### **Forensic Investigator Tool**

### SOHAIB KHAN 2022551

### **CYBER SECURITY**

**DIGITAL FORENSICS PROJECT  
  
Week 10-11**

**Week 10-11: Implementation and Testing**

The Forensic Investigator Tool received its main development with testing during Weeks 10 through 11. During this phase I implemented and integrated forensic functionality into a working tool by transitioning from design work to practical programming activities.

**Implementation**

**Setting up the Development Environment**

* I first confirmed the installation of every library and dependency the tool required. These included:
* Tkinter for the graphical user interface (GUI).
* pytsk3 for disk image analysis.
* python-magic for file type identification.
* hashlib for file integrity checking.
* ReportLab for PDF report generation.
* pandas for data management.
* My development of Python scripts occurred through the Visual Studio Code program which served as my integrated development environment (IDE).

**Modular Design and Development**

* The **forensic project** work divided itself into separate functional modules that managed individual crime investigation tasks. I developed the following modules:
* **File Integrity Checker** produces file hashes which process against hash databases to verify unauthorized file modifications.
* I built a Digital **Evidence Timeline Generator** to extract timestamps from digital evidence files followed by a timeline creation process.
* The system checks for **unusual file types and file names** together with detection patterns of possibly harmful files.
* **The built-in functionality of pytsk3** enabled me to develop code for disk image analysis alongside deleted file recovery and metadata extraction from the images.
* **The Log File Analysis tool** I developed functioned to read log files for spotting security alerts including failed identity attempts and unauthorized system access activities.
* ****The automated artifact collection module** harvests forensic artifacts from the system that includes browsing history and recent files together with indicators of compromise.**
* **The application employs ReportLab to produce PDF report summaries containing analysis findings from all assessment modules.**

****Integration of Modules****

**I developed and integrated separate modules into two interfaces including the Command Line Interface (CLI) as well as the Tkinter-based graphical user interface (GUI). Through the GUI users can perform file and directory selection task for analysis by interacting with provided buttons and dialogs. Results from analysis modules got transmitted automatically to the Report Generation module which generated the final forensic report.**

****Error Handling and Validation****

**A wide range of error-handling capabilities was implemented into the tool for managing typical mistakes alongside type and path errors and files types the application does not support. File type validation took place within each module of the system where the Log File Analysis module verified the usage of .log and .txt files. The tool stays dependable regardless of users giving it incorrect input.**

#### ****Testing****

#### ****Unit Testing****

#### **Every individual module received unit tests that validated its basic operational components. These tests included:**

#### **The File Integrity Checker undertook assessments to verify the hash generation process along with the validation of known values.**

#### **The application evaluated its ability to restore erased files while documenting file system information belonging to disk images.**

#### **The tool demonstrated proper identification of suspect files by following established criteria for pattern detection.**

#### **The unittest Python module served to ensure all components executed according to design specifications.**

#### ****Integration Testing****

#### **I performed integration tests that verified the tool operated without glitches after ensuring individual modules operated correctly. The tool went through testing from when users enter files or directories until data reaches its final report stage.**

#### **The test included different file formats (logs and disks and documents) to verify that system components operated correctly throughout the data processing workflow.3.**

****Bug Fixing****

**The testing period allowed me to detect various bugs which I resolved successfully.**

**I solved the problems which affected file handling functionality specifically when dealing with file paths containing spaces or special characters.**

**The graphical user interface of the application became unresponsive when disk image analysis processes lasted too long. The tool received improvements through my work on progress indicators that utilized loading spinners to make the user experience better.**

****User Testing****

**The user testing phase included my offering the tool to several of my colleagues for testing purposes. Users provided essential feedback that showed how the tool needed to advance regarding:**

* **Simplifying the file selection process.**
* **The system should deliver advanced error information for all unsupported entries.**
* **The report generation process should display increased details regarding analysis outcomes.**

****Performance Optimization****

**I performed enhancements on the tool's modules which conduct large dataset work including disk image examination and log file assessment to guarantee better performance. The system uses multithreading to speed up time-intensive operations which results in higher performance levels.**

****Challenges Faced****

* **The management of extensive disk images together with extensive log files created processing delays for the system. The tool became more efficient by optimizing mathematical operations and running these tasks through background computing processes.**
* **Integrating all modules smoothly proved difficult as I worked to keep the user interface both simple and easy to use. The tool received my attention because I strived to achieve maximum usability without compromising performance or functionality.**

**Lessons Learned**

* **Every forensic module must pass through complete tests since working with crucial data requires meticulous examination.**
* **A stable and user-friendly tool demands effective error handling.**
* **Performance Optimization demands careful assessment of both the user interface and algorithms when working with big datasets.**