Shri Vaishnav Vidyapeeth Vishwavidyalaya

Shri Vaishnav Institute of Information Technology

Session-2021-22

Project Report

On

HackBox (A complete pentesting toolkit)

Subject- Software Engineering and project management (BTCS503)
Year/Semester- III year / V Semester
Class / Section – CSE-ICS / G

Submitted by –

Siddharth Khandelwal (19100BTCSICS05457)
Kartik Bajpay (19100BTCSICS05442)
Somuya Sharma (19100BTCSICS05459)
Sneha Shukla (19100BTCSICS05458)
Sumit Prajapat (19100BTCSICS05463)

Under the guidance of –

Prof. Richa Jain

Disclaimer

Unethical Hacking is any activity that aims to exploit and illegally access a computer system, device, or network, without explicit permission from its owner. Causing harm is sometimes only a side-product of hacking, not a necessary element. Usage of HackBox for attacking targets without prior mutual consent is illegal. It's the end user's responsibility to obey all applicable local, state and federal laws. Team HackBox assume no liability and are not responsible for any misuse or damage caused by the tools. All content within the toolkit that relates to hacking is for educational purposes only with the goal of furthering cybersecurity and should be regarded as 'ethical hacking.'

Abstract:-

Red Team Exercise is an imitation of multi-layered cyber-attack targeting agreed upon objectives that include networks, technical and physical assets, storage devices and many more. The exercise and assessment performed helps in improving your security defenses by letting you experience a real-world data breach and thereby giving a bigger picture of your organization's risk posture, security architecture, and your team's readiness in detecting and mitigating the threat proactively. Red Team Exercise unfolds security vulnerabilities by penetrating your networks, assessing your processes, and testing the defensive capabilities of your security teams in all possible ways. This helps in taking the necessary steps to update your security layers accordingly.

For conducting these penetration tests ethical hackers need tools to perform various hack to check the security of a network, website, application, etc.
The problem which is faced by mostly new comers is to find all the latest tools required for a pentest at a single place with all the instructions on how to use that tool.
With our project hackbox we have work on this problem not only this toolkit offers various different tools from different author's from GitHub but also it provide the information of that tool and how that tool works along with its original GitHub page so that the new comers would understand there tools rather than using them blindly as a script kiddie.

Index

Contents----------page no.

Introduction to HACK BOX: -

Hack box is a simple cli based penetration testing toolkit which works on CLI & GUI both Linux distribution. It is beginner friendly and can work on low end hardware it includes almost all type of attack tested in a Red team exercises such as Xss payload injection and reverse shell etc.

With this toolkit we no longer need to search or remember different types of hacking tools present in the system. This is beginner friendly tool as the newbies don't know much about the tools and other then the default tools present in the os's such as KALI LINUX and PERROT OS

It is highly customizable as all the tool are scripted in shell and python which are much easier to understand.

Required hardware

The hardware required for this os is very low although there are some basic requirements for proper functioning of this toolkit.

1) Dual core processor
2) 2GB RAM
3) At least 40GB hard disk
4) Wi-Fi adaptor for internet access (With 802.11 IEEE for wireless attack recommended)

Objective:-

- To develop a simple beginner friendly low end software for ethical hackers so that they can understand the tools more efficiently.
- To provide all the tools at a single place.
- To create a hacking toolkit that can run on any Linux distribution.
- To develop a tool for cyber awareness and understanding basic tools
- Every hacker has their preferred hacking tools in their toolbox to perform specific attacks, collect information and secure their systems, but that doesn't mean you should miss out on the latest hacking tools released in the InfoSec community. After all, these tools are designed to improve and simplify existing penetration testing tools, automate the process or exploit current vulnerabilities.
- Attacks are on the rise as more employees are working from home.
- HackBox has really a lot of differences from other tools, but without knowing your requirements it is not possible to guess what is relevant for you.

Innovation and usefulness

1) It can run on a very low end hardware.
2) Al the required tools are at one place.
3) Easy to install your favourite tools.
4) Command line interface based alternative for Graphical user interface tools for low end cli based operating system
5) Highly customizable

Need for HACK BOX

1. **Who should use: -**
   Although this tool kit is perfect for any pentester or red team member the tool is recommended for new comers as all the tools required are at one place and they don't need to remember every tool (just select the tool kit and run the tool).

2. **Reduce cost: -**
   As it run's on any Linux distribution (which is open source) & can run on a low end hardware, a perfect hacking setup can be easily build at a low cost.

3. **Highly customizable: -**
   As all the script are written in shell or python which are much easier to understand and all the code is open source hence it can be customized.

Feasibility Study

TECHINICAL FEASIBILITY: -

Hack box is a simple cli based penetration testing toolkit which works on CLI & GUI both Linux distribution. The scripts are written in shell or python which are much easier to understand and all the code is open source hence it can be customized. And Linux and our tool kit both are free.

OPERATIONAL FEASIBILITY: -

User-friendly

Although this tool kit is perfect for any pentester or red team member the tool is recommended for new comers as all the tools required are at one place and they don't need to remember every tool (just select the tool kit and run the tool).

Portability

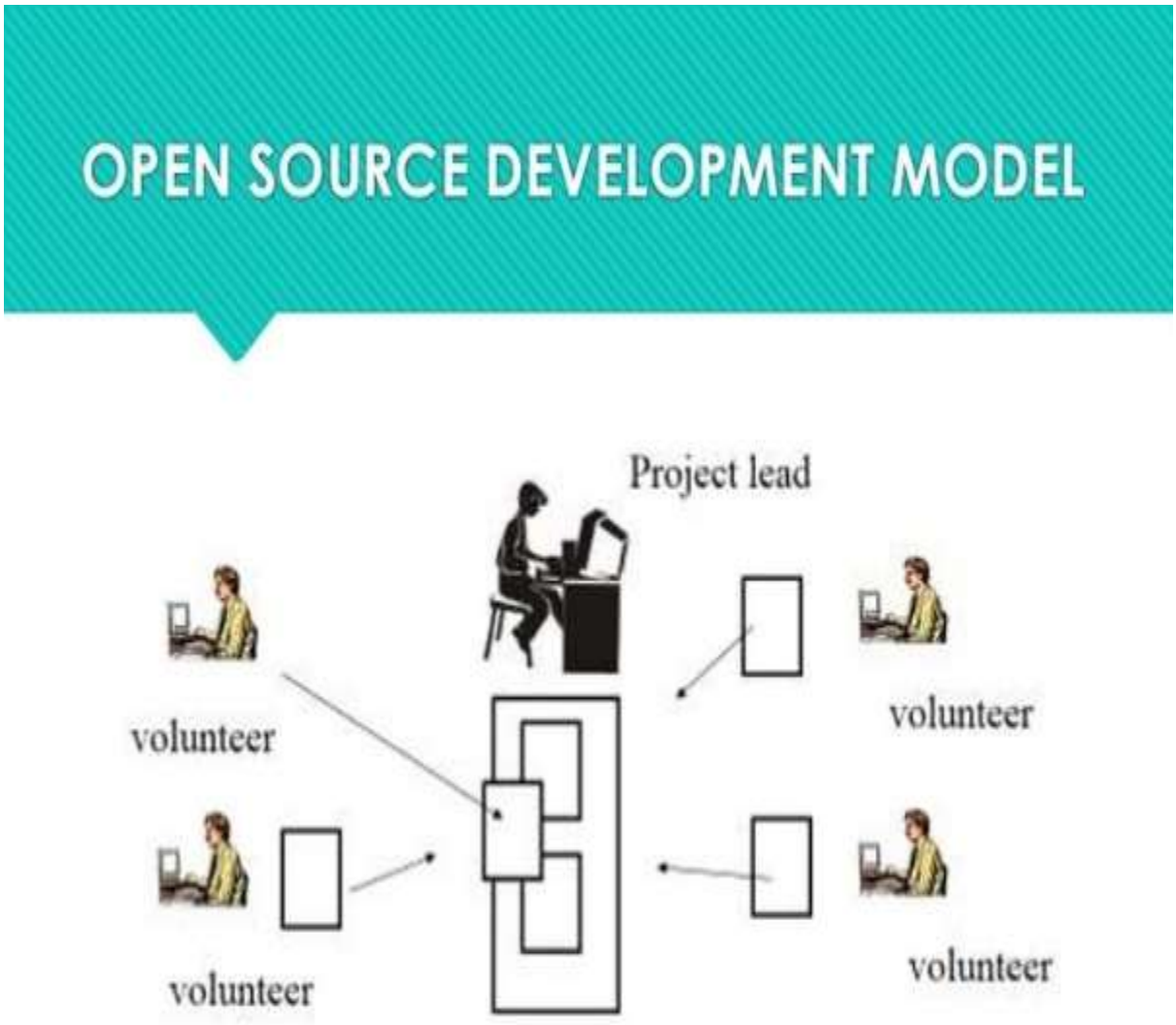
As all the script are written in shell or python which are much easier to understand and all the code is open source hence it can be customized. And Linux and our tool kit both are free.

Availability

This software will be available always.

ECONOMIC FEASILITY: -

The computerized system takes care of the present existing system's data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports. As it run's on any Linux distribution (which is open source) & can run on a low end hardware, a perfect hacking setup can be easily build at a low cost using our toolkit.



Literature study

Overview

Our project is based on the open source technology all the things we need or every tool in our tool kit are open source even the operating system on which it runs (Linux) is open source.

What is penetration testing?

A penetration test, also called a pen test or ethical hacking, is a cybersecurity technique organizations use to identify, test and highlight vulnerabilities in their security posture. These penetration tests are often carried out by ethical hackers. These in-house employees or third parties mimic the strategies and actions of an attacker in order to evaluate the hack ability of an organization's computer systems, network or web applications. Organizations can also use pen testing to test their adherence to compliance regulations.

What is Red Team test?

Red Team Exercise is an imitation of multi-layered cyber-attack targeting agreed upon objectives that include networks, technical and physical assets, storage devices and many more. The exercise and assessment performed helps in improving your security defenses by

letting you experience a real-world data breach and thereby giving a bigger picture of your organization's risk posture, security architecture, and your team's readiness in detecting and mitigating the threat proactively. Red Team Exercise unfolds security vulnerabilities by penetrating your networks, assessing your processes, and testing the defensive capabilities of your security teams in all possible ways. This helps in taking the necessary steps to update your security layers accordingly.

What is an open source software?

Open source software is the computer software developed either by an individual, group or an organization to meet certain requirements and it is available for any modifications based on its developing body's interest. Open source software is published openly for general public and here the source code is open for all. For open source software the users do not need to spend any cost. It is available under free licensing. It depends on donations and support as its main source of find.

