Air University Islamabad, Main Campus.

National Cyber Security Academy (NCSA)

Department of Cyber Security

Course name:

Programming Fundamentals

Semester Project

Submitted by:

Aqsa Khaliq

Areeba Shafiq

Khadija Zia

DARK LITE TOOLKIT

Table Of Contents

DARK LITE – A Command-Line Cybersecurity Simulation Toolkit

1. Introduction

In today's digital era, cybersecurity has become a fundamental requirement due to the rapid increase in cyber threats such as phishing attacks, weak authentication practices, and unauthorized system access. To understand these threats practically, it is essential to simulate and analyze common attack techniques in a controlled environment.

Dark Lite is a command-line based cybersecurity tool developed using **C++** that demonstrates core security concepts such as password strength, brute-force attack simulation, phishing detection, and basic firewall logic. The project aims to provide students with hands-on exposure to cybersecurity principles while strengthening their understanding of programming concepts like file handling, conditional logic, and input validation.

The toolkit features an interactive ASCII art banner and a menu-driven interface that makes cybersecurity concepts accessible to beginners while maintaining technical depth for educational purposes.

2. Problem Statement

With the growth of online systems, several security issues continue to affect users:

1. Weak or reused passwords increase vulnerability to attacks.
2. Phishing websites deceive users into revealing sensitive information.
3. Open or poorly monitored ports expose systems to unauthorized access.
4. Inadequate firewall rules allow malicious traffic to enter systems.
5. Beginners lack safe environments to understand cybersecurity attacks and defense mechanisms.

This project addresses these issues by offering a simplified, educational simulation of real-world cybersecurity scenarios.

3. Proposed Solution

The **Dark Lite Toolkit** provides a menu-driven system that allows users to interact with multiple cybersecurity modules. Each module demonstrates a specific concept in a safe and controlled way.

The system includes:

- Secure password generation
- Brute-force attack simulation
- Phishing URL detection
- Port scanning simulation
- Firewall rule validation
- Activity logging and summary generation

This approach helps users understand both attack techniques and defensive strategies.

4. Objectives

- To design a command-line cybersecurity application using C++
- To simulate common cyberattacks in a safe environment
- To understand password security and brute-force mechanisms
- To learn file handling and structured programming
- To promote cybersecurity awareness among beginners

5. Tools and Technologies Used

Component	**Description**
Programming Language	C++
Compiler	JDoodle (Online C++ Compiler)
IDE	JDoodle (Online IDE)

Component	Description
Libraries Used	<iostream>, <fstream>, <string>, <ctime>, <cstdlib>, <limits>
Platform	Web / Online
Operating System	Windows (Command Line)

6. System Design Overview

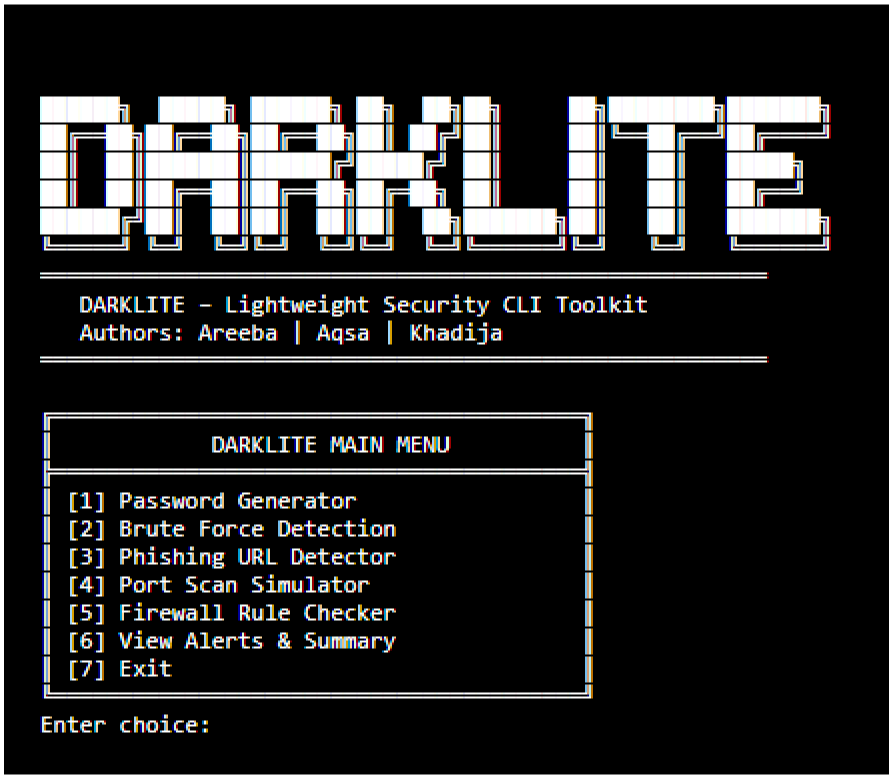
The application follows a menu-driven architecture with modular function design. Each menu option triggers a specific security function that operates independently while sharing common data structures for logging.

