Zip Sentinel

2차 보고서

목차

[1. 프로젝트 개요 및 결과 보고서 본문 4](#_Toc198223285)

[1.1. 핵심 목적 4](#_Toc198223286)

[1.2. 배경 및 필요성 4](#_Toc198223287)

[1.3. 시스템 환경 5](#_Toc198223288)

[1.4. 이용자 환경 5](#_Toc198223289)

[1.5. 사전 조건 5](#_Toc198223290)

[1.6. 후사 조건 6](#_Toc198223291)

[1.7. 성공 시나리오 6](#_Toc198223292)

[1.8. 주요 변수 정의 6](#_Toc198223293)

[2. Roles & Responsibilities (책임자) 7](#_Toc198223294)

[3. Ground Rule (규칙) 8](#_Toc198223295)

[4. Use Case 시나리오 8](#_Toc198223296)

[5. Flow Chart 9](#_Toc198223297)

[6. WBS (Work Breakdown Structure) 9](#_Toc198223298)

[7. 요구 사항 명세서 대장 13](#_Toc198223299)

[7.1. 요구 사항 명세서 13](#_Toc198223300)

[8. 기능사항 명세서 대장 14](#_Toc198223301)

[8.1. 요구 사항 명세서 14](#_Toc198223302)

[9. 비기능사항 명세서 대장 15](#_Toc198223303)

[9.1. 요구 사항 명세서 15](#_Toc198223304)

문서 관리 대장

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 항목 | 버전 | 수정일자 | 수정자 | 설명 |
| Zip\_Sentinel 2차 보고서 | V1.0 | 2025-05-13 |  |  |
| Use Case 시나리오 | V1.1 | 2025-05-0 |  |  |
| Flow Chart | V1.0 | 2025-0-0 |  |  |
| WBS | V1.0 | 2025-0-0 |  |  |
| 명세서 | V2.0 | 2025-0-0 |  |  |

문서별 버전

|  |  |  |  |
| --- | --- | --- | --- |
| 버전 | 변경일자 | 변경자 | 변경내용 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. 프로젝트 개요 및 결과 보고서 본문
   1. 핵심 목적

본 시스템은 비밀번호로 보호된 압축파일의 보안 사각지대를 해소하기 위해 개발되었습니다. 게시글에 포함된 압축파일의

비밀번호를 자동으로 추출하여 파일을 해제하고, 내부 컨텐츠의 SHA-256 해시값과 VirusTotal 링크를 생성합니다.

다중 압축 구조까지 자동 분석하여 보안 담당자가 잠재적 위험 요소를 신속하게 식별할 수 있도록 지원합니다.



* 1. 배경 및 필요성

최근 CMS 환경에서는 사용자가 암호화된 압축파일(.zip, .7z, .rar 등)을 업로드해 공유하는 빈도가 높아지고 있습니다.