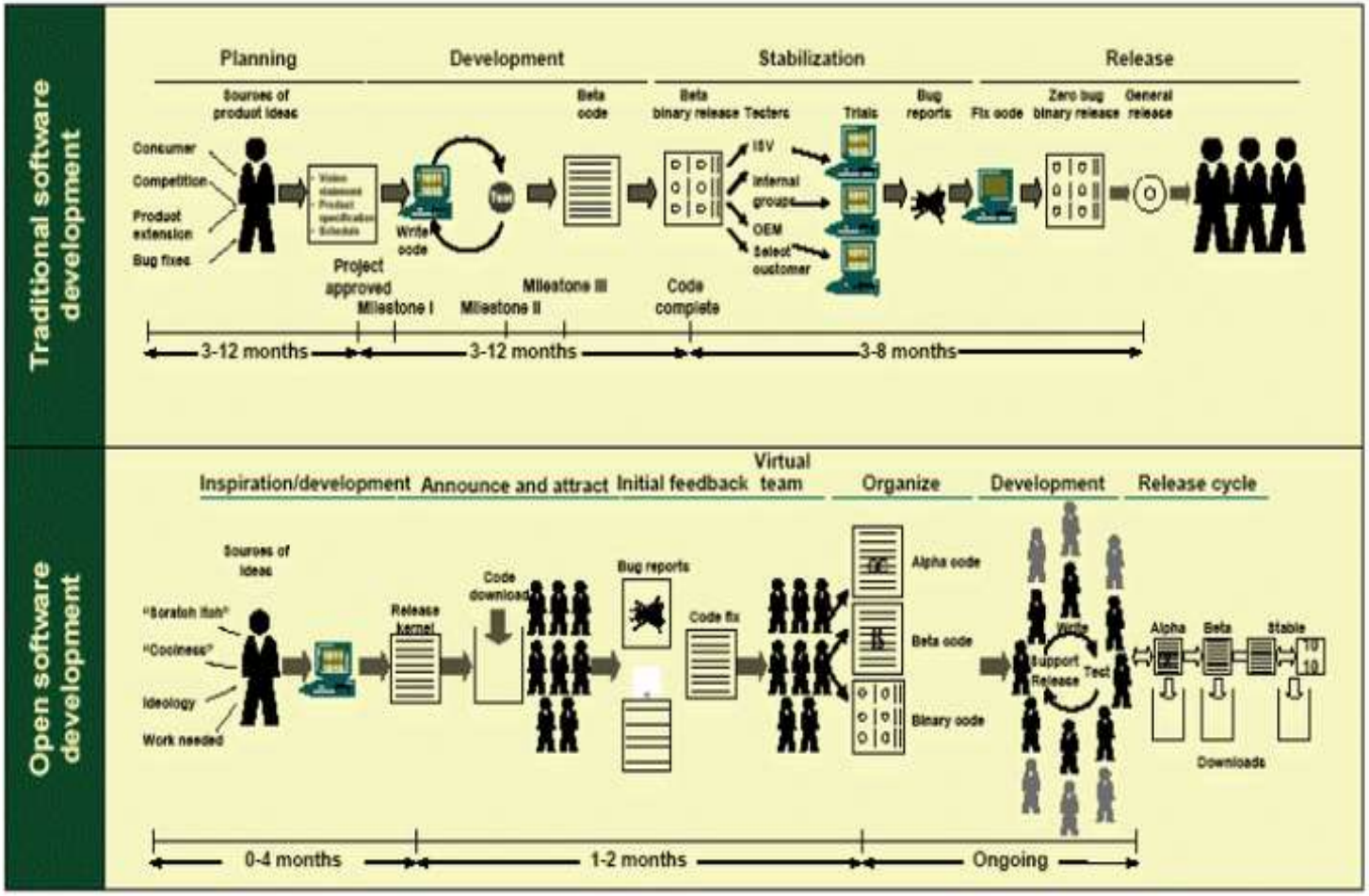
Some examples of open source software are Firefox, OpenOffice, Zimbra, VCL media player, Thunderbird.

Benefits of making an open source software

- Although this tool kit is perfect for any pentester or red team member the tool is recommended for new comers as all the tools required are at one place and they don't need to remember every tool (just select the tool kit and run the tool).
- As it run's on any Linux distribution (which is open source) & can run on a low end hardware, a perfect hacking setup can be easily build at a low cost.
- All the script are written in shell or python which are much easier to understand and all the code is open source hence it can be customized.
- Development become easy and efficient as more and more people can contribute to the project and test it.
- Bugs finding and adding features become much easier.
- Here in open source software users can customize.

OPEN SOURCE VERSUS TRADITIONAL SOFTWARE DEVELOPMENT

Fundamental Different Principles Adapted



Work done by others

There are various toolkits made by different developers for different purposes what they lack behind is the completeness there is no complete pentesting toolkit along with the description of each and every tool which is one more thing they are missing out.
Understanding the hacking tool is very important for any ethical hacker. Without the prior knowledge of tools he/she is just a script kiddie running some scripts downloaded from internet.
The demand for cyber security services was not as great as it is now. Companies were more interested in IT, CRMs and digitalization. As a result, new graduates went to pursue careers in other cyber fields, which is why there has been a shortage of cyber-security professionals. As a result, the shortage of cyber-security professionals will likely continue until 2028.

Projected Demand for Cybersecurity Skills

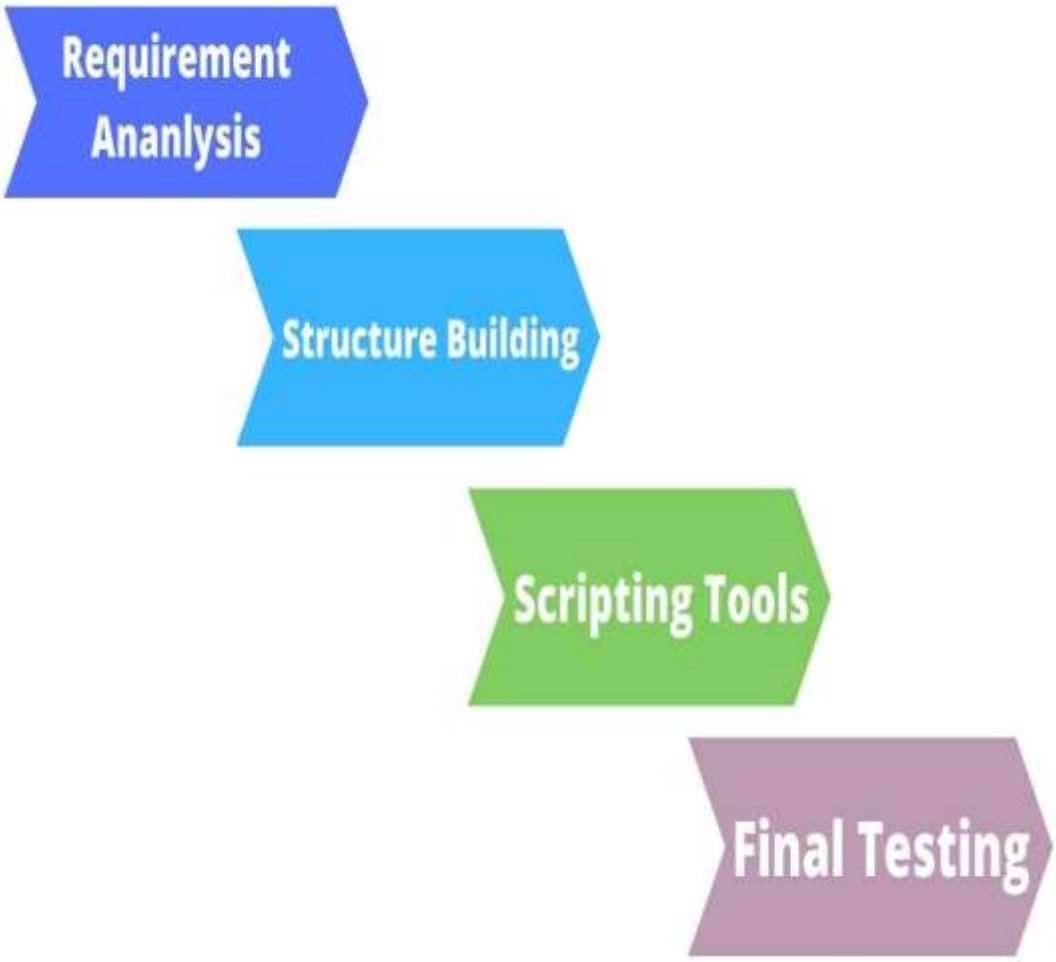
Skill	2018 Openings	2018–2023 5-Year Projected Growth
Public Cloud Security	1,333	170%
IoT Network Security	1,092	144%
Cybersecurity Strategy	3,184	120%
Dynamic Application Security Testing	1,892	120%
Cloud Security Architecture	1,902	113%
NIST Cybersecurity Framework	19,508	111%
Cybersecurity Assessments	9,543	93%
Cloud Security Applications	1,148	87%
Salesforce Security	2,300	77%
Open Web Application Security Project (OWASP)	10,410	61%

Software Engineering Approach

Software model

Waterfall model

The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. We have divided are project in 4 phases that are:-



Benefits:

- The project scope stays relatively static, meaning cost and timelines can be determined early on in the project.
- By completing a full design early in the project, changes to systems stay minimal, meaning the cost to fix and alter designs is kept low.
- A structured approach to a project means that everyone understands what needs to be done and when. SMEs can effectively plan their time over the fixed period.
- By having detailed documentation and designs, a project can lose key members without too much hassle since the documentation describes in reasonable detail how any SME of the product or skill are needed to complete the work.

Constraints:

- It is hard to allow for new requirements in an ever-changing world.
- A project that has dependencies on relatively unstable products which are constantly in flux may also cause constraint.
- It is difficult to estimate the total time a project will take to complete.
- A large amount of contingency is, more often than not, added into timescales.

Why to use waterfall model

As our project is an open source software so by using the waterfall model, project's scope stays relatively static timelines can be determined early on in the project. By completing a full design early in the project, changes to systems stay minimal. A structured approach to a project means that everyone understands what needs to be done and when. By having detailed documentation and designs, a project can lose key members without too much hassle since the documentation describes in reasonable detail.

Requirement analysis

Data requirement: - As the tool kit is open source that means that there is no user credential required for accessing the tools and the toolkits. But the tools may require some input from the user such as the I.P address of victim or the victim website or the place where you want to save the information or the payload etc.