6.1 Architecture Components

1. **User Interface Layer**: ASCII banner and menu system
2. **Functional Modules**: Independent security simulation functions
3. **Data Storage Layer**: File-based logging system
4. **Alert Management**: In-memory alert tracking using structures and arrays

6.2 Basic Flow

1. User launches the application
2. ASCII banner and welcome message displayed
3. Main menu is presented with 7 options
4. User selects an operation (1-7)
5. Corresponding function executes with input validation
6. Output is displayed to console
7. Results are logged to appropriate text files
8. Alert counters are updated
9. Program returns to menu (unless exit selected)
10. Process repeats until user chooses to exit

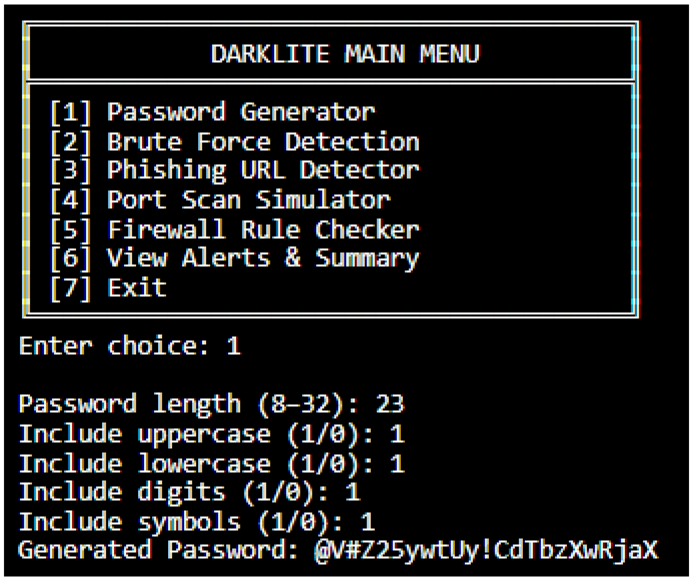


```
DARKLITE - Lightweight Security CLI Toolkit
Authors: Areeba | Aqsa | Khadija

DARKLITE MAIN MENU

[1] Password Generator
[2] Brute Force Detection
[3] Phishing URL Detector
[4] Port Scan Simulator
[5] Firewall Rule Checker
[6] View Alerts & Summary
[7] Exit

Enter choice:
```

7. Functional Modules

7.1. Password Generator

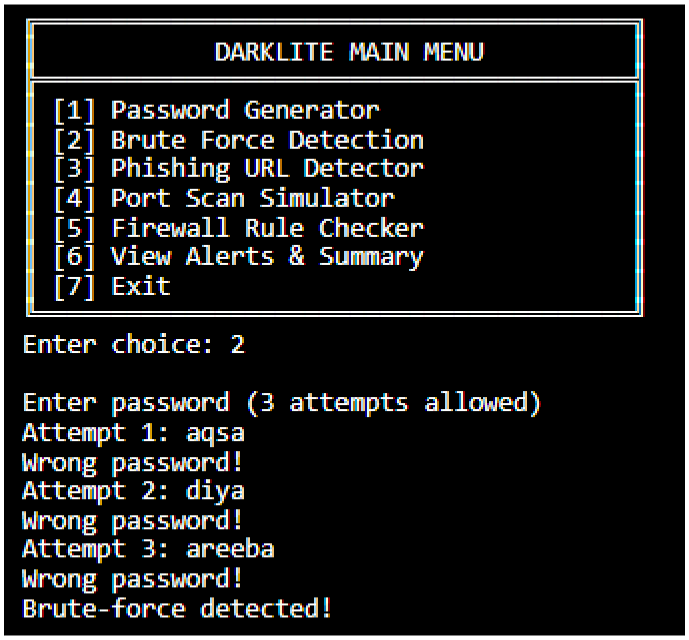
- Generates strong random passwords
- Allows user-defined length and character selection
- Stores generated passwords in memory
- Demonstrates secure password creation



```
DARKLITE MAIN MENU

[1] Password Generator
[2] Brute Force Detection
[3] Phishing URL Detector
[4] Port Scan Simulator
[5] Firewall Rule Checker
[6] View Alerts & Summary
[7] Exit

Enter choice: 1

Password length (8-32): 23
Include uppercase (1/0): 1
Include lowercase (1/0): 1
Include digits (1/0): 1
Include symbols (1/0): 1
Generated Password: @V#Z25ywtUy!CdTbzXwRjaX
```

