BB assist project

23 September 2023

Table of Contents

[Introduction 3](#_Toc165962188)

[BBAP version 1 4](#_Toc165962189)

[Features of app 9](#_Toc165962190)

[1.Define different site for learning 9](#_Toc165962191)

[2.Define docker image costume 10](#_Toc165962192)

[3.Define checklist 11](#_Toc165962193)

[4.ReconBBassist.py 11](#_Toc165962194)

[BBapp Verision 2 11](#_Toc165962195)

[ReconBBassist.py 20](#_Toc165962196)

# **Introduction**

**The Bug Bounty Assistance Program**

The Bug Bounty Assistance Program (BBAP) aims to empower security hunters by providing them with the knowledge, resources, and support they need to excel in the field of ethical hacking. By offering educational materials, mentorship, tools, and a collaborative community, the BBAP seeks to lower barriers to entry and foster the growth of a vibrant bug bounty community. Through its efforts, the BBAP ultimately contributes to the collective goal of making digital systems more secure, protecting both organizations and individuals from cyber threats. In an era where cybersecurity is paramount, initiatives like the BBAP are crucial in building a safer digital world.

The Bug Bounty Assistance Program (BBAP) aims to address these challenges and create an environment conducive to both seasoned security hunters and newcomers alike. It operates on several fundamental principles:

1. **Education and Training**: The BBAP will offer comprehensive educational resources and training modules. These resources will cover various aspects of bug hunting, including web application security, network security, and mobile application security. Educational materials will be available in various formats, such as written guides, video tutorials, and interactive courses.
2. **Mentorship and Support**: To help newcomers navigate the complexities of bug bounty programs, the BBAP will establish a mentorship system. Experienced security researchers will volunteer their time to guide and assist newcomers, providing valuable insights and advice.
3. **Resource Repository**: A centralized repository of tools, scripts, and resources will be maintained. Security hunters can access this repository to streamline their testing processes, making their bug hunting more efficient and productive.
4. **Platform Integration**: The BBAP will collaborate with major bug bounty platforms (e.g., HackerOne, Bugcrowd) to facilitate seamless integration. Researchers can access program details, submit reports, and receive feedback directly from within the BBAP platform.
5. **Community Forums**: The BBAP will host community forums where security hunters can collaborate, share knowledge, and discuss their experiences. These forums will serve as a hub for networking and building a sense of camaraderie among bug bounty enthusiasts.
6. **Recognition and Rewards**: To incentivize and acknowledge the contributions of security hunters, the BBAP will maintain a leaderboard and recognize top performers. Furthermore, the program will periodically host bug bounty challenges and competitions with attractive prizes.

# BBAP version 1

import docker

import sys

from PIL import Image

from termcolor import colored

import colorama

print("welcome to BB Assist")

from ascii\_magic import AsciiArt

my\_art = AsciiArt.from\_image('/Volumes/Personal/images/joker2.jpg')

my\_art.to\_terminal()

#################################################################

def Information\_Gathering():

print("Information Gathering process started...")

#################################################################

def Scanning\_and\_Enumeration():

print("Scanning and Enumeration process started...")

#################################################################

def Passive\_Scanning\_and\_Assessment():

print("Passive\_Scanning\_and\_Assessment process started...")

#################################################################

def Active\_Scanning\_and\_Testing():

print("Active\_Scanning\_and\_Testing process started...")

#################################################################

def Manual\_Testing():

print("Manual\_Testing process started...")

#################################################################

def Exploit\_Development():

print("Exploit\_Development process started...")

#################################################################

def Documentation\_and\_Reporting():

print("Documentation\_and\_Reporting process started...")

#################################################################

def Responsible\_Disclosure():

print("Responsible\_Disclosure process started...")

#################################################################

def Communication():

print("Communication process started...")

#################################################################

def Validation\_and\_Confirmation():

print("Validation\_and\_Confirmation process started...")