Software requirements: - There are certain software requirements for the proper functioning of HackBox. All these requirement would be checked by Install.sh file and the not fulfilled one would be fulfilled instantly.

1. **Operating System: -** HackBox can work on any Linux based operating system but it would be much easier to use on a Linux based operating system specifically for Hacking.

2. **Up to date system: -** It is always a better option to run it on an updated operating system for less time consumption.

3. **Lolcat: -** Lolcat is a utility for Linux, BSD and OSX which concatenates like similar to cat command and adds rainbow colouring to it. Lolcat is primarily used for rainbow colouring of text in Linux Terminal.

4. **FIGlet: -** FIGlet prints its input using large characters made up of ordinary screen characters. FIGlet output is generally reminiscent of the sort of "signatures" many people like to put at the end of e-mail and UseNet messages. It is also reminiscent of the output of some banner programs, although it is oriented normally, not sideways.

5. **Boxes: -** Boxes is a simple, configurable command line program which can draw any kind of box around its input text. It filters text and draws shapes around it – it's practically a text filter. In fact it is designed to be integrated with your editor as a text filter (supports Vim default). It can draw shapes ranging from simple boxes to complex ASCII art.

6. **Flask: -** Flask is one of the most popular web application frameworks written in Python. It is a micro framework designed for an easy and quick start. Extending with tools and libraries adds more functionality to Flask for more complex projects.

7. **Requests: -** Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to urllib3.

System requirements: - The hardware required for this os is very low although there are some basic requirements for proper functioning of this toolkit.

1) Dual core processor
2) 2GB RAM
3) At least 40GB hard disk
4) Wi-Fi adaptor for internet access (With 802.11 IEEE for wireless attack recommended)

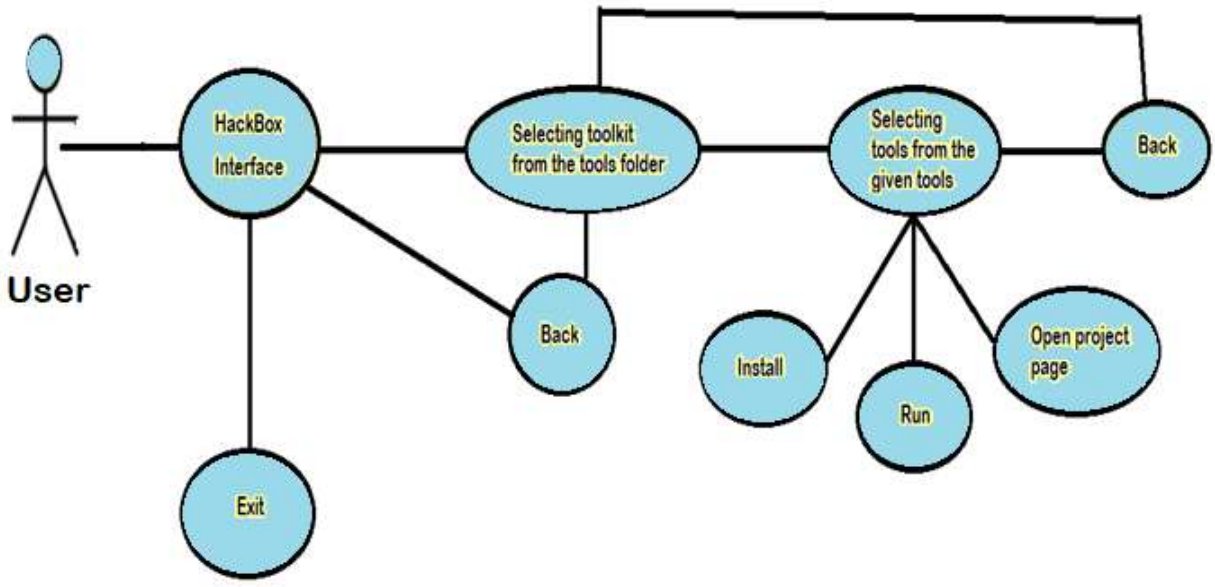
Supplementary Specification

Unethical Hacking is any activity that aims to exploit and illegally access a computer system, device, or network, without explicit permission from its owner. Causing harm is sometimes only a side-product of hacking, not a necessary element.

Usage of hacking-tool for attacking targets without prior mutual consent is illegal. It's the end user's responsibility to obey all applicable local, state and federal laws. Team HackBox assume no liability and are not responsible for any misuse or damage caused by this program.

Use Case Model

In software and system engineering design, creating use cases allows developers to show how a system intends to work and how users might apply it instead. Brainstorming other uses can help developers fix potential issues or model its features and techniques to refine how users interact with the platform or program. Knowing more about use cases can help you develop professional capabilities to use in a software development career. In this article, we explore what use cases are and what elements get included, how they benefit software development and provide example scenarios to better help you understand this technological term.



Design

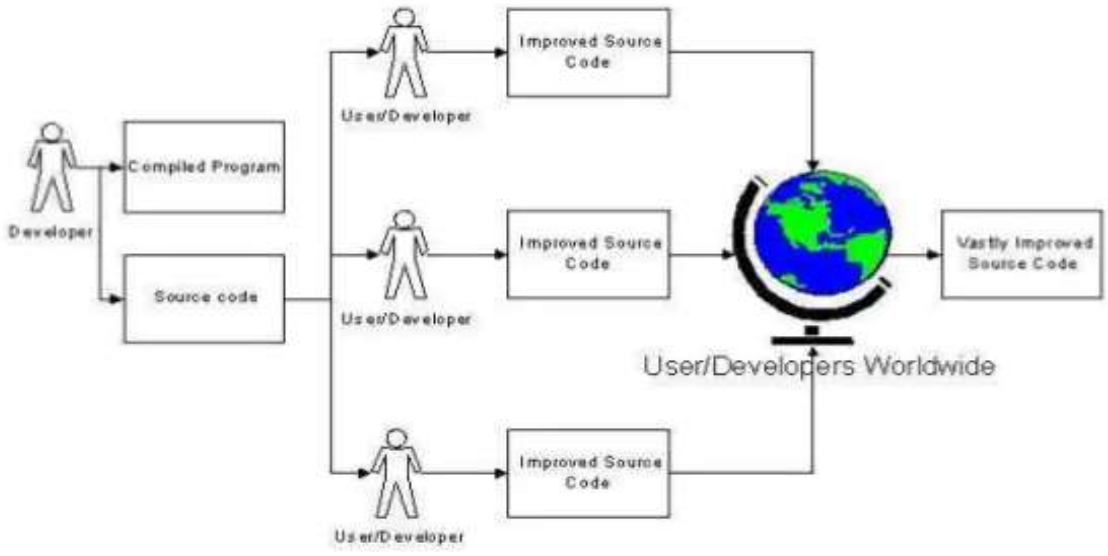
Software Design Concepts

Every software process is characterized by basic concepts along with certain practices or methods. Methods represent the manner through which the concepts are applied. As new technology replaces older technology, many changes occur in the methods that are used to apply the concepts for the development of software. However, the fundamental concepts underlining the software design process remain the same, some of which are described here

Architecture

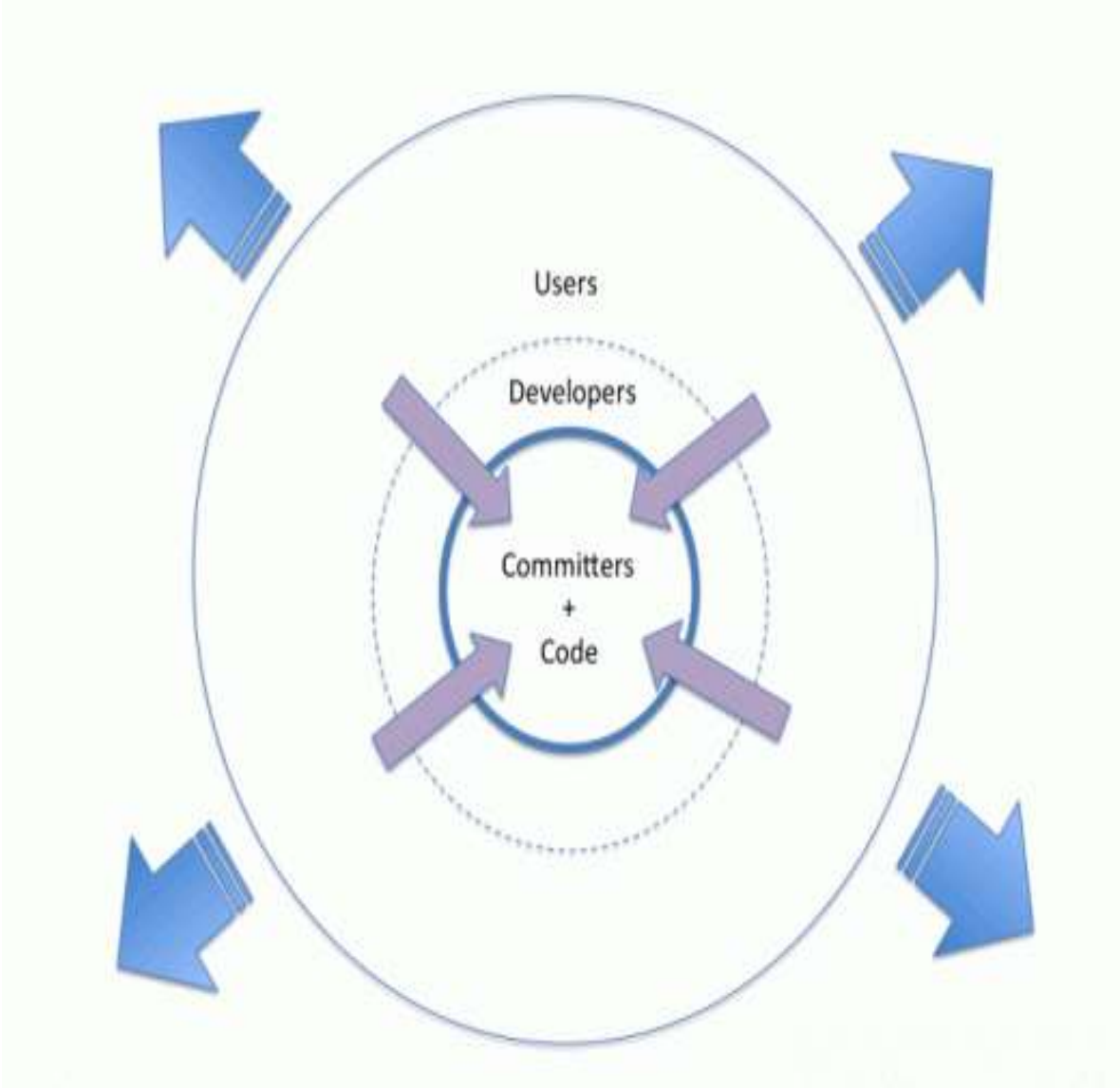
Since we prefer open source software development to build high-quality solutions at an affordable cost, it has become a hot topic in the development market. The term "open source" itself represents its meaning. It refers to something that is publicly available so that everyone has easy access to share and customize it.