7.2. Brute Force Attack Simulation

- Simulates login attempts using predefined credentials
- Limits number of attempts
- Detects and reports brute-force behavior
- Logs results for analysis

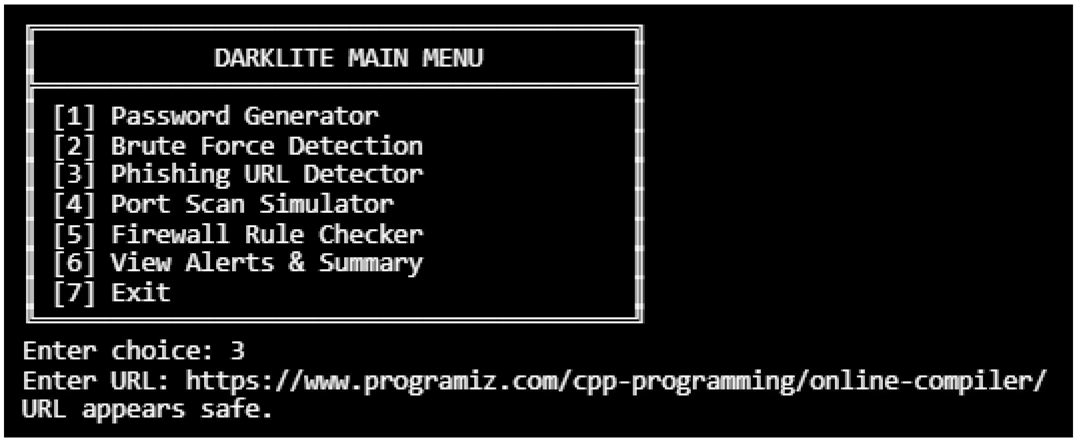


```
DARKLITE MAIN MENU

[1] Password Generator
[2] Brute Force Detection
[3] Phishing URL Detector
[4] Port Scan Simulator
[5] Firewall Rule Checker
[6] View Alerts & Summary
[7] Exit

Enter choice: 2

Enter password (3 attempts allowed)
Attempt 1: aqsa
Wrong password!
Attempt 2: diya
Wrong password!
Attempt 3: areeba
Wrong password!
Brute-force detected!
```

7.3. Phishing URL Detection

- Accepts a URL from the user

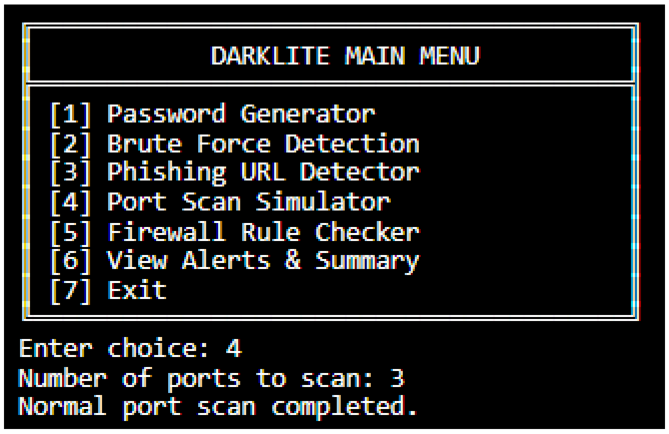
- Checks for suspicious keywords such as *login*, *verify*, *secure*
- Flags potentially malicious URLs
- Stores detection results



```
╔══════════════════════════════════════╗
║           DARKLITE MAIN MENU         ║
╠══════════════════════════════════════╣
║ [1] Password Generator               ║
║ [2] Brute Force Detection            ║
║ [3] Phishing URL Detector            ║
║ [4] Port Scan Simulator              ║
║ [5] Firewall Rule Checker            ║
║ [6] View Alerts & Summary            ║
║ [7] Exit                             ║
╚══════════════════════════════════════╝
Enter choice: 3
Enter URL: https://www.programiz.com/cpp-programming/online-compiler/
URL appears safe.
```