이들 파일 내부에는 악성코드, 멀웨어 변종, 혹은 민감 정보가 은닉될 가능성이 크며, 단순 다운로드만으로는 위협을 사전

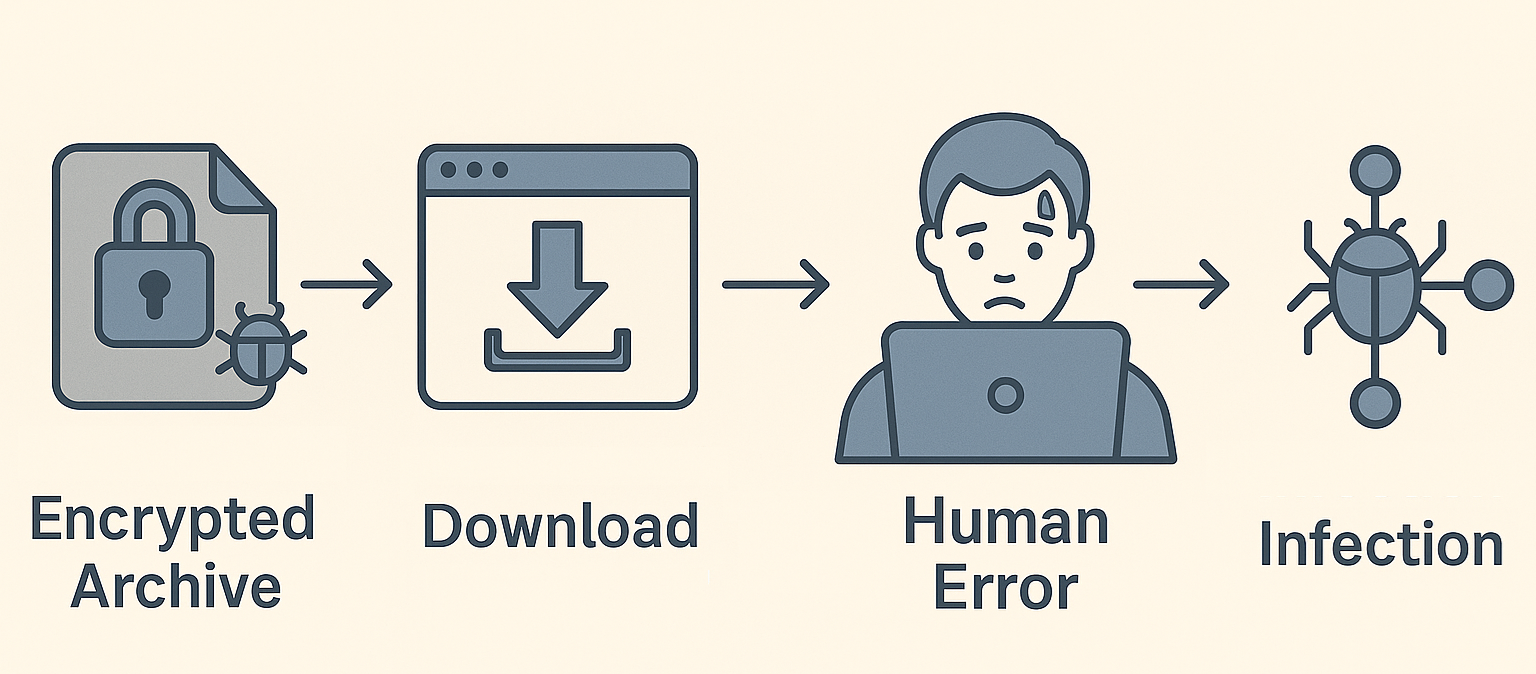
식별하기 어렵습니다. 기존에는 보안 담당자가 압축파일을 수동으로 해제하고 내부 파일을 하나하나 분석하여 다운로드 링크,

비밀번호, 해시값을 추출해야 했습니다.

이 과정은 시간 소모가 크고, 휴먼 에러로 인한 중요한 위협 누락 위험이 매우 높습니다. 따라서 게시글 내 암호화되어 있는

압축파일을 자동으로 수집·해제하고, 내부 데이터까지 포함해 SHA-256 해시값을 산출하는 한편, AI 기반 정적 분석 기법을 적용하여 잠재적 악성 패턴을 사전 식별·차단하는 통합 시스템이 필수적입니다.

이렇게 생성된 고품질의 JSON 구조화 데이터셋은 위협 대응 자동화, 보안 정책 수립, AI 모델 학습 등 모든 후속 보안   
프로세스의 핵심 자원으로 활용될 것입니다.

****

* 1. 시스템 환경

본 시스템은 Python 3.13.2 환경에서 동작하며, zipfile·rarfile(unrar 연동)·py7zr·tarfile 모듈로 ZIP/AES, RAR, 7Z, TAR.GZ 등 다양한 압축 포맷을 해제합니다. Selenium(Headless Chrome)과 requests를 통해 게시글 접근 및 다운로드 링크 추출을 자동화하고, hashlib로 SHA-256 해시값을 산출합니다. 처리 결과는 JSON과 logging으로 구조화·기록되며, VMware/  
Ubuntu 22.04 및 Docker 환경에서 독립 가상머신과 컨테이너 기반으로 운영됩니다.