Development Architecture of our project



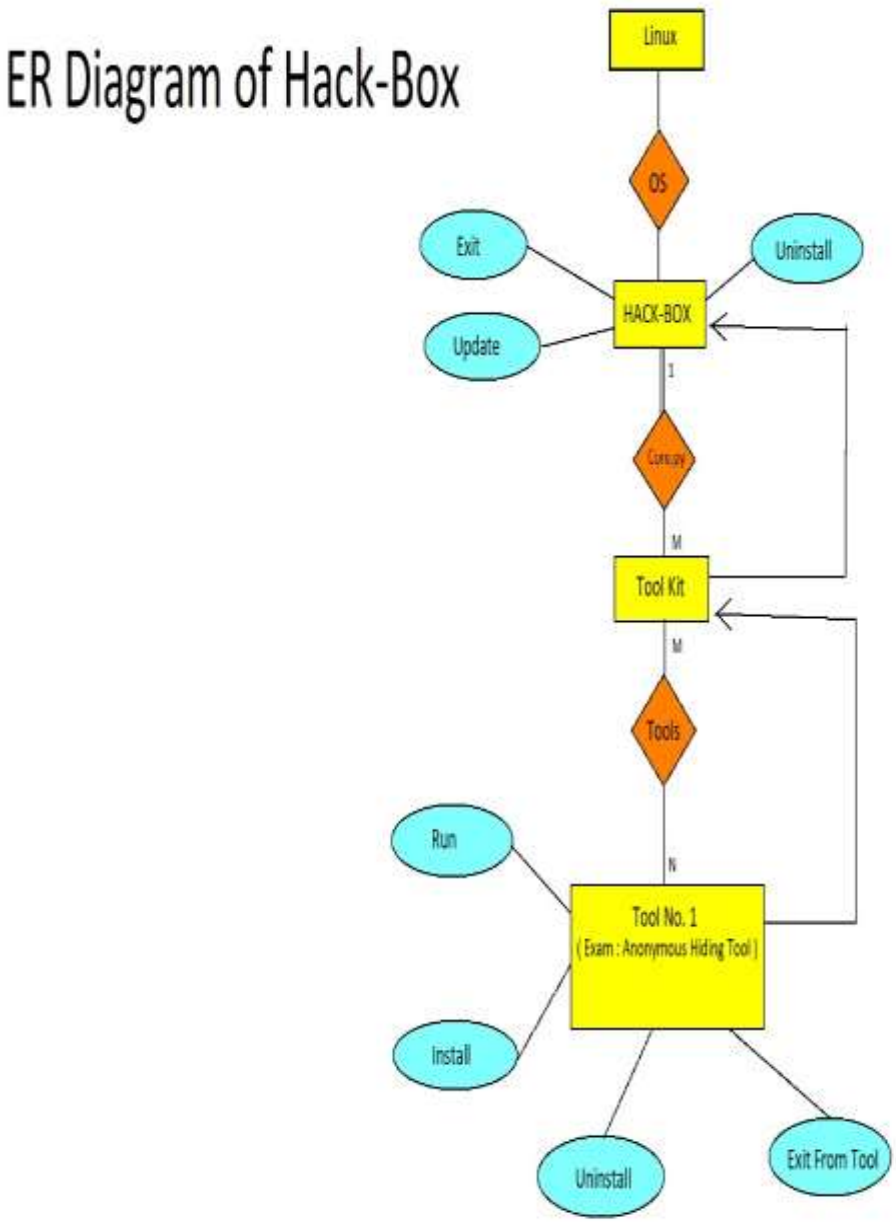
Patterns

Our project works on Contribution flow pattern. A free or open source software project is at its simplest a discussion in software, and without contributions the conversation fades and fails. From a more complex community perspective, a FOSS project is about the economics of collaborative innovation and development. Without a continuous contribution flow, the dynamic aspect of a software project will become static and brittle and lose its relevancy.