#################################################################

def Validation\_and\_Confirmation():

print("Validation\_and\_Confirmation process started...")

#################################################################

def Rewards\_and\_Acknowledgment():

print("Rewards\_and\_Acknowledgment process started...")

#################################################################

def Continuous\_Monitoring():

print("Continuous\_Monitoring process started...")

#################################################################

def Build\_a\_Portfolio():

print("Build\_a\_Portfolio process started...")

#################################################################

def Learning\_and\_Skill\_Enhancement():

print("Build\_a\_Portfolio process started...")

#################################################################

def Network\_and\_Collaboration():

print("Network\_and\_Collaboration process started...")

#################################################################

def Tools\_and\_Automation():

print("Tools\_and\_Automation process started...")

def kali():

print("kali process started...")

client = docker.from\_env()

# Replace 'my\_image' with the name of your Docker image

client.containers.run('my\_image', detach=True)

print("Docker container started.")

def Recon():

print("Recon Tools...")

Tools\_Items = {

'1': Recon,

}

while True:

print("\n Tools menu:")

print("1. Recon")

print("B. Back")

tools\_choice = input("Choose an option: ")

if tools\_choice in Tools\_Items:

Tools\_Items[tools\_choice]()

elif tools\_choice == 'B':

break

else:

print("Invalid option, please try again.")

#################################################################

def Ethical\_Conduct\_and\_Compliance():

print("Ethical\_Conduct\_and\_Compliance process started...")

# Add the remaining process steps in a similar way...

# Create a dictionary of process steps

process\_steps = {

'1': Information\_Gathering,

'2': Scanning\_and\_Enumeration,

'3': Passive\_Scanning\_and\_Assessment,

'4': Active\_Scanning\_and\_Testing,

'5': Manual\_Testing,

'6': Exploit\_Development,

'7': Documentation\_and\_Reporting,

'8': Responsible\_Disclosure,

'9': Communication,

'10': Validation\_and\_Confirmation,

'11': Rewards\_and\_Acknowledgment,

'12': Continuous\_Monitoring,

'13': Build\_a\_Portfolio,

'14': Learning\_and\_Skill\_Enhancement,

'15': Network\_and\_Collaboration,

'16': Tools\_and\_Automation,

'17': Ethical\_Conduct\_and\_Compliance,

# Add the remaining process steps to this dictionary...

}

while True:

print("\nMain Menu:")

print("1. Information Gathering (Reconnaissance)")

print("2. Scanning and Enumeration")

print("3. Passive Scanning and Assessment")

print("4. Active Scanning and Testing")

print("5. Manual Testing")

print("6. Exploit Development")

print("7. Documentation and Reporting")

print("8. Responsible Disclosure")

print("9. Communication")

print("10. Validation and Confirmation")

print("11. Rewards and Acknowledgment")

print("12. Continuous Monitoring")

print("13. Build a Portfolio")

print("14. Learning and Skill Enhancement")

print("15. Network and Collaboration")

print("16. Tools and Automation")

print("17. Ethical Conduct and Compliance")

print("18. Exit")

main\_choice = input("Choose an option: ")

if main\_choice in process\_steps:

process\_steps[main\_choice]()

elif main\_choice == '18':

break

else:

print("Invalid option, please try again.")

#################################################################

# Features of app

1. Define docker image costume
2. Define different site for learning
3. Define checklist
4. ReconBBassist.py

## 1.Define different site for learning

def Learning\_and\_sites():

print("Build\_a\_Portfolio process started...")

#some sites for refrences

import webbrowser

def open\_website(url):

webbrowser.open(url, new=2)

websites = {

'1': {

'name': 'Google',

'url': 'https://www.google.com'

},

'2': {

'name': 'StackOverflow',

'url': 'https://www.stackoverflow.com'

},

'3': {

'name': 'Real Python',

'url': 'https://www.realpython.com'

},

'4': {

'name': 'Exit',

'url': ''

},

}

while True:

print("\nWebsites Menu:")

for key, site in websites.items():

print(f"{key}. {site['name']}")

choice = input("Choose a website to open: ")

if choice in websites:

if choice == '4':

break

open\_website(websites[choice]['url'])

else:

print("Invalid option, please try again.")