|  |  |  |
| --- | --- | --- |
| **구분** | **기술** | **용도** |
| 언어/실행 환경 | Python 3.13.2 | 전체 시스템 구현 및 실행 기반 |
| 압축 해제 모듈 | Zipfile | ZIP 및 AES 암호화 ZIP 해제 |
| Rarfile | RAR 파일 해제 (unrar 연동 필요) |
| Py7zr | 7Z 파일 해제 |
| Tarfile | TAR.GZ 등 POSIX 압축 해제 |
| 크롤링/자동화 | Selenium (Headless Chrome) | 게시글 접근 및 다운로드 링크 추출 |
| Request | 정적 페이지 요청 및 파일 다운로드 |
| 해시 계산/보안 분석 | Hashlib | SHA-256 해시값 계산 |
| 파일 처리 | Os, shutil, pathlib | 해시 기반 위협 분석 페이지 자동 연결 |
| 예외 구조/구조화 | JSON, logging | 예외 기록, 결과 구조 저장 |
| 실행 환경 | VMware, Ubuntu 22.04, Docker | 독립 가상머신 및 클라우드 환경 기반 테스트 |

* 1. 이용자 환경

본 시스템은 커뮤니티 관리자, 보안 운영자, 침해 대응 팀, 기획자/PM, QA/보안 검증팀, 그리고 교육·연구자 등 다양한 이해관계자가 각자의 업무 목적에 맞춰 실시간 모니터링, 자동화된 위협 분석, 보고서 작성, 검증 및 연구 실습까지 일관되게   
수행할 수 있는 통합 플랫폼 환경을 제공합니다.

|  |  |
| --- | --- |
| **사용자** | **활용 방안** |
| 커뮤니티 관리자 | 게시글 내 파일 업로드 현황 및 압축파일 모니터링 |
| 보안 운영자 | 내부 파일 해시값을 Virus Total 링크로 직접 확인하여 위협 여부 판단 |
| 침해 대응 팀 | 위협 탐지 결과 기반으로 공격자 활동 추적 또는 IOC 수집 |
| 기획자/PM | JSON 기반 파일 유형별 통계 분석 및 리포트 작성 |
| QA/보안 | VM 내 시스템을 주기적으로 실행하여 자동 검증 로직 확보 |
| 교육·연구자 | 암호 압축파일 내부 위협 데이터셋 구축 및 분석 실습 |

* 1. 사전 조건

본 시스템이 정상적으로 작동하기 위해서는 다음과 같은 사전 조건이 충족되어야 합니다.   
이는 시스템 자체의 기능이 아닌, 시스템이 효과적으로 동작하기 위한 환경적 요구사항입니다.

|  |  |
| --- | --- |
| **조건 항목** | **설명** |
| 비밀번호 추출 가능 여부 | 게시글 내 압축파일 비밀번호가 명확히 명시되어 있어야 함 |
| 지원 압축 포맷 | .zip, .rar, .7z, .tar.gz로 한정 |
| 내부 파일 검증 | 압축 해제 후 내부 파일이 실제 존재하며 읽기 가능한 상태인지 확인 필요 |
| 크롤링 차단 요소 없음 | CAPTCHA, reCAPTCHA, Bot 차단 등 요소가 없어야 함 |
| HTML 구조 요구사항 | 게시글 내 비밀번호, 다운로드 링크, 문서 번호 등이 HTML 내에서 추출 가능해야 함 |
| 접근 방식 | 정적 및 동적 로딩 페이지 모두 지원, requests 또는 selenium으로 접근 가능 |
| 실행 환경 | Python 3.13.2 환경에서 동작 가능해야 함 |
| 설정 변수 관리 | 크롤러 설정 및 자동화 설정 (MAIN\_URL, MAX\_PAGES 등)의 주요 변수는 코드의 유연성을 고려하여 코드 상단에서 수정 가능해야 함 |

* 1. 후사 조건

본 시스템은 분석 완료 후 데이터를 체계적으로 관리하며 시스템 자원을 효율적으로 활용합니다.   
모든 분석 결과는 .json 파일에 누적 저장되어 데이터 손실 없이 완전한 이력을 유지합니다. 분석이 완료된 압축파일은 즉시   
삭제되며, 내부에 중첩된 압축파일이 발견될 경우 UNZIP\_DEPTH 설정에 따라 재귀적 분석을 수행합니다.