Modelling & Diagrams

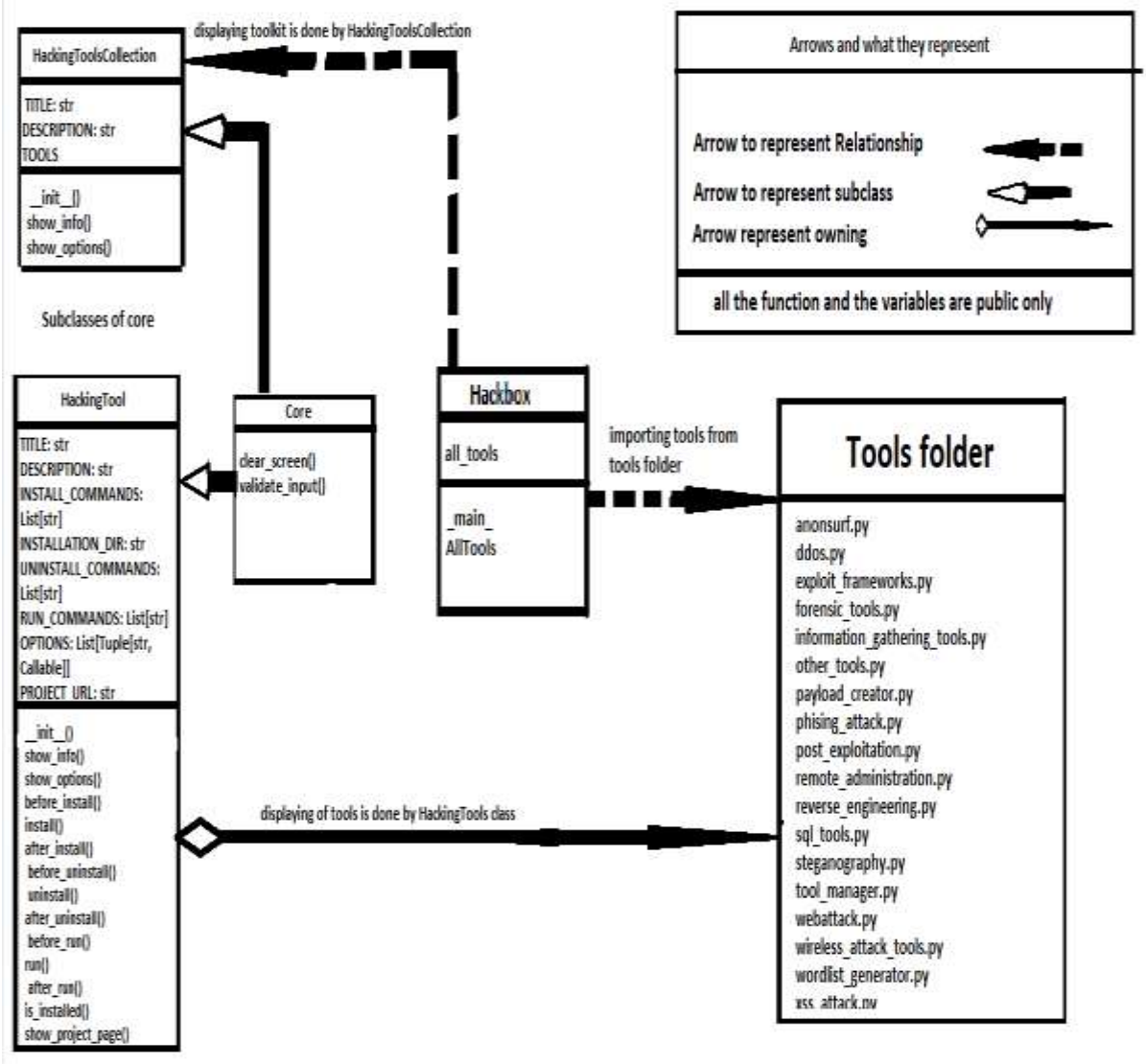
E-R diagram



UML Diagrams

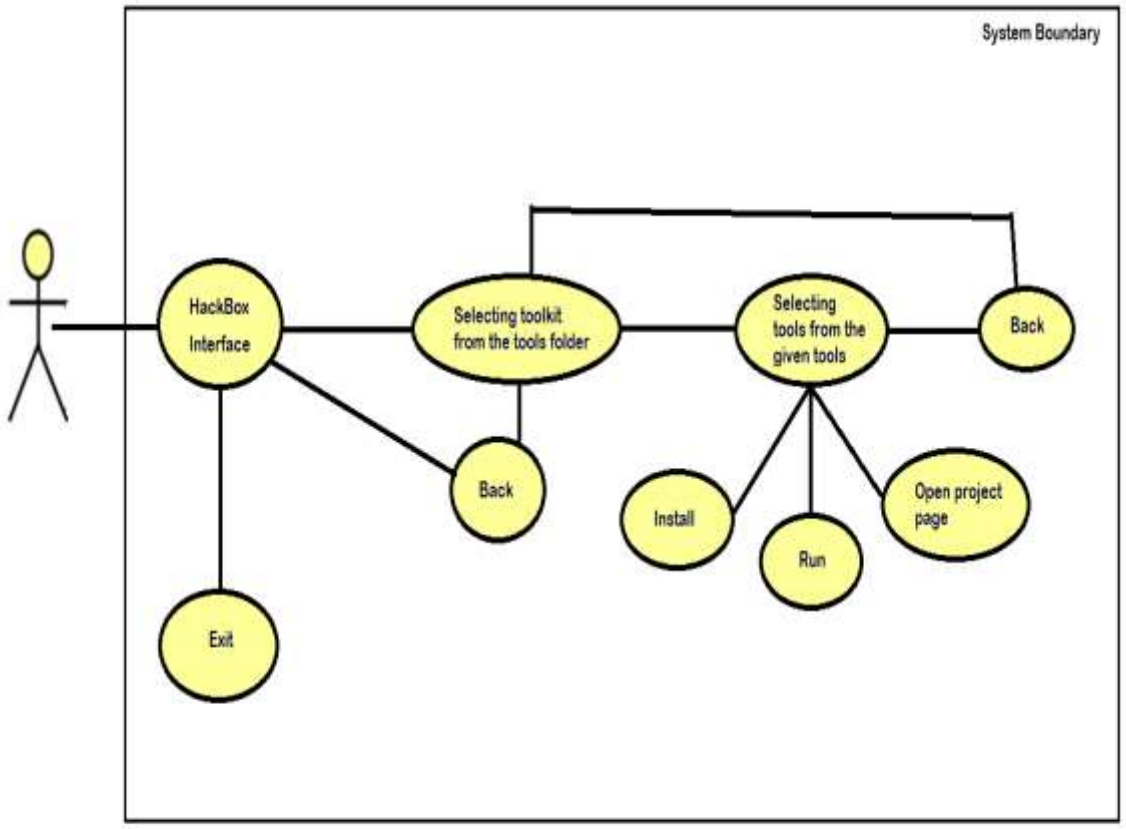
1) Class Diagrams: -

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



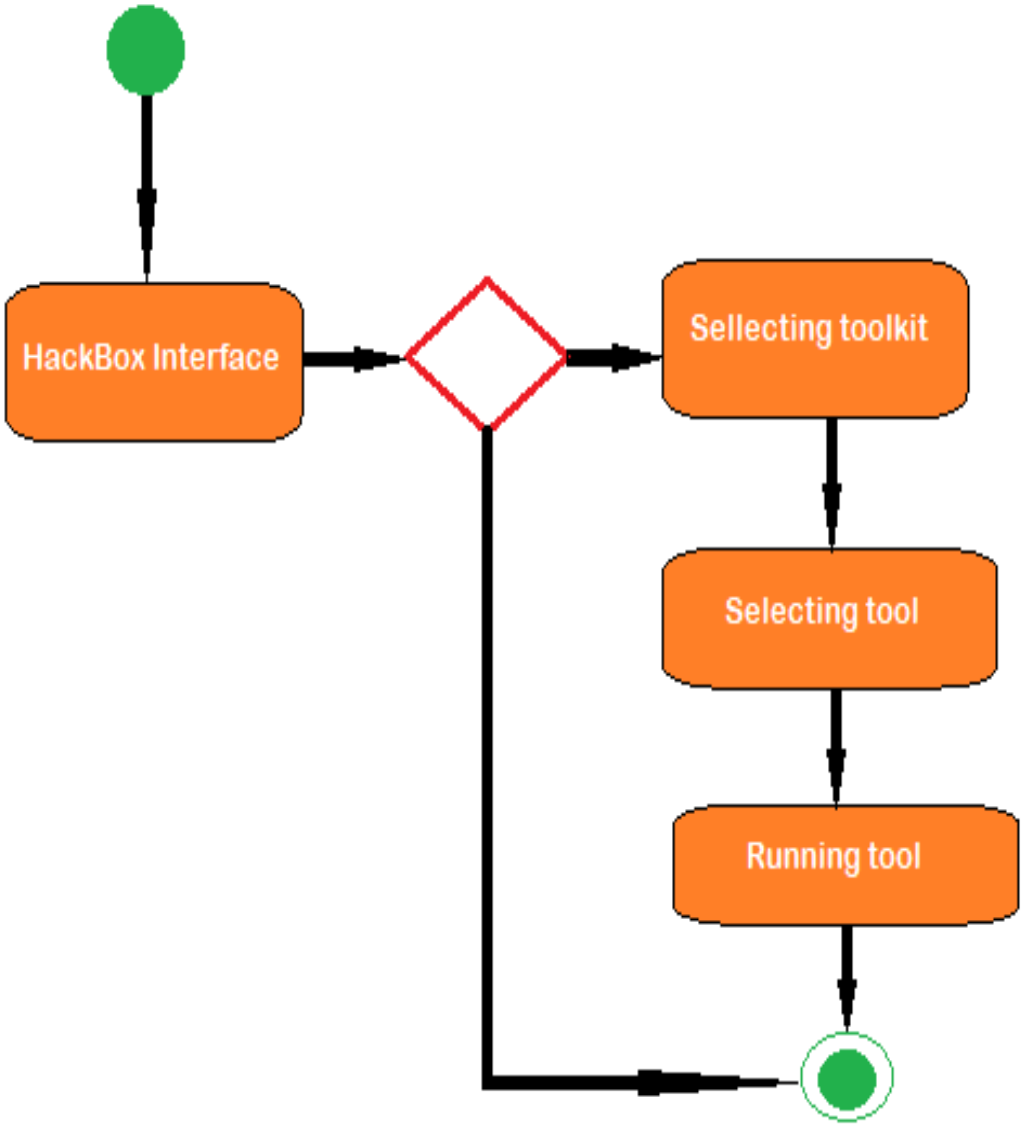
2) Use- Case diagram: -

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.



Activity diagram: -

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).



Implementation Phase

Languages used characteristics

Python:-

1. Presence of third-party modules
2. Extensive support libraries (NumPy for numerical calculations, Pandas for data analytics etc)
3. Open source and community development
4. User-friendly data structures and High-level language
5. Dynamically typed language(No need to mention data type based on the value assigned, it takes data type)
6. Object-oriented language
7. Portable and Interactive
8. Ideal for prototypes – provide more functionality with less coding and portable across Operating systems

Shell:-

1. To automate the frequently performed operations
2. To run sequence of commands as a single command
3. Easy to use
4. Portable (It can be executed in any Unix-like operating systems without any modifications)

Coding

The coding and implementation of our project has been completed. Everything is ready for demonstration. The status of our project is as follows:

1. Install.sh to install the HackBox in the system.
2. HackBox.py for displaying the starting page and all the toolkits we are offering.
3. Core.py for designing the core design for all the tools collections and tools.
    - HackingToolsCollection class displays the content under all different tools.
    - HackingTool class display the content under different tools under different toolkits.
4. Update.sh to update the latest tools under the HackBox.
5. Requirements.txt to install all the required software's for proper functioning
6. Tools folder contain the scripts for all the toolkit.

Code

Install.sh

```
clear

BLACK='\e[30m'
RED='\e[31m'
GREEN='\e[92m'
YELLOW='\e[33m'
ORANGE='\e[93m'
BLUE='\e[34m'
PURPLE='\e[35m'
CYAN='\e[36m'
WHITE='\e[37m'
NC='\e[0m'
purpal='\033[35m'

echo -e "${RED} "
echo ""
echo "
echo "
echo "
echo "
echo "
echo "
echo "

echo -e "${BLUE}                              Created by team Hack Box  ${NC}"

echo -e "${ORANGE}                               [!] This Tool Must Run As ROOT [!]${NC}"
echo ""
echo -e ${CYAN}             "Select Best Option : "
echo ""
echo -e "${WHITE}              [1] Kali Linux / Parrot-Os "
echo -e "${WHITE}              [0] Exit "
echo -n -e "HackBox >> "
read choice
INSTALL_DIR="/usr/share/doc/hackbox"
BIN_DIR="/usr/bin/"
if [ $choice == 1 ]; then
        echo "[*] Checking Internet Connection .."
        wget -q --tries=10 --timeout=20 --spider https://google.com
        if [[ $? -eq 0 ]]; then
            echo -e ${BLUE}"[✓] Loading ... "
            sudo apt-get update && apt-get upgrade
            sudo apt-get install python-pip
            echo "[✓] Checking directories..."
            if [ -d "$INSTALL_DIR" ]; then
                echo "[!] A Directory hackbox Was Found.. Do You Want To Replace It ? [y/n]:" ;
                read input
                if [ "$input" = "y" ]; then
                    rm -R "$INSTALL_DIR"
                else
                    exit
                fi
            fi
                echo "[✓] Installing ...";
                echo "";
                git clone https://github.com/Z4nzu/hackbox.git "$INSTALL_DIR";
                echo "#!/bin/bash
                python3 $INSTALL_DIR/hackbox.py" '${1+"$@"}' > hackbox;
                sudo chmod +x hackbox;
                sudo cp hackbox /usr/bin/;
                rm hackbox;
                echo "";
                echo "[✓] Trying to installing Requirements ..."
                sudo pip3 install lolcat
                sudo apt-get install -y figlet
                sudo pip3 install boxes
                sudo apt-get install boxes
                sudo pip3 install flask
                sudo pip3 install requests
        else
                echo -e $RED "Please Check Your Internet Connection ..!!"
        fi
```

```
    if [ -d "$INSTALL_DIR" ]; then
        echo "";
        echo "[✓] Successfuly Installed !!! ";
        echo "";
        echo "";
        echo -e $ORANGE "              [+]+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[+]"
        echo            "              [+]                                                         [+]"
        echo -e $ORANGE  "             [+]    ✓✓✓ Now Just Type In Terminal (hackbox) ✓✓✓  [+]"
        echo            "              [+]                                                         [+]"
        echo -e $ORANGE "              [+]+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++[+]"
    else
        echo "[✗] Installation Failed !!! [✗]";
        exit
    fi
elif [ $choice -eq 0 ];
then
    echo -e $RED "[✗] THank YOu !! [✗] "
    exit
else
    echo -e $RED "[!] Select Valid Option [!]"
fi
```