## 2.Define docker image costume

Based on the sources provided, you can use the docker-py library (also known as docker in Python) to interact with Docker and run a Docker image from your Python script [stackoverflow.com](https://stackoverflow.com/questions/47433576/starting-docker-container-using-python-script).

First, you need to install the docker Python library. You can do this using pip:

pip install docker

Then, you can modify the Recon function in the script to run a Docker image. Here's an example of how you might do this:

import docker

def Recon():

print("Recon process started...")

client = docker.from\_env()

# Replace 'my\_image' with the name of your Docker image

client.containers.run('my\_image', detach=True)

print("Docker container started.")

In this example, docker.from\_env() creates a DockerClient instance configured with appropriate environment variables [stackoverflow.com](https://stackoverflow.com/questions/47433576/starting-docker-container-using-python-script). The run method of DockerClient.containers runs a Docker container with the specified image. The detach=True argument means the container will run in the background [stackoverflow.com](https://stackoverflow.com/questions/47433576/starting-docker-container-using-python-script).

Please replace 'my\_image' with the name of your Docker image. If your Docker image requires any command line arguments, you can provide them as additional arguments to the run method.

Remember, this is a very basic example. In a real-world application, you would need to handle errors and edge cases. For example, you might need to check if the Docker image exists and handle the case where the Docker container fails to start. You would also need to ensure that your Python script has the necessary permissions to interact with Docker.

## 3.Define checklist

## 4.ReconBBassist.py

# BBapp Verision 2

import docker

import sys

import mysql.connector

from PIL import Image

from termcolor import colored

from ascii\_magic import AsciiArt

from ReconBBassist import ReconBB

##########image of start###############

print("welcome to BB Assist")

#my\_art = AsciiArt.from\_image('/Volumes/myjob/Computer/Programming/Projects/python/BB assist/BB assist copy.png')

#clearmy\_art.to\_terminal()

#################################################################

def Recon():

print("Manual\_Testing process started...")

ReconBB()

#################################################################

def Manual\_Testing():

print("Manual\_Testing process started...")

#################################################################

def Exploit\_Development():

print("Exploit\_Development process started...")

#################################################################

def Documentation\_and\_Reporting():

print("Documentation\_and\_Reporting process started...")

#################################################################

def Communication():

print("Communication process started...")

#################################################################

def Validation\_and\_Confirmation():

print("Validation\_and\_Confirmation process started...")

#################################################################

def Continuous\_Monitoring():

print("Continuous\_Monitoring process started...")

#################################################################

def Build\_a\_Portfolio():

print("Build\_a\_Portfolio process started...")

#################################################################

def Learning\_and\_sites():

print("Build\_a\_Portfolio process started...")

#some sites for refrences

import webbrowser

def open\_website(url):

webbrowser.open(url, new=2)

websites = {

'1': {

'name': 'awesome-bugbounty-tools',

'url': 'https://github.com/vavkamil/awesome-bugbounty-tools'

},

'2': {

'name': 'Shodan',

'url': 'https://www.shodan.io'

},

'3': {

'name': 'eyezoome',

'url': 'https://www.zoomeye.org'

},

'4': {

'name': 'Censys',

'url': 'https://censys.com'

},

'5': {

'name': 'ypugetsignal',

'url': 'https://www.yougetsignal.com'

},

'6': {

'name': 'virustotal',

'url': 'https://www.virustotal.com/gui/home/upload'

},

'E': {

'name': 'Exit',

'url': ''

}

}

while True:

print("\nWebsites Menu:")

for key, site in websites.items():

print(f"{key}. {site['name']}")

choice = input("Choose a website to open: ")

if choice in websites:

if choice == 'E':

break

open\_website(websites[choice]['url'])

else:

print("Invalid option, please try again.")