각 파일에 대해 파일명, SHA-256 해시값, VirusTotal 링크를 일관된 구조로 기록하며, 오류 발생 시에도 데이터 구조의 일관성을 유지합니다. 압축 해제 과정에서 문제가 발생하면 오류 사유를 명확히 기록한 후 다음 게시글로 진행하여 전체 분석 흐름이 중단되지 않도록 합니다. 모든 작업이 종료되면 임시 생성된 파일과 폴더가 완전히 제거됩니다.

**텍스트, 포스트잇 노트, 라인, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**VirusTotal 연동은 API 호출 없이 해시 기반 웹 링크로 제공되며, 향후 API 응답을 통합한 상세 보안 리포트로 확장 가능한 유연한 구조를 갖추고 있습니다.

* 1. 성공 시나리오

|  |  |  |
| --- | --- | --- |
| **단계** | **사용자 동작** | **시스템 응답** |
| 1 | 크롤러/자동화설정 | 설정 파일 로드 → 다운로드·추출 디렉터리 생성 |
| 2 | 크롤러 실행  (python crawler2.1.2.py) | run\_crawler() → 게시글 URL 수집 및 중복 제거 |
| 3 | OUTPUT\_FILE 초기화 ([] 빈 리스트 저장) |
| 4 | process\_post() 실행 → 메타정보(제목·작성자·링크 등) 추출 |
| 5 | 파일 다운로드 → 중첩 압축 해제 → 해시·VT 링크 생성 → 임시 파일 삭제 |
| 6 | |  | | --- | |  |   결과를 JSON 리스트 저장 (save\_result()) |
| 7 | 결과 확인 | 처리 과정 로깅 → 다운로드·추출 디렉터리 정리 → **메타데이터 + VT 링크 포함 JSON 데이터** |

* 1. 주요 변수 정의

|  |
| --- |
|  |
| **변수명** | | | | | **설명** |
| **MAIN\_URL** | | | | | 크롤링 대상 CMS 게시판의 기본 주소 |
| **MAX\_PAGES** | | | | | 크롤링할 페이지 수 (1부터 시작해 최대 N페이지까지) |
| **OUTPUT\_FILE** | | | | | 분석 결과가 JSON 형태로 저장될 파일 경로 및 이름 |
| **DOWNLOAD\_DIR** | | | | | 다운로드된 원본 파일을 임시 저장할 디렉터리 |
| **EXTRACT\_DIR** | | | | | 압축 해제된 내부 파일들을 저장할 디렉터리 |
| **UNZIP\_DEPTH** | | | | | 중첩 압축파일 해제를 허용할 최대 재귀 깊이 |
| **MAX\_INNER\_FILES** | | | | | 한 번의 압축 해제 시 추출할 내부 파일의 최대 개수 |
| **PASSWORD** | | | | | 암호화된 압축파일 해제에 사용할 비밀번호 |
| **VT\_URL\_TEMPLATE** | | | | | SHA-256 해시값을 기반으로 생성할 VirusTotal 웹 링크 포맷  (예: https://www.virustotal.com/gui/file/{hash}) |
|  |  |

1. Roles & Responsibilities (책임자)

텍스트, 인간의 얼굴, 스크린샷, 사람이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

1. Ground Rule (규칙)

Project Zip Sentinel은 **애자일(Agile) 방식을 기반으로 유연한 협업과 반복적 개선을 지향합니다.**   
일일 스크럼, 마감 회의, 실시간 성과 공유 등 애자일의 핵심 원칙을 바탕으로 팀워크와 업무 품질을 함께 높이고자 합니다.

텍스트, 폰트, 원, 로고이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

|  |  |
| --- | --- |
| **규칙** | **설명** |
| 회의 및 업무 동기화 | 오전 스크럼과 마감 회의를 통해 업무 방향성과 일정을 매일 공유합니다. |
| 부정적 언어 사용 금지 | 감정적 언행은 삼가고, 협업에 적합한 긍정적 표현을 사용합니다. |
| 존칭 문화 사용 | 모든 구성원은 ‘○○님’으로 호칭하며, 수평적 커뮤니케이션 문화를 지향합니다. |
| 일일 회의록 작성 기준 | 회의록은 디스코드 스레드로 기록, 마감 회의 시 해당 자료를 토대로 회의를 진행합니다. |
| 성과 정리 규칙 | 완료된 업무는 실시간으로 공유하며, 몰아서 공유하는 것은 금지합니다. |
| 파일 공유 원칙 | 슬랙은 스레드 기반 공유, 노션은 버전별 관리 및 업무 동기화에 집중합니다. |
| 워크스페이스 통합 및 동기화 | 각 팀원은 자신의 워크스페이스 링크를 노션에 URL로 등록합니다.  작업이 완료된 문서는 노션에 업로드 하여 팀원과의 싱크를 맞춥니다. |