Hackbox.py

```
import os
import webbrowser
from platform import system
from time import sleep

from core import HackingToolsCollection
from tools.anonsurf import AnonSurfTools
from tools.ddos import DDOSTools
from tools.exploit_frameworks import ExploitFrameworkTools
from tools.forensic_tools import ForensicTools
from tools.information_gathering_tools import InformationGatheringTools
from tools.other_tools import OtherTools
from tools.payload_creator import PayloadCreatorTools
from tools.phising_attack import PhishingAttackTools
from tools.post_exploitation import PostExploitationTools
from tools.remote_administration import RemoteAdministrationTools
from tools.reverse_engineering import ReverseEngineeringTools
from tools.sql_tools import SqlInjectionTools
from tools.steganography import SteganographyTools
from tools.tool_manager import ToolManager
from tools.webattack import WebAttackTools
from tools.wireless_attack_tools import WirelessAttackTools
from tools.wordlist_generator import WordlistGeneratorTools
from tools.xss_attack import XSSAttackTools

logo = """\033[31m
```

```
                                \033[34m[✓]     Created by team Hack Box          [✓]
                                \033[34m[✓]           Version 1.1.0               [✓]
                                \033[91m[X] Please Don't Use For illegal Activity  [X]
\033[97m """

all_tools = [
    AnonSurfTools(),
    InformationGatheringTools(),
    WordlistGeneratorTools(),
    WirelessAttackTools(),
    SqlInjectionTools(),
    PhishingAttackTools(),
    WebAttackTools(),
    PostExploitationTools(),
    ForensicTools(),
    PayloadCreatorTools(),
    ExploitFrameworkTools(),
    ReverseEngineeringTools(),
    DDOSTools(),
    RemoteAdministrationTools(),
    XSSAttackTools(),
    SteganographyTools(),
    OtherTools(),
    ToolManager()
]


class AllTools(HackingToolsCollection):
    TITLE = "All tools"
    TOOLS = all_tools

    def show_info(self):
        print(logo + '\033[0m \033[97m')

if __name__ == "__main__":
    try:
        if system() == 'Linux':
            fpath = "/home/hackingtoolpath.txt"
            if not os.path.exists(fpath):
                os.system('clear')
                # run.menu()
                print("""
                        [@] Set Path (All your tools will be installed in that directory)
                        [1] Manual
                        [2] Default
                """)
                choice = input("HackBox =>> ")

                if choice == "1":
                    inpath = input("Enter Path (with Directory Name) >> ")
                    with open(fpath, "w") as f:
                        f.write(inpath)
                    print(f"Successfully Set Path to: {inpath}")
                elif choice == "2":
                    autopath = "/home/hackingtool/"
                    with open(fpath, "w") as f:
                        f.write(autopath)
                    print(f"Your Default Path Is: {autopath}")
                    sleep(3)
                else:
                    print("Try Again..!!")
                    exit(0)

            with open(fpath) as f:
                archive = f.readline()
                if not os.path.exists(archive):
                    os.mkdir(archive)
                os.chdir(archive)
                all_tools = AllTools()
```

```
            all_tools.show_options()

        # If not Linux and probably Windows
        elif system() == "Windows":
            print(
                "\033[91m Please Run This Tool On A Debian System For Best Results " "\e[00m")
            sleep(2)
            webbrowser.open_new_tab("https://tinyurl.com/y522modc")

        else:
            print("Please Check Your System or Open New Issue ...")

    except KeyboardInterrupt:
        print("\nExiting ..!!!")
        sleep(2)
```

core.py

```
import os
import sys
import webbrowser
from platform import system
from traceback import print_exc
from typing import Any
from typing import Callable
from typing import List
from typing import Tuple


def clear_screen():
    if system() == "Linux":
        os.system("clear")
    if system() == "Windows":
        os.system("cls")


def validate_input(ip, val_range):
    try:
        ip = int(ip)
        if ip in val_range:
            return ip
        else:
            return None
    except:
        return None


class HackingTool(object):
    # About the HackingTool
    TITLE: str = ""  # used to show info in the menu
    DESCRIPTION: str = ""

    INSTALL_COMMANDS: List[str] = []
```

```
    INSTALLATION_DIR: str = ""

    UNINSTALL_COMMANDS: List[str] = []

    RUN_COMMANDS: List[str] = []

    OPTIONS: List[Tuple[str, Callable]] = []

    PROJECT_URL: str = ""

    def __init__(self, options = None, installable: bool = True,
                 runnable: bool = True):
        if options is None:
            options = []
        if isinstance(options, list):
            self.OPTIONS = []
            if installable:
                self.OPTIONS.append(('Install', self.install))
            if runnable:
                self.OPTIONS.append(('Run', self.run))
            self.OPTIONS.extend(options)
        else:
            raise Exception(
                "options must be a list of (option_name, option_fn) tuples")

    def show_info(self):
        desc = self.DESCRIPTION
        if self.PROJECT_URL:
            desc += '\n\t[*] '
            desc += self.PROJECT_URL
        os.system(f'echo "{desc}"|boxes -d boy | lolcat')

    def show_options(self, parent = None):
        clear_screen()
        self.show_info()
        for index, option in enumerate(self.OPTIONS):
            print(f"[{index + 1}] {option[0]}")
        if self.PROJECT_URL:
            print(f"[{98}] Open project page")
        print(f"[{99}] Back to {parent.TITLE if parent is not None else 'Exit'}")
        option_index = input("Select an option : ")
        try:
            option_index = int(option_index)
            if option_index - 1 in range(len(self.OPTIONS)):
                ret_code = self.OPTIONS[option_index - 1][1]()
                if ret_code != 99:
                    input("\n\nPress ENTER to continue:")
            elif option_index == 98:
                self.show_project_page()
            elif option_index == 99:
                if parent is None:
                    sys.exit()
                return 99
        except (TypeError, ValueError):
            print("Please enter a valid option")
            input("\n\nPress ENTER to continue:")
        except Exception:
            print_exc()
            input("\n\nPress ENTER to continue:")
        return self.show_options(parent = parent)

    def before_install(self):
        pass

    def install(self):
        self.before_install()
        if isinstance(self.INSTALL_COMMANDS, (list, tuple)):
            for INSTALL_COMMAND in self.INSTALL_COMMANDS:
                os.system(INSTALL_COMMAND)
            self.after_install()
```

```
    def after_install(self):
        print("Successfully installed!")

    def before_uninstall(self) -> bool:
        """ Ask for confirmation from the user and return """
        return True

    def uninstall(self):
        if self.before_uninstall():
            if isinstance(self.UNINSTALL_COMMANDS, (list, tuple)):
                for UNINSTALL_COMMAND in self.UNINSTALL_COMMANDS:
                    os.system(UNINSTALL_COMMAND)
            self.after_uninstall()

    def after_uninstall(self):
        pass

    def before_run(self):
        pass

    def run(self):
        self.before_run()
        if isinstance(self.RUN_COMMANDS, (list, tuple)):
            for RUN_COMMAND in self.RUN_COMMANDS:
                os.system(RUN_COMMAND)
            self.after_run()

    def after_run(self):
        pass

    def is_installed(self, dir_to_check = None):
        print("Unimplemented: DO NOT USE")
        return "?"
    def show_project_page(self):
        webbrowser.open_new_tab(self.PROJECT_URL)


class HackingToolsCollection(object):
    TITLE: str = ""  # used to show info in the menu
    DESCRIPTION: str = ""
    TOOLS = []  # type: List[Any[HackingTool, HackingToolsCollection]]

    def __init__(self):
        pass

    def show_info(self):
        os.system("figlet -f standard -c {} | lolcat".format(self.TITLE))
        # os.system(f'echo "{self.DESCRIPTION}"|boxes -d boy | lolcat')
        # print(self.DESCRIPTION)

    def show_options(self, parent = None):
        clear_screen()
        self.show_info()
        for index, tool in enumerate(self.TOOLS):
            print(f"[{index} {tool.TITLE}")
        print(f"[{99}] Back to {parent.TITLE if parent is not None else 'Exit'}")
        tool_index = input("Choose a tool to proceed: ")
        try:
            tool_index = int(tool_index)
            if tool_index in range(len(self.TOOLS)):
                ret_code = self.TOOLS[tool_index].show_options(parent = self)
                if ret_code != 99:
                    input("\n\nPress ENTER to continue:")
            elif tool_index == 99:
                if parent is None:
                    sys.exit()
                return 99
```

```
except (TypeError, ValueError):
    print("Please enter a valid option")
    input("\n\nPress ENTER to continue:")
except Exception as e:
    print_exc()
    input("\n\nPress ENTER to continue:")
return self.show_options(parent = parent)
```

Requirement.txt

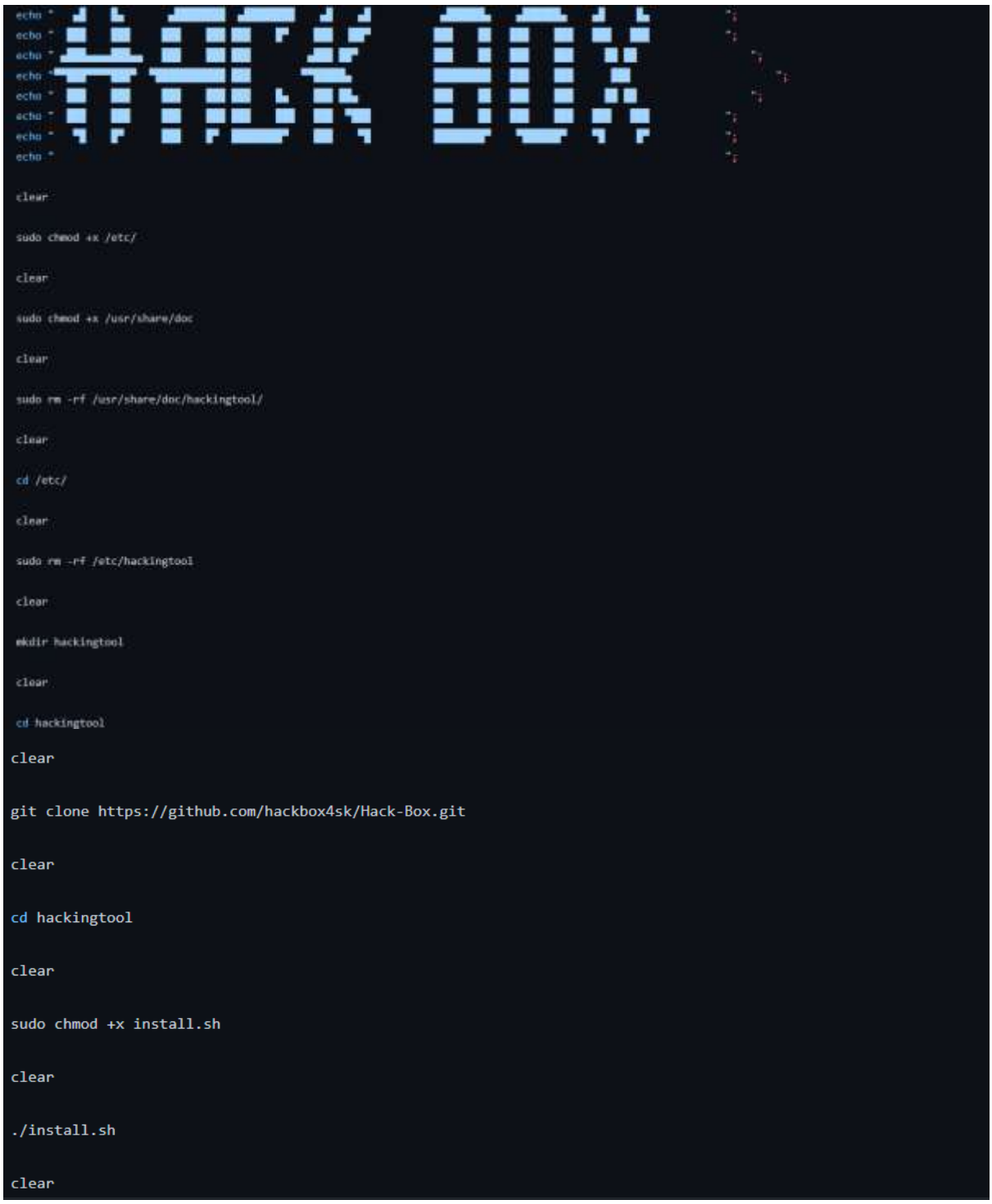
```
lolcat

boxes

flask

requests
```

Update.sh



```
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
clear

sudo chmod +x /etc/

clear

sudo chmod +x /usr/share/doc

clear

sudo rm -rf /usr/share/doc/hackingtool/

clear

cd /etc/

clear

sudo rm -rf /etc/hackingtool

clear

mkdir hackingtool

clear

cd hackingtool
clear

git clone https://github.com/hackbox4sk/Hack-Box.git

clear

cd hackingtool

clear

sudo chmod +x install.sh

clear

./install.sh

clear
```

Testing

Testing Objectives

To test the project for a successful and fail-safe operation with respect to all the feature mentioned even with the increased number of users.