7.4. Port Scanner

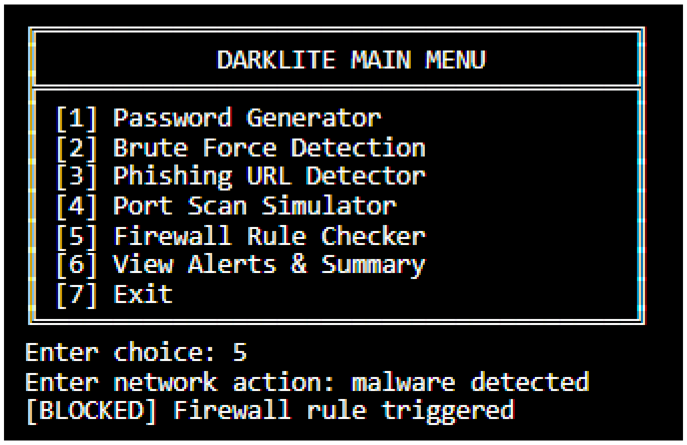
- Simulates scanning of multiple ports
- Detects unusually high scanning behavior
- Helps users understand open-port risks



```
╔══════════════════════════════════════╗
║           DARKLITE MAIN MENU         ║
╠══════════════════════════════════════╣
║ [1] Password Generator               ║
║ [2] Brute Force Detection            ║
║ [3] Phishing URL Detector            ║
║ [4] Port Scan Simulator              ║
║ [5] Firewall Rule Checker            ║
║ [6] View Alerts & Summary            ║
║ [7] Exit                             ║
╚══════════════════════════════════════╝
Enter choice: 4
Number of ports to scan: 3
Normal port scan completed.
```

7.5. Firewall Rule Checker

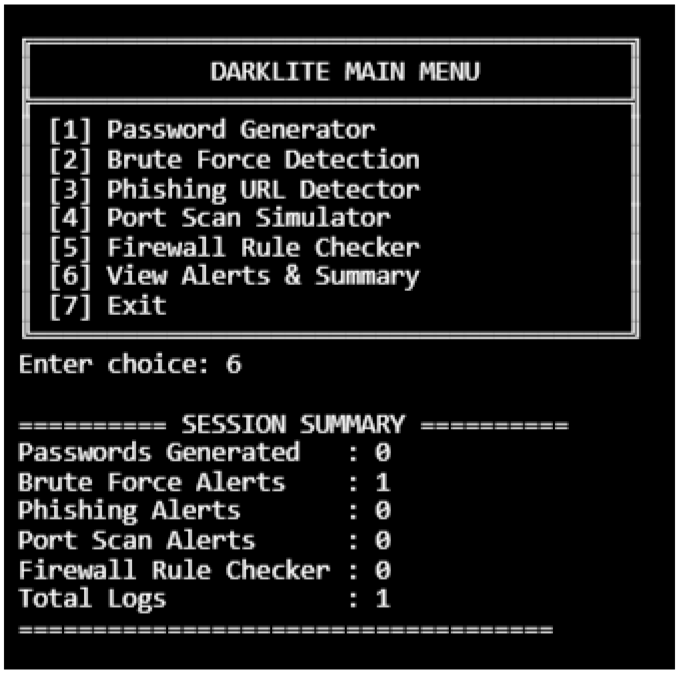
- Analyzes network input strings
- Blocks traffic containing suspicious patterns
- Simulates firewall behavior using conditional rules



```
DARKLITE MAIN MENU

[1] Password Generator
[2] Brute Force Detection
[3] Phishing URL Detector
[4] Port Scan Simulator
[5] Firewall Rule Checker
[6] View Alerts & Summary
[7] Exit

Enter choice: 5
Enter network action: malware detected
[BLOCKED] Firewall rule triggered
```