#################################################################

def Network\_and\_Collaboration():

print("Network\_and\_Collaboration process started...")

#################################################################

def Tools():

print("Tools\_and\_Automation process started...")

#use image docker custome

def kali():

print("kali process started...")

client = docker.from\_env()

# Replace 'my\_image' with the name of your Docker image

client.containers.run('my\_image', detach=True)

print("Docker container started.")

def Recon():

print("Recon Tools...")

Tools\_Items = {

'1': kali,

}

while True:

print("\n Tools menu:")

print("1. Recon")

print("B. Back")

tools\_choice = input("Choose an option: ")

if tools\_choice in Tools\_Items:

Tools\_Items[tools\_choice]()

elif tools\_choice == 'B':

break

else:

print("Invalid option, please try again.")

#################################################################

def Ethical\_Conduct\_and\_Compliance():

print("Ethical\_Conduct\_and\_Compliance process started...")

#################################################################

def Vulnerabilities():

print("Ethical\_Conduct\_and\_Compliance process started...")

# Define your functions

#Server-side-attack

def SQL\_Injection():

pass # Add your code here

def Authentication():

pass # Add your code here

def Directory\_Traversal():

pass # Add your code here

def Command\_Injection():

pass # Add your code here

def Business\_logic\_vulnerabilities():

pass # Add your code here

def Inforamtion\_Disclosure():

pass # Add your code here

def Access\_Control():

pass # Add your code here

def File\_upload\_vulnerabilities():

pass # Add your code here

def Race\_conditions():

pass # Add your code here

def SSRF():

pass # Add your code here

def XXE():

pass # Add your code here

#Client-side attacks

def XSS():

pass # Add your code here

def CSRF():

pass # Add your code here

def ClickJacking():

pass # Add your code here

def DomBased():

pass # Add your code here

def CORS():

pass # Add your code here

def WebSockets():

pass # Add your code here

#Advanced

def Http\_Request\_Smuggling():

pass # Add your code here

def Serversidetemplate\_injection():

pass # Add your code here

def Insecure\_deserialization():

pass # Add your code here

def OAuth\_Authentication():

pass # Add your code here

def web\_cache\_poisoning():

pass # Add your code here

def HTTP\_host\_header\_attacks():

pass # Add your code here

def JWT\_attacks():

pass # Add your code here

def Prototype\_Pollution():

pass # Add your code here

def GraphQL\_API\_vulnerabilities():

pass # Add your code here

# Create a dictionary to map vulnerability names to functions

vulnerabilities = {

"SQL Injection": SQL\_Injection,

"Authentication": Authentication,

"Directory\_Traversal":Directory\_Traversal,

"Command\_Injection":Command\_Injection,

"Business\_logic\_vulnerabilities":Business\_logic\_vulnerabilities,

"Inforamtion\_Disclosure":Inforamtion\_Disclosure,

"Access\_Control":Access\_Control,

"File\_upload\_vulnerabilities":File\_upload\_vulnerabilities,

"Race\_conditions":Race\_conditions,

"SSRF":SSRF,

"XXE":XXE,

"XSS":XSS,

"CSRF":CSRF,

"ClickJacking":ClickJacking,

"DomBased":DomBased,

"CORS":CORS,

"WebSockets":WebSockets,

"Http\_Request\_Smuggling":Http\_Request\_Smuggling,

"Serversidetemplate\_injection":Serversidetemplate\_injection,

"Insecure\_deserialization":Insecure\_deserialization,

"OAuth\_Authentication":OAuth\_Authentication,

"web\_cache\_poisoning":web\_cache\_poisoning,

"HTTP\_host\_header\_attacks":HTTP\_host\_header\_attacks,

"JWT\_attacks":JWT\_attacks,

"Prototype\_Pollution":Prototype\_Pollution,

"GraphQL\_API\_vulnerabilities":GraphQL\_API\_vulnerabilities,

# Add the rest of your vulnerabilities here...

}

while True:

# Print the menu

for i, vulnerability in enumerate(vulnerabilities, start=1):

print(f"{i}. {vulnerability}")

# Ask the user to choose a vulnerability

choice = int(input("Enter the number of a vulnerability to run, or 0 to exit: "))

# Exit the loop if the user chooses 0

if choice == 0:

break

# Get the chosen vulnerability

vulnerability = list(vulnerabilities.values())[choice-1]

# Run the chosen vulnerability

vulnerability()

#################################################################

def Web\_app\_Checklist():

print("web\_app\_checklist process started...")

# Add the remaining process steps in a similar way...

def main():

print("test")

# Create a dictionary of process steps

process\_steps = {

'1': Recon,

'5': Manual\_Testing,

'6': Exploit\_Development,

'7': Documentation\_and\_Reporting,

'12': Continuous\_Monitoring,

'13': Build\_a\_Portfolio,

'14': Learning\_and\_sites,

'15': Network\_and\_Collaboration,

'16': Tools,

'17': Ethical\_Conduct\_and\_Compliance,

'18': Vulnerabilities,

'19': Web\_app\_Checklist,

# Add the remaining process steps to this dictionary...

}

while True:

print("\nMain Menu:")

print("1.Recon")

print("5. Manual Testing")

print("6. Exploit Development")

print("7. Documentation and Reporting")

print("12. Continuous Monitoring")

print("13. Build a Portfolio")

print("14. Learning and Sites")

print("15. Network and Collaboration")

print("16. Tools")

print("17. Ethical Conduct and Compliance")

print("18. vulnerabities")

print("19. Web app Checklist")

print("E. Exit")

main\_choice = input("Choose an option: ")

if main\_choice in process\_steps:

process\_steps[main\_choice]()

elif main\_choice == 'E':

break

else:

print("Invalid option, please try again.")

main()

## ReconBBassist.py

#BB assist project

#here we check steps of Recon

#1.find sunbdomains

def ReconBB():

while True:

print("\nMain Menu:")

print("1. subfinder")

print("2. dnsx")

print("3. httpx")

print("4. nnuclei")

print("5. Gowitness")

print("6. ffuf")

main\_choice = input("Choose an option: ")

if main\_choice in process\_steps:

process\_steps[main\_choice]()

elif main\_choice == 'E':

break

else:

print("Invalid option, please try again.")

def Information\_Gathering():

print("Information Gathering process started...")

#Recon

def print\_checklist(checklist):

for i, (item, checked) in enumerate(checklist.items(), start=1):

status = "[x]" if checked else "[ ]"

print(f"{i}. {status} {item}")

def toggle\_item(checklist, item\_number):

item = list(checklist.keys())[item\_number-1]

checklist[item] = not checklist[item]

checklist = {

"Item 1": False,

"Item 2": False,

"Item 3": False,

# Add more items...

}

while True:

print\_checklist(checklist)

item\_number = int(input("Enter item number to check/uncheck, or 0 to exit: "))

if item\_number == 0:

break

elif 1 <= item\_number <= len(checklist):

toggle\_item(checklist, item\_number)

else:

print("Invalid item number, please try again.")

def subfinder():

print("Running subfinder...")

def dnsx():

print("Running dnsx...")

def httpx():

print("Running httpx...")

#def ni():

#print("Running nuclei...")

def Gowitness():

print("Running Gowitness...")

def ffuf():

print("Running ffuf...")

process\_steps = {

'1': subfinder,

'2': dnsx,

'3': httpx,

#'4'; ni,

'5': Gowitness,

'6': ffuf,

# Add the remaining process steps to this dictionary...

}