1. Use Case 시나리오

암호화된 압축파일은 일반적인 보안 솔루션으로는 내부 내용을 확인할 수 없어, 악성코드 유포 시 탐지를 회피하는 수단으로 자주 악용됩니다.

특히 CMS 기반 커뮤니티 환경에서는 압축파일에 비밀번호를 설정하고 이를 게시글 내에 함께 명시하는 방식이 널리 사용되어, 보안 분석의 사각지대를 형성하는 주요 원인으로 작용합니다.

본 시스템은 이러한 한계를 해결하기 위해 설계되었습니다. 게시글에 포함된 비밀번호를 자동으로 추출하여 압축파일을 해제하고, 내부 파일에 대해 SHA-256 해시 기반의 정적 분석 및 VirusTotal 링크 생성\*\*을 수행함으로써 기존 보안 체계가 탐지하지 못하는 잠재적 위협까지 자동으로 식별할 수 있도록 지원합니다.

|  |  |  |
| --- | --- | --- |
| **사용자 유형** | **활용 방식** | **기대 효과** |
| 커뮤니티 관리자 | 게시글별 첨부파일 현황 및 암호 설정된 압축파일 추이 분석 | 정책 설정 (예: 암호 설정 여부 기반 자동 검토 또는  경고 표시), 우선 조치 대상 선별 |
| 보안 운영자 | 내부 파일 해시 기반 VirusTotal 조회  및 악성 의심 여부 확인 | 암호 압축으로 은폐된 위협에 대한 선제 대응 가능 |
| 침해 대응 팀 | 해시값을 통해 알려진 악성파일과의 일치 여부 검증 | 위협 인텔리전스 연계 및 빠른 차단 조치 가능 |
| 기획자/PM | 수집된 JSON 기반 트렌드 분석 및 시스템 개선 기획 | 암호 압축 사용률/탐지율 통계로 향후 전략 수립 |

1. Flow Chart

**스크린샷, 도표, 라인, 노랑이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

1. WBS (Work Breakdown Structure)

**텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

명세서 문서 대장

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **시스템** | **기술 그룹** | **ID** | **명칭** | **상태** | **최초생성**  **일자** | **최종수정**  **일자** | **기능**  **요구사항** | **기술스택**  **선정이유** | **프로세스**  **요구사항** |
|  | **REQ** | **REQ-001** | **게시글 URL 수집** |  |  |  |  |  |  |
|  | **REQ-002** | **다운로드 파일 해제 및 해시 분석** |  |  |  |  |  |  |
|  | **REQ-003** | **1GB 초과 파일 무시** |  |  |  |  |  |  |
|  | **REQ-004** | **결과 로그 및 JSON 확인** |  |  |  |  |  |  |
|  | **FUNC** | **FUNC-001** |  |  |  |  |  |  |  |
|  | **FUNC-001** |  |  |  |  |  |  |  |
|  | **FUNC-002** |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **기술 그룹** | **ID** | **명칭** | **상태** | **최초생성**  **일자** | **최종수정**  **일자** | **기능**  **요구사항** | **기술스택**  **선정이유** | **프로세스**  **요구사항** |
|  | **FUNC** | **FUNC-003** |  |  |  |  |  |  |  |
|  | **FUNC-004** |  |  |  |  |  |  |  |
|  | **FUNC-005** |  |  |  |  |  |  |  |
|  | **FUNC-006** |  |  |  |  |  |  |  |
|  | **NFR** | **FUNC-007** |  |  |  |  |  |  |  |
|  | **NFR-001** |  |  |  |  |  |  |  |
|  | **NFR-002** |  |  |  |  |  |  |  |