7.6. Summary and Logging System

- Displays total activities performed
- Shows number of alerts generated
- Helps users review session behavior



```
DARKLITE MAIN MENU

[1] Password Generator
[2] Brute Force Detection
[3] Phishing URL Detector
[4] Port Scan Simulator
[5] Firewall Rule Checker
[6] View Alerts & Summary
[7] Exit

Enter choice: 6

========== SESSION SUMMARY ==========
Passwords Generated    : 0
Brute Force Alerts     : 1
Phishing Alerts        : 0
Port Scan Alerts       : 0
Firewall Rule Checker  : 0
Total Logs             : 1
=====================================
```

8. Files Generated

File Name	Description
passwords.txt	Stores generated passwords
brute_force_results.txt	Logs brute-force attempts
phishing_results.txt	Stores detected phishing URLs
port_scan_results.txt	Stores scanned port data
firewall_results.txt	Logs firewall decisions

9. C++ Concepts Used in the Project

The Dark Lite Cybersecurity Toolkit is developed using core C++ programming concepts. These concepts help in building a structured, modular, and efficient application. Each concept plays a specific role in achieving the overall functionality of the system.

9.1 Header Files / Libraries

C++ libraries provide predefined functions and tools that simplify programming tasks. The following libraries are used in this project:

Library	Purpose
<iostream>	Used for input and output operations such as cin and cout.
<fstream>	Enables file handling operations such as reading from and writing to files.
<string>	Allows use of string data type and string manipulation functions.
<ctime>	Provides date and time functions, mainly used for random number seeding.
<cstdlib>	Used for random number generation (rand(), srand()).
<limits>	Helps in clearing input buffer and handling invalid inputs safely.

These libraries together support input/output handling, randomness, file management, and system interaction.

9.2 Functions

Functions are self-contained blocks of code that perform specific tasks. They help make the program modular, reusable, and easy to understand.

Purpose of Using Functions

- Break large programs into smaller manageable parts
- Improve code readability
- Avoid repetition of code
- Simplify debugging and maintenance

Functions Used in the Project

Function Name	Purpose
passwordGenerator()	Generates secure passwords based on user preferences
bruteForceDetection()	Simulates brute-force login attempts
phishingDetector()	Detects suspicious URLs using keyword matching
portScanner()	Simulates port scanning activity
firewallChecker()	Simulates firewall rule checking
viewSummary()	Displays session activity summary
menu()	Displays the main menu
clearInput()	Clears invalid input from buffer

9.3 Structures

Structures (struct) are used to group related data items under a single name.

Purpose of Structures

- Store multiple related data fields together

- Improve data organization
- Make program logic cleaner and more readable

Example Used in Project

```
struct Alert {
   string type;
   string message;
};
```

This structure stores:

- Type of alert (e.g., Phishing, Brute Force)
- Description of the alert

An array of this structure is used to store multiple alert logs.

9.4 Arrays

Arrays are used to store multiple values of the same data type.

Global Arrays Used:

1. **Alert Array:** Alert alerts[MAX_ALERTS] - Stores up to 50 security alerts
2. **Keyword Array**: string keywords[] = {"login", "verify", "secure", "bank"} - Phishing detection patterns

Usage in Project:

- Store multiple alert records
- Store keyword lists for phishing detection
- Maintain a fixed number of logs

Benefits

- Fast access to data
- Efficient memory usage
- Easy iteration using loops

9.5 File Handling

File handling allows the program to store data permanently using files.

Purpose

- Save logs for later review
- Maintain records even after program termination
- Simulate real-world logging systems

Files Used

File Name	Purpose
passwords.txt	Stores generated passwords
brute_force_results.txt	Stores brute-force attempt logs
phishing_results.txt	Stores phishing detection data
port_scan_results.txt	Stores port scanning results
firewall_results.txt	Stores firewall activity

File Operations Used

- ofstream → writing data to files
- ifstream → reading data from files

9.6 Conditional Statements

Conditional statements control the program flow based on specific conditions.

Examples:

- Checking valid menu selection
- Detecting suspicious URLs
- Identifying brute-force attempts

Commonly Used Conditions:

- if

- else if
- else
- switch-case

These conditions enable decision-making in the program.

