without the secret key would be able to view the content zipped.

Lesson 4.2: Brute Forcing Passwords with Python part 2
Skills Learned from this lesson: parsers, arguments, error handling

- The code of the program (libraries needed zipfile and argparse)
    - For our second application, we require to use parsers. This is one type of module that provides an interface to Python's internal parser and byte-code compiler. The primary purpose of this interface is to allow Python code to edit the parse tree of a Python expression and create executable code from this. This is better than trying to parse and modify an arbitrary Python code fragment as a string because parsing is performed in a manner identical to the code forming the application. It is also faster. (1)
- The program codes

```
from zipfile import ZipFile
import argparse
parser = argparse.ArgumentParser(description="\nUsage: python zipbrute.py -z -p ")
parser.add_argument("-z", dest="ziparchive", help="Zip archive file")
parser.add_argument("-p", dest="passfile", help="Password file")
parsed_args = parser.parse_args()
try:
ziparchive = ZipFile(parsed_args.ziparchive)
passfile = parsed_args.passfile
foundpass = ””

except:
print(parser.description)
 exit(0)

 with open(passfile, "r") as f:
```

```
 for line in f:
        password = line.strip("\n")
        password = password.encode("utf-8")

        try:
                foundpass = ziparchive.extractall(pwd=password)
                if foundpass == None:
                        print("\nFound password: ",password.decode())
        except RuntimeError:
                 pass
if foundpass == "":
 print("\nPassword not found. Try a bigger password list.")
```

Also have in mind that one function requires arguments for process the data, an argument is a data that is passed to the function for execute specific instructions, if this data not exist the function will not execute or will fall in an error. One example in our code is the fragment:

```
parser.add_argument("-z", dest="ziparchive", help="Zip archive file")
parser.add_argument("-p", dest="passfile", help="Password file")
```

- The arguments will be “-z” and “-p”
- Error handling is one of the most important parts in developing code programs. This way you drive the error, and the program will execute as planned or could crash the system. In our code you could identify the block.

```
try:
        foundpass = ziparchive.extractall(pwd=password)
        if foundpass == None:
                print("\nFound password: ",password.decode())
except RuntimeError:
```

- This means that the program will show the pass key in screen. At least one error in the program has occurred.

(1) Definition from: https://docs.python.org/2/library/parser.html

Lesson 4.3: Bruteforcing Passwords with Python part 3
Skills Learned from this lesson: Bruteforcing on ethical hacking, automation scripts, out the box thinking.

To execute successfully, our code is only necessary to open our program located in the windows machine and execute the program with the parameters that we set.

This is:

```
C:\users\”path”>python zipbrute.py -z archive.zip -p passlist.txt
```

Have in mind:

Zipbrute.py (our program)
Archive.zip (our target)
Passlist.txt (our list of predefined passwords)

- Must be in the same path. If not, you need to provide the complete route in the parameter and set the configuration of environment variables.
- There are a lot of password files on the internet. One of the more complete is "rock_you". This has a greater history that is out of the scope of the course, is near about 15 Gb of size, and could be useful for your ethical hacking tasks.
- In the same way, it is necessary to highlight that ethical hacking bruteforcing is a key part of the hacking process. The bad actors use techniques such as this to access systems and encrypted information that needs to be protected for obvious reasons.Understanding this process could help the ethical hacker to implement right countermeasures against the bad actors.
- Now, one of the topics that we touch in this course is automation. For an ethical hacker, this concept could bring advantages and disadvantages. For example, the ethical hacker needs to always be updated and must try to keep one step ahead of the bad actors. If you don't think about how to optimize your tools, your scripts, your attacks and your defenses, you and your company will take a huge loss.

- Thinking outside of the box is a topic that is coming to appear in many fields but especially in our infosec space. Integrating tools, automating tasks, improving response time is necessary for staying out of the line of fire. Python is one key tool that you could explode for integrating and automating many of the current tools that exist without the necessity of developing everything from scratch. One simple example could be the use of the Nmap library. Sending the results to one list of variables that need to be analyzed for one pattern configured in Splunk libraries is available in https://pypi.org/, with this result you could set one simulated attack in a period or a proper manner.

Module 5: Going Forward

Lesson 5.1: Additional resources and conclusions

Skills Learned from this lesson: Python in Cybersecurity, Python Resources, beyond cybersecurity.

That's it. The Cybrary team hopes that you learn as many things as possible from this course. There are many sources of information about python in cybersecurity and your election to use Cybrary is appreciated. Cybrary is a community that grows each day with the intention to share knowledge.
We encourage you to keep investigating automation in cybersecurity. There are a lot of resources and courses that could help you reach your objectives.

Review our content and remember the key phrase "**think outside of the box**." This will help you to be creative and integrate tools and knowledge that will support your daily infosec tasks.

Bibliography

Christopher Duffy (2016) Python penetration testing for developers. Birmingham, UK, Packt Publishing Ltd. ISBN 978-1-78712-818-7

Mark A. R. Kleiman (2010). When Brute Force Fails: How to Have Less Crime and Less Punishment. Princeton University Press. ISBN: 978-0691148649

Methods and classes (n.d) in docs python. Retrieved from https://docs.python.org/2/tutorial/classes.html

Differences between methods and functions (n.d) in geeks for geeks. Retrieved from https://www.geeksforgeeks.org/difference-method-function-python/