Testing Methods & Strategies

Unit Testing

Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

GUI Testing

The objective of this GUI Testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mock-up screens.

System Testing

Under System Testing technique, the entire system is tested as per the requirements. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.

Security Testing

Security Testing is done to check how the software or application is secure from internal and external threats. This testing includes how much software is secure from the malicious program viruses and how secure and strong the authorization and authentication processes.

White Box Testing

It is also known as Glass box Testing. Internal software and code working should be known for performing this type of testing. Under these tests are based on the coverage of code statements, branches, paths, conditions, etc.

Black Box Testing

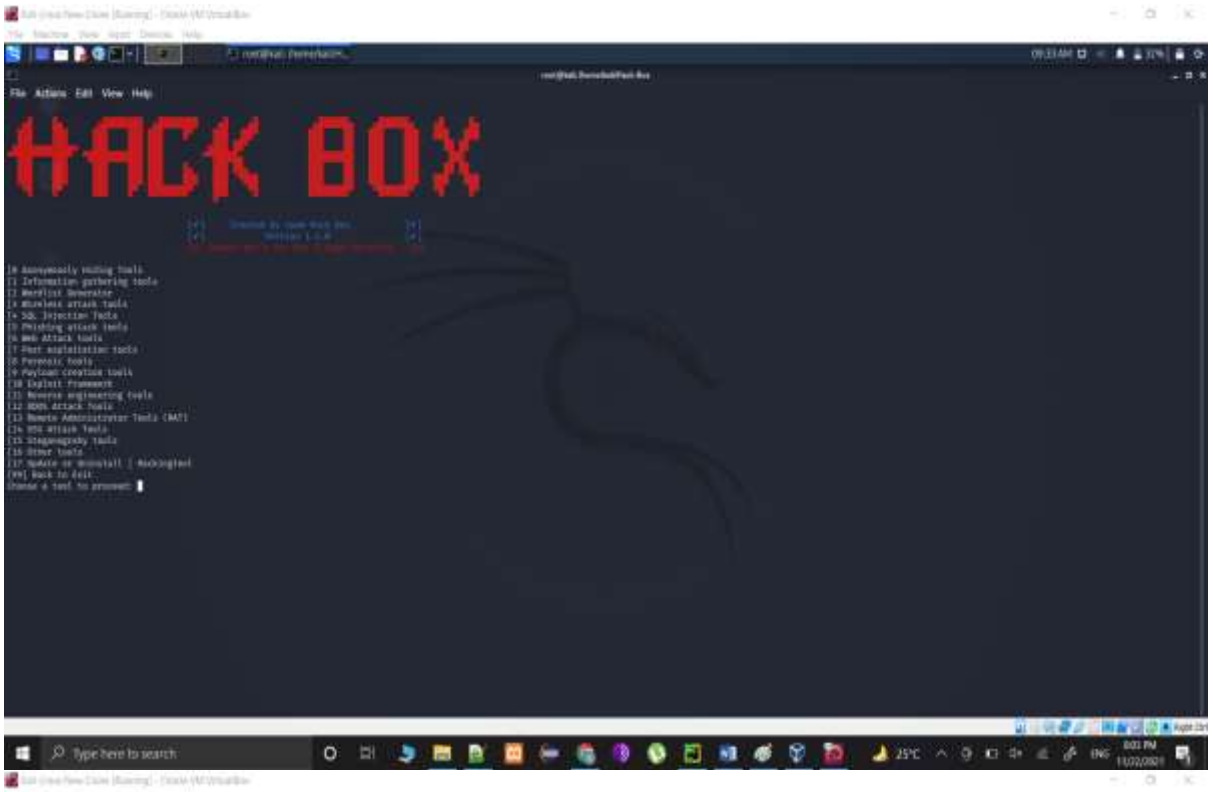
Black Box Testing, also known as Behavioural Testing, is a software testing method in which the structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

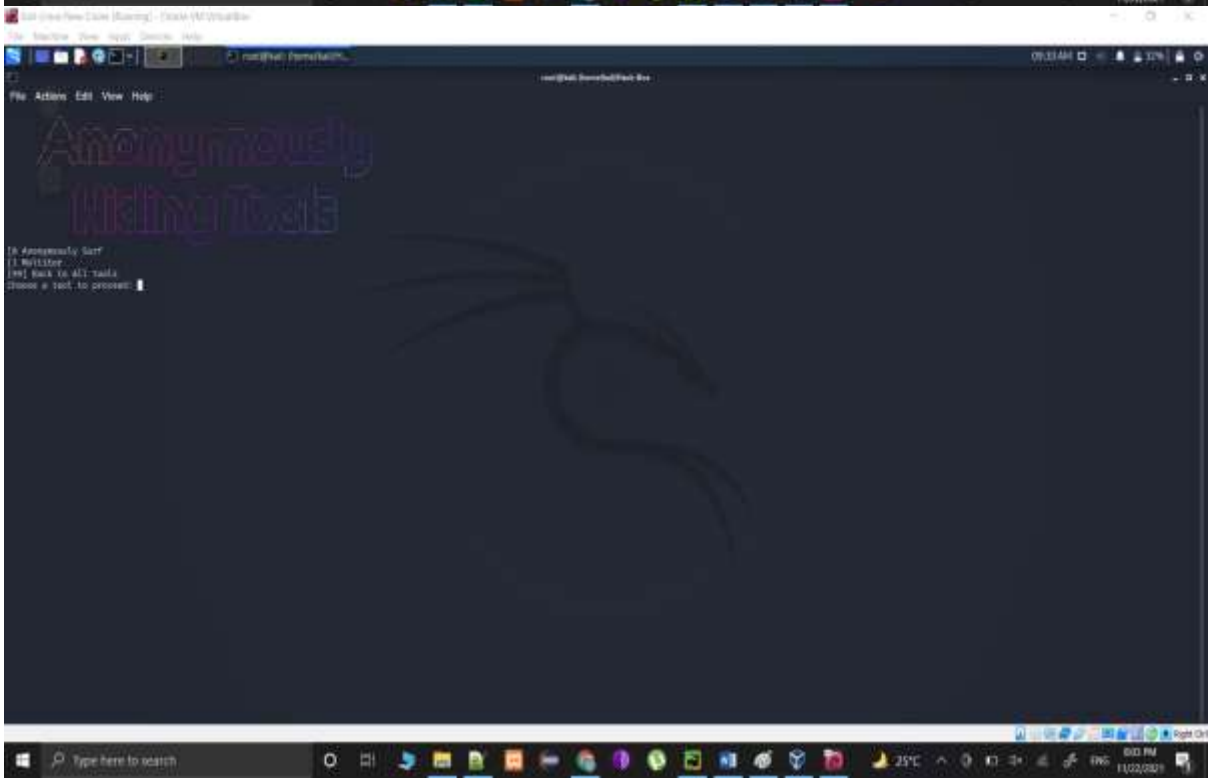
Conclusion: -

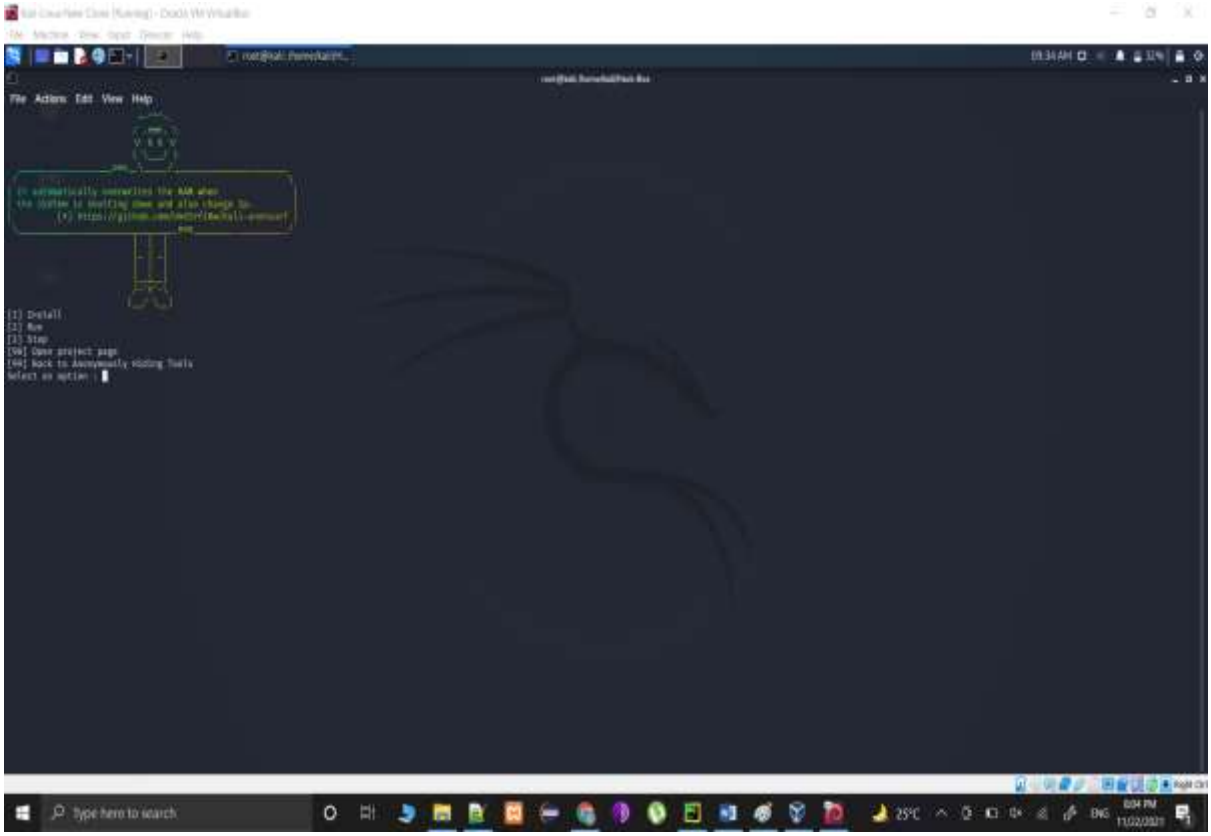
We as a group of ethical hackers recommend HACK BOX for its customizability, user friendly UI, simplicity and being a beginner's friendly complete hacking tool kit.