9.7 Loops

Loops are used to repeat operations efficiently.

Types Used:

- while loop → used for menu repetition
- for loop → used in password generation and scanning
- do-while loop → ensures menu displays at least once

Purpose:

- Repeated execution of tasks
- Reducing code duplication
- Handling repeated user inputs

9.8 Input Validation

Input validation ensures that the program handles incorrect or unexpected inputs safely.

Examples:

- Preventing non-numeric menu input
- Ensuring password length falls within valid range
- Avoiding program crashes due to invalid input

This improves the reliability and user-friendliness of the application.

9.9 Random Number Generation

Randomness is used for password generation.

Implementation:

- srand(time(0)) initializes the random seed
- rand() generates random characters

This helps in creating unpredictable and secure passwords.

10. Program Structure

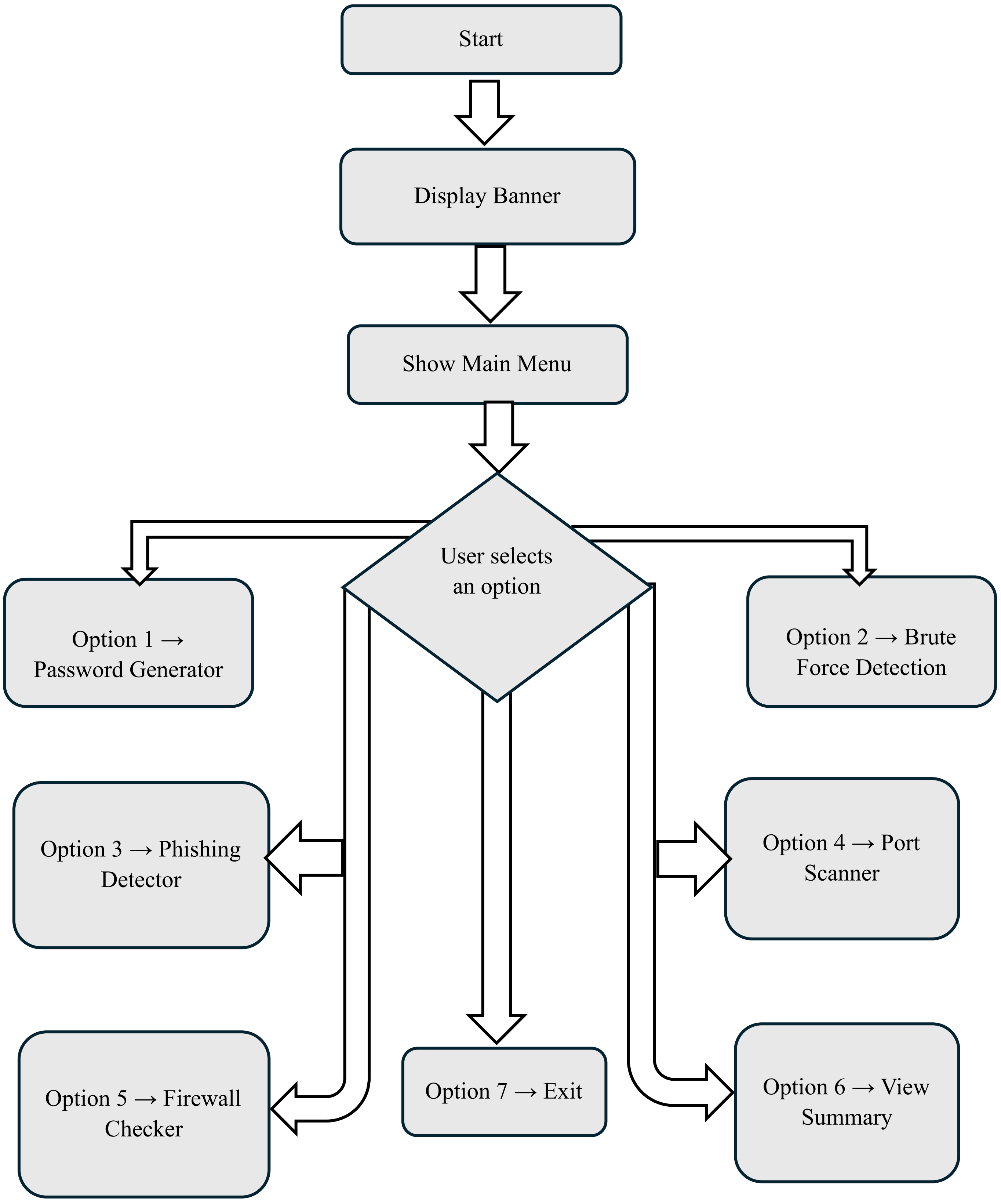
Dark Lite Toolkit follows a modular, menu-driven structure for clarity and ease of use. The main() function acts as the central coordinator, displaying a welcome banner and the main menu with various cybersecurity modules.