1. 요구 사항 명세서 대장

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **생성 일자** | **수정 일자** | **설명** |
| REQ-001 | 25.05.13 | - | URL을 통한 첨부 파일 다운로드 |
| REQ-002 | 25.05.13 | - | 압축 형식(ZIP, RAR, 7Z, TAR.GZ) 해제 및 분석 |
| REQ-003 | 25.05.13 | - | 비밀번호로 보호된 압축 파일 구분 및 처리 |
| REQ-004 | 25.05.13 | - | 중첩된 압축 파일 처리 |
| REQ-005 | 25.05.13 | - | 파일 분석이 마무리되었는지 확인 |
| REQ-006 | 25.05.13 | - | 파일 확장자, 해시값을 JSON파일로 가공 |
| REQ-007 | 25.05.13 | - | VirusTotal과 같은 보안 서비스와의 연동으로 보고서 링크 생성 |
| REQ-008 | 25.05.13 | - | 메인 압축파일명과 보고서 목록이 포함된 json 데이터셋 생성 |

* 1. 요구 사항 명세서

|  |  |  |
| --- | --- | --- |
| **요구 사항 번호** | **REQ-001** | **비고** |
| 우선 순위 | 필수 |  |
| 요구 사항 | URL을 통한 첨부 파일 다운로드 |  |
| 요구 사항 설명 | URL을 통해 압축 파일을 다운로드하여,  타임아웃 설정과 자동 디렉토리 생성을 기능한다. |  |
| 관련 요구 사항 | URL을 통해 압축 파일을 다운로드할 수 있어야 한다. 다운로드 과정에서 적절한 타임아웃 설정이 있어야 하며, 디렉토리에 파일이 존재하지 않을 경우 자동으로 디렉토리를 생성해야 한다. 다운로드된 파일은 명확하게 식별 가능한 이름으로 저장되어야 한다. |  |
| 전체 조건 | 각 압축 형식을 처리할 수 있는 라이브러리 모듈이  설치되어 있어야 한다. 충분한 처리 성능과 메모리가 확보되어 있어야 한다. |  |
| 애로 사항 | 1. 대용량 파일 다운로드 시간 지연 2. 서버 측 제한으로 인한 실패 가능성 |  |
| 메모 |  |  |
| 기능 명세서 바인딩 |  |  |

1. 기능사항 명세서 대장

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **생성 일자** | **수정 일자** | **설명** |
| REQ-001 | 25.05.13 | - | URL을 통한 첨부 파일 다운로드 |
| REQ-002 | 25.05.13 | - | 압축 형식(ZIP, RAR, 7Z, TAR.GZ) 해제 및 분석 |
| REQ-003 | 25.05.13 | - | 비밀번호로 보호된 압축 파일 구분 및 처리 |
| REQ-004 | 25.05.13 | - | 중첩된 압축 파일 처리 |
| REQ-005 | 25.05.13 | - | 파일 분석이 마무리되었는지 확인 |
| REQ-006 | 25.05.13 | - | 파일 확장자, 해시값을 JSON파일로 가공 |
| REQ-007 | 25.05.13 | - | VirusTotal과 같은 보안 서비스와의 연동으로 보고서 링크 생성 |
| REQ-008 | 25.05.13 | - | 메인 압축파일명과 보고서 목록이 포함된 json 데이터셋 생성 |

* 1. 요구 사항 명세서

|  |  |  |
| --- | --- | --- |
| **요구 사항 번호** | **FUNC-001** | **비고** |
| **기능 사항 번호** |  |  |
| **기능 목적** | create\_session() 함수는 HTTP 요청 중 발생할 수 있는 일시적인 오류(429, 500 등)에 대해 자동 재시도 가능한 세션 객체를 생성합니다. |  |
| **요구사항 바인딩** | URL을 통해 압축 파일을 다운로드하여,  타임아웃 설정과 자동 디렉토리 생성을 기능한다. |  |
| **관련 기능 사항** | URL을 통해 압축 파일을 다운로드할 수 있어야 한다. 다운로드 과정에서 적절한 타임아웃 설정이 있어야 하며, 디렉토리에 파일이 존재하지 않을 경우 자동으로 디렉토리를 생성해야 한다. 다운로드된 파일은 명확하게 식별 가능한 이름으로 저장되어야 한다. |  |
| **요구 인자** | 각 압축 형식을 처리할 수 있는 라이브러리 모듈이  설치되어 있어야 한다. 충분한 처리 성능과 메모리가 확보되어 있어야 한다. |  |
| **OUTPUT** | 1. 대용량 파일 다운로드 시간 지연 2. 서버 측 제한으로 인한 실패 가능성 |  |

