**Software Requirements Specification**

**for**

**Password & Encryption Detector**

Version 1.0

by Sertaç Ataç & Ramazan Bağış

24.06.2025

**Table of Contents:**

[1. Introduction 3](#_72xyxkfqoy2w)

[1.1 Purpose 3](#_8xq9q6cloq3z)

[1.2 Scope 3](#_7ufq1q9fekit)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_kl21is9b9t2d)

[1.4 Files 3](#_dmy1ah3et0cs)

[1.5 Overview 4](#_cbsvadxr1gou)

[2. General Description 4](#_apmrfeetxs47)

[2.1 Product Perspective 4](#_vlvibc1yfcna)

[2.2 Product Functions 4](#_e7z3nbnf8arm)

[2.3 Users and Characteristics 4](#_jcutfr952hfi)

[2.4 General Constraints 5](#_wcfhglh2q2uf)

[2.5 Assumptions and Dependencies 5](#_6ne9fdlgt0ku)

[3. The Specific Requirements 5](#_ayyo9ke7em88)

[3.1 External Interface Requirements 5](#_em2pz0g221qb)

[3.1.1 User Interfaces 5](#_595tvn2h5rp6)

[3.1.2 Communications Interfaces 5](#_mfbd9fg0mdnd)

[3.2 Functional Requirements 6](#_2va0zeqxgi3d)

[3.2.1 File Processing 6](#_6i7ps4lww099)

[3.2.2 File Analysis Workflow 6](#_79b8g41arb24)

[3.2.3 Results Reporting 6](#_8d2a0kwrbmfp)

[4. Other Nonfunctional Requirements 7](#_ddqrv0cj8ks)

[4.1 Performance Requirements 7](#_f1hayadxr775)

[4.2 Software Quality Attributes 7](#_vwaxxt7mjmha)

### 

### 

### 

### 

### **1. Introduction**

#### **1.1 Purpose**

This document provides a detailed description of the requirements for the "Password Protection & Encryption Detector" software. The purpose of this tool is to analyze files and directories to identify whether they are password-protected or encrypted. This document is intended for project managers, developers, and testers to understand the system's functionalities and constraints.

#### **1.2 Scope**

The software is a command-line utility designed to:

* Scan individual files or entire directories recursively.
* Detect the file type using advanced machine-learning techniques.
* Analyze supported file formats for signs of password protection and encryption using dedicated methods.
* Employ entropy and statistical analysis as a secondary method for unknown or ambiguous file types.
* Report the findings with a status, confidence level, and performance metrics for each file.

The system supports a wide range of common file types, including Microsoft Office (modern and legacy), PDF, ZIP, RAR, 7z, and others.

#### **1.3 Definitions, Acronyms, and Abbreviations**

* **CLI:** Command-Line Interface. The user interface for this tool.
* **PPDED:** Password Protection & Encryption Detector.
* **Entropy:** A measure of randomness or disorder. In this context, high entropy in a file's data is a strong indicator of encryption.
* **Handler:** A software component responsible for analyzing a specific file format.
* **Magika:** A Google-developed tool for file type identification using deep learning.

#### **1.4 Files**

This document is based on the analysis of the following source code files:

* main.py
* detector.py
* file\_handlers.py
* entropy.py
* magika\_detector.py
* type\_utils.py

#### **1.5 Overview**

This document is structured with our knowledge on the course we got in this semester “SENG205 - Software Requirements Specifications“ as follows: Section 2 provides a general description of the product. Section 3 details the specific functional and interface requirements. Section 4 outlines other non-functional requirements such as performance and software quality.

### **2. General Description**

#### **2.1 Product Perspective**

The Password Protection & Encryption Detector operates locally on a user's machine and does not require an internet connection for its core analysis functions, although the underlying Magika model may be updated independently.

#### **2.2 Product Functions**

The main functions of the software are:

1. **Command-Line Interaction:** Provide a CLI for users to specify a file or directory for analysis.
2. **File Type Identification:** Accurately determine the type of each file before analysis.
3. **Format-Specific Analysis:** Use specialized handlers to check for protection mechanisms unique to each file type.
4. **Entropy-Based Analysis:** Provide a fallback detection mechanism by analyzing the statistical properties of the file's data.
5. **Recursive Directory Scanning:** Process all files within a given directory and its subdirectories.
6. **Results Reporting:** Output a clear, human-readable summary of the analysis for each file.