Each menu option triggers a dedicated function, such as password generation, phishing detection, or firewall checks. The program uses loops and conditional statements to manage flow, returning to the main menu after each operation until the user chooses to exit.

Key Components:

- **Main Function:** Controls execution and user interaction.
- **Menu System:** Lets users select different security operations.
- **Modular Functions:** Each feature is implemented separately for readability.
- **Input Validation:** Prevents invalid input and crashes.
- **Alert Management:** Tracks phishing attempts, brute-force attempts, and firewall events.
- **Summary Module:** Provides a session activity report.

Execution Flow:



Start
Display Banner
Show Main Menu
User selects an option
Option 1 → Password Generator
Option 2 → Brute Force Detection
Option 3 → Phishing Detector
Option 4 → Port Scanner
Option 5 → Firewall Checker
Option 7 → Exit
Option 6 → View Summary

11. System Limitations

1. **Simulation Only**: The tool does not perform real-time network scanning or actual cybersecurity testing.
2. **Keyword-Based Detection**: Phishing detection relies on simple keyword matching, not advanced AI or machine learning algorithms.
3. **Simulated Port Scanning**: Port scanning is simulated through user input, not actual network-level scanning.
4. **Fixed Credentials**: Brute-force simulation uses a hardcoded password ("admin123"), not real authentication systems.
5. **No Real Network Interaction**: The firewall checker analyzes text strings, not actual network packets.
6. **Limited Alert Storage**: Maximum 50 alerts can be stored per session due to fixed array size.
7. **Case-Sensitive Detection**: Phishing and firewall keyword matching is case-sensitive.
8. **No Encryption**: Generated passwords and logs are stored in plain text format.
9. **Educational Purpose Only**: Not suitable for real-world penetration testing or security auditing.
10. **No User Authentication**: The tool itself has no login system or user management.

12. Future Enhancements

12.1 Potential Improvements

1. **Advanced Phishing Detection**:
   - Implement URL analysis (check for suspicious TLDs, IP addresses)
   - Add HTTPS verification
2. **Enhanced Password Features**:
   - Password strength analyzer
   - Password hash comparison

3. **Real Network Capabilities**:
    - Actual port scanning using socket programming
    - Real firewall rule implementation
4. **User Interface Improvements**:
    - Colorized console output
    - GUI version using Qt or similar framework
5. **Data Management**:
    - Dynamic array allocation for unlimited alerts
    - Database integration (SQLite)
    - Encrypted log files
    - Export reports to PDF/CSV
6. **Additional Modules**:
    - SQL injection detector
    - XSS vulnerability scanner
    - Encryption/Decryption tools
    - Network traffic simulator
7. **Authentication System**:
    - User login with hashed passwords
    - Multi-user support
8. **Advanced Features**:
    - Machine learning for threat detection
    - Integration with external APIs (VirusTotal, Have I Been Pwned)
    - Real-time dashboard
    - Scheduled automated scans

13. Contribution Table

Names	Contribution
Aqsa Khaliq	Brute Force Detection, Firewall Rule Checker, File Handling Implementation, Main Function Integration
Areeba Shafiq	Phishing URL Detector, Port Scanner, Input Validation & Error Handling, ASCII Banner Design
Khadija Zia	Password Generator, Alert & Summary System, Firewall Rule Checker, File Handling Implementation, ASCII Banner Design, Report

14. Conclusion

The **Dark Lite Toolkit** successfully demonstrates fundamental cybersecurity concepts through an interactive and user-friendly command-line interface. By combining secure password practices, attack simulations, and logging mechanisms, the project provides valuable insight into cybersecurity fundamentals. This project serves as a strong foundation for students aspiring to explore ethical hacking, network security, and secure software development.

....