1. 비기능사항 명세서 대장

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **생성 일자** | **수정 일자** | **설명** |
| REQ-001 | 25.05.13 | - | URL을 통한 첨부 파일 다운로드 |
| REQ-002 | 25.05.13 | - | 압축 형식(ZIP, RAR, 7Z, TAR.GZ) 해제 및 분석 |
| REQ-003 | 25.05.13 | - | 비밀번호로 보호된 압축 파일 구분 및 처리 |
| REQ-004 | 25.05.13 | - | 중첩된 압축 파일 처리 |
| REQ-005 | 25.05.13 | - | 파일 분석이 마무리되었는지 확인 |
| REQ-006 | 25.05.13 | - | 파일 확장자, 해시값을 JSON파일로 가공 |
| REQ-007 | 25.05.13 | - | VirusTotal과 같은 보안 서비스와의 연동으로 보고서 링크 생성 |
| REQ-008 | 25.05.13 | - | 메인 압축파일명과 보고서 목록이 포함된 json 데이터셋 생성 |

* 1. 요구 사항 명세서

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-001** | **비고** |
| 기능 목적 | create\_session() 함수는 HTTP 요청 중 발생할 수 있는 일시적인 오류 (429, 500 등)에 대해 자동 재시도 가능한 세션 객체를 생성합니다. |  |
| 요구사항 바인딩 | REQ-001 |  |
| 관련 기능 사항 | def create\_session():  session = requests.Session()  retries = Retry(  total=3,  backoff\_factor=1,  status\_forcelist=[429, 500, 502, 503, 504]  )  adapter = HTTPAdapter(max\_retries=retries)  session.mount('http://', adapter)  session.mount('https://', adapter)  return session  requests.Session(): HTTP 요청 객체 생성  Retry(total=3, ...): 최대 3회 재시도 설정  status\_forcelist: 재시도할 상태코드 지정  session.mount(...): 세션 객체에 리트라이 정책 연결 |  |
| 요구 인자 |  |  |
| OUTPUT | 성공 시: 자동 재시도 기능이 내장된 requests.Session 객체 반환  실패 시: 예외 발생 가능 (단, 호출부에서 try-except로 제어됨) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-002** | **비고** |
| 기능 목적 | create\_webdriver\_options() 함수는 리소스를 최소화하고 자동화에 적합한 Chrome WebDriver 옵션을 설정합니다. |  |
| 요구사항 바인딩 | REQ-002 |  |
| 관련 기능 사항 | def create\_webdriver\_options():  options = webdriver.ChromeOptions()  options.add\_argument("--headless")  options.add\_argument("--disable-gpu")  options.add\_argument("--no-sandbox")  --headless: 브라우저 UI 없이 실행  --disable-gpu, --disable-dev-shm-usage: 메모리 사용 최소화  --user-agent=...: 탐지 방지용 사용자 에이전트 설정  excludeSwitches: 자동화 탐지 방지 |  |
| 요구 인자 |  |  |
| OUTPUT | 성공 시: webdriver.ChromeOptions() 객체 반환  실패 시: 예외 발생 가능 (일반적으로 없음) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-003** | **비고** |
| 기능 목적 | collect\_post\_urls(driver) 함수는 Selenium 드라이버를 통해 대상 웹사이트의  게시글 목록 URL을 수집하고 JSON 파일로 저장합니다. |  |
| 요구사항 바인딩 | REQ-003 |  |
| 관련 기능 사항 | def collect\_post\_urls(driver):  with open(config['URLS\_FILE'], 'w', encoding='utf-8') as f:  json.dump(unique\_urls, f, ensure\_ascii=False, indent=2)  return unique\_urls  driver.get(url): 웹페이지 접근  driver.find\_elements(...): 게시글 링크 수집  dict.fromkeys(): 중복 URL 제거  json.dump(...): URL 목록을 파일로 저장 |  |
| 요구 인자 | driver: Selenium WebDriver 객체 |  |