#### **2.3 Users and Characteristics**

The intended users of this software include:

* **Software Engineers**
* **Software Developers**
* **System Administrators**

Users are expected to have basic knowledge of using a command-line interface.

#### **2.4 General Constraints**

* The tool is developed in Python and requires a Python environment to run.
* Its effectiveness for certain file types depends on the presence of optional third-party libraries (e.g., msoffcrypto, rarfile). The system is designed to handle the absence of these libraries gracefully but with reduced functionality.
* The analysis is based on metadata and statistical properties; it does not attempt to crack or bypass any security measures.

#### **2.5 Assumptions and Dependencies**

* The user has the necessary permissions to read the files being scanned.
* The primary dependency is Google's magika library for file type detection.
* The system assumes that files with extremely high data entropy are likely encrypted, while accounting for file formats that are naturally compressed and high in entropy.

### **3. The Specific Requirements**

#### **3.1 External Interface Requirements**

##### **3.1.1 User Interfaces**

The system provides a Command-Line Interface (CLI).

* **Invocation:** The user runs the tool from the command line (e.g., python main.py <path>).
* **Arguments:**
  + A mandatory path argument specifying the target file or directory.
  + An optional --batch flag to enable scanning of an entire directory.
* **Output:** The results of the scan be printed to the standard output on the console.
* **Error Handling:** The system displays a user-friendly error message if an invalid path or incorrect arguments are provided.

##### **3.1.2 Communications Interfaces**

Not applicable. The software operates entirely on the local machine.

#### **3.2 Functional Requirements**

##### **3.2.1 File Processing**

1. **Single File Mode:** When provided with a path to a single file, the system analyzes that file and output the result.
2. **Batch Directory Mode:** When provided with a path to a directory and the --batch flag, the system traverses the directory and all its subdirectories, analyzing each file found.
3. **File Validation:** The system checks if a file path is valid and if the file size is greater than zero before attempting analysis.

##### **3.2.2 File Analysis Workflow**

For each file, the system performs the following steps:

1. **File Type Detection:** The file's type is identified using the MagikaDetector.
2. **Handler Selection:** Based on the detected type, the system selects the appropriate file handler (e.g., PDFHandler, OfficeOpenXMLHandler).
3. **Primary Analysis (Handler-Based):** The selected handler analyzes the file for known password or encryption markers. This method provides the highest confidence.
4. **Secondary Analysis (Entropy-Based):** If the handler-based analysis returns a low confidence score (e.g., below 0.5), the system performs an entropy analysis as a fallback.
5. **Result Aggregation:** The system consolidates the results from the analysis to determine the final status.

##### **3.2.3 Results Reporting**

For each file analyzed, the system outputs the following information:

* The full file path.
* A status: "PASSWORD PROTECTED" or "NOT PASSWORD PROTECTED".
* An encryption indicator: (Encrypted: True/False).
* A confidence score for the analysis: (Confidence: X.XX).
* The time taken for the analysis: (Time: X.XXXXs). The system will also report the total execution time at the end of a scan.

### **4. Other Nonfunctional Requirements**

#### **4.1 Performance Requirements**

Program is capable of processing 1000 total files or 150GB total sized files for less than 6 seconds.

#### **4.2 Software Quality Attributes**

* **Modularity:** The project is structured on separate files. detector.py handles the process, file\_handlers.py contains the logic for each file type, entropy.py handles statistical analysis, and main.py manages the user interface.
* **Extensibility:** The architecture makes it able to add support for new file formats. A developer only needs to create a new handler class in file\_handlers.py and register it in the PasswordProtectionDetector class in detector.py.
* **Robustness:** The system uses try...except blocks to handle file I/O errors, corrupted files, and missing optional dependencies, preventing crashes and providing informations.
* **Maintainability:** The code is commented for understandings, debugs, and modifies in the future.