HackBox demo Images

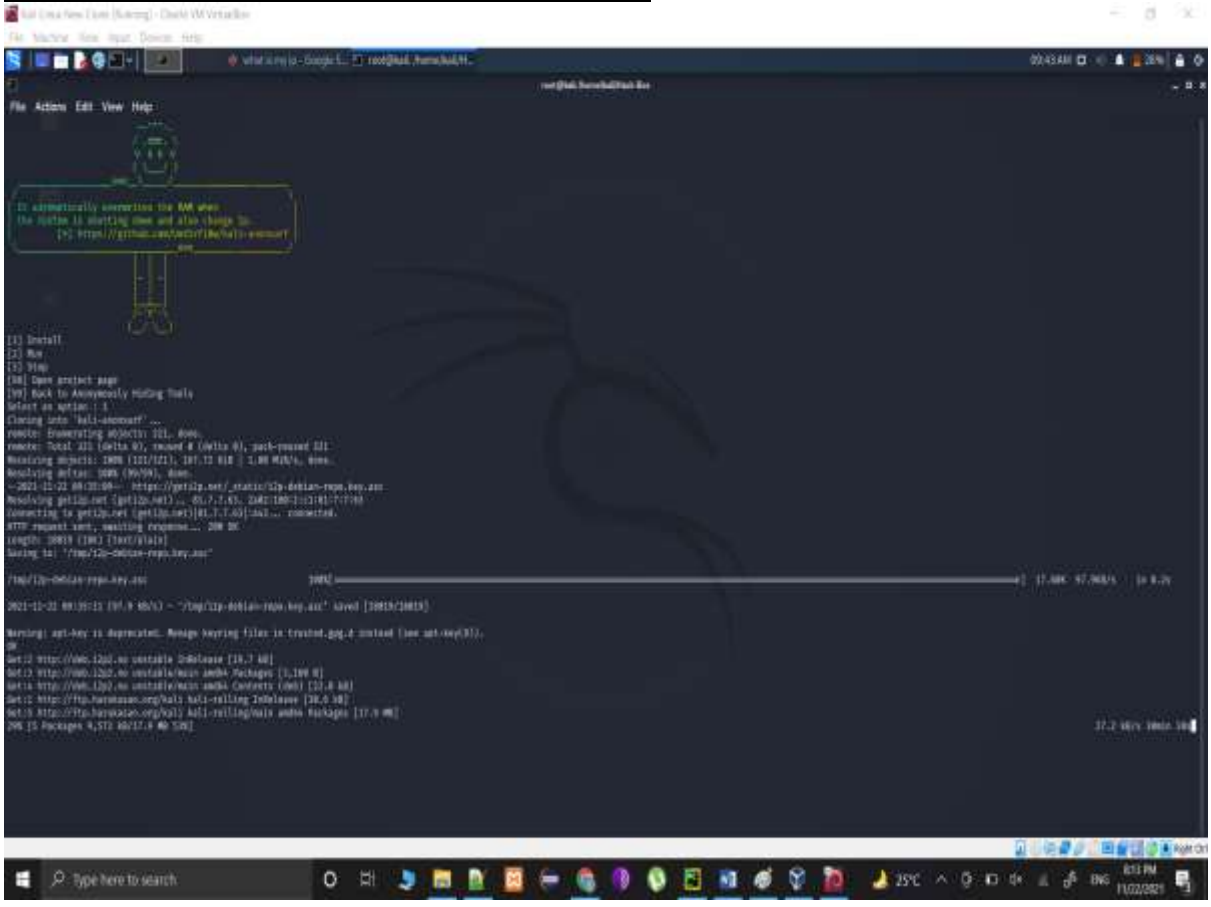
(Example of using Anonymously Surf tool under Anonymously Hiding Tools toolkit)



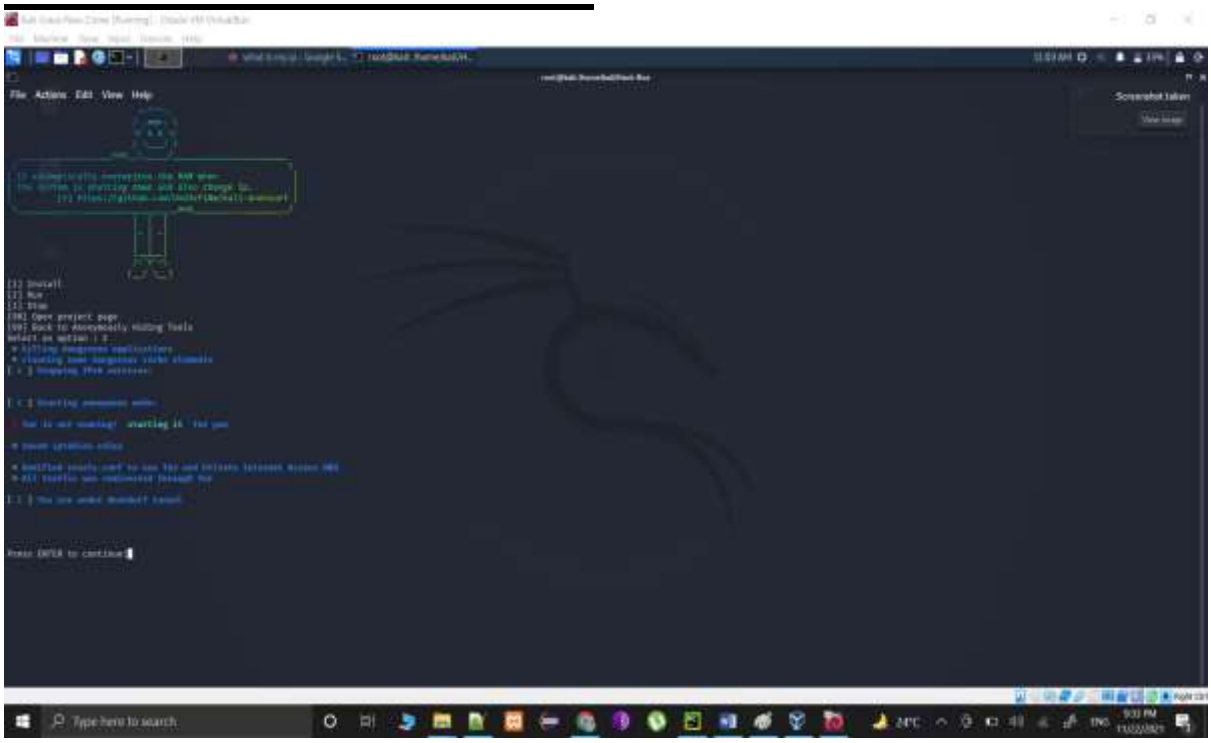




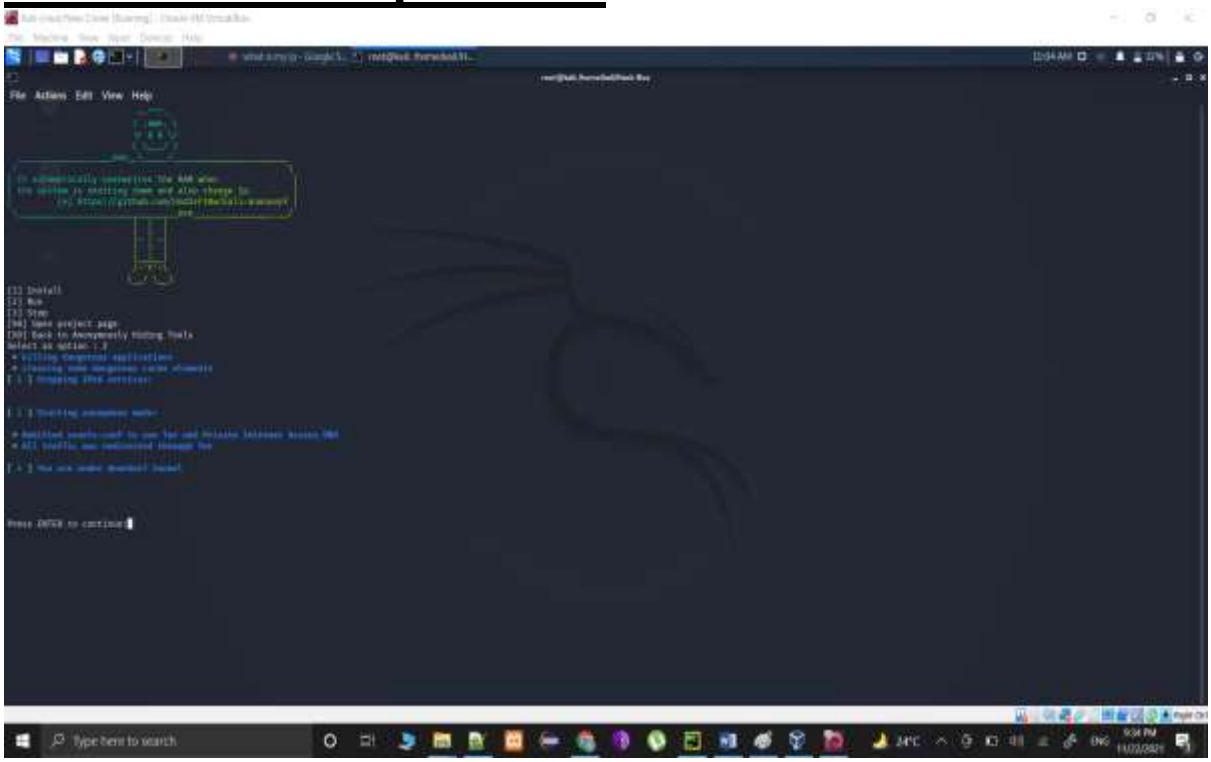
If we chose to install the tool



If we chose to run the tool



If we chose to stop the tool



Similarly the other tools in the various toolkits can be Installed and used easily.