| OUTPUT | 성공 시: 게시글 URL 목록을 담은 리스트 (list[str]) 반환  실패 시: 예외 발생 및 로그 경고 후 빈 리스트 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-004** | **비고** |
| 기능 목적 | collect\_download\_links(driver) 함수는 웹페이지 내에서 .exe, .zip,  .rar, .7z, .tar.gz 등 다운로드 가능한 파일의 직접/간접 링크를 수집합니다. |  |
| 요구사항 바인딩 | REQ-004 |  |
| 관련 기능 사항 | def collect\_download\_links(driver):  for a in driver.find\_elements(By.TAG\_NAME, 'a'):  href = a.get\_attribute('href') or ''  if re.search(r'\.(exe|zip|rar|msi|7z|tar\.gz)', href):  links.append(href)  By.TAG\_NAME, By.XPATH: 다양한 방식으로 다운로드 링크 탐색  re.search(...): 정규식으로 파일 확장자 필터링  driver.page\_source에서 정규식 기반 링크 추출 (백업 전략) |  |
| 요구 인자 | driver: Selenium WebDriver 객체 |  |
| OUTPUT | 성공 시: 유효한 다운로드 링크들의 리스트 (list[str]) 반환  실패 시: 예외 로그 출력 후 빈 리스트 반환 가능 (최소 한 번 백업 탐색 수행) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-005** | **비고** |
| 기능 목적 | download\_file(session, link, download\_dir) 함수는 다운로드 대상 파일의 HEAD 요청을 통해 용량을 확인한 후, 1GB 이하인 경우에만 파일을 다운로드하고 해시값 및 확장자를 포함한 상세 정보를 반환합니다. |  |
| 요구사항 바인딩 | REQ-005 |  |
| 관련 기능 사항 | def download\_file(session, link, download\_dir):  response = session.get(link, timeout=60, stream=True)  file\_hash = FileUtils.calculate\_sha256(full\_path)  return {  'url': link,  'filename': fname,  'sha256': file\_hash,  session.head: 용량 선확인  response.iter\_content: 스트리밍 다운로드  mimetypes.guess\_extension: 확장자 보완  FileUtils.calculate\_sha256: 파일 해시 계산 |  |
| 요구 인자 | session: HTTP 세션 객체  link: 다운로드 대상 링크 (str)  download\_dir: 파일 저장 디렉토리 (str) |  |
| OUTPUT | 성공 시: 파일 메타데이터 포함 딕셔너리 반환 (dict)  실패 시: 'filename': 'N/A', 'error': str(e) 등 포함된 실패 정보를 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-006** | **비고** |
| 기능 목적 | extract\_author\_id(driver) 함수는 게시글 내에서 게시자 정보를 다양한 CSS 구조에서 탐색하여 추출합니다. |  |
| 요구사항 바인딩 | REQ-006 |  |
| 관련 기능 사항 | def extract\_author\_id(driver):  if '게시자' in txt:  elif name\_elem:  elif row.find\_element(...).text == '작성자':  li, span[itemprop="name"], table tbody tr: 여러 HTML 구조 대응  '게시자', '작성자' 문자열 기반 탐색  .strip(): 불필요한 공백 제거 |  |
| 요구 인자 | driver: Selenium WebDriver 객체 |  |
| OUTPUT | 성공 시: 게시자의 ID 또는 이름 (str) 반환  실패 시: 'N/A' 문자열 반환 (기본값) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-007** | **비고** |
| 기능 목적 | process\_post(post\_url, driver, session) 함수는 게시글의 URL을 입력받아 제목, 작성자, 다운로드 링크, 파일 메타데이터, 내부 해시 정보 등을 수집하고 정리된 딕셔너리 형태로 반환합니다. |  |
| 요구사항 바인딩 | REQ-007 |  |
| 관련 기능 사항 | def process\_post(post\_url, driver, session):  title = WebDriverWait(driver, 15).until(  EC.visibility\_of\_element\_located((By.CSS\_SELECTOR, 'h1.text-big span.sw-name'))  ).text.strip()  author\_id = extract\_author\_id(driver)  links = collect\_download\_links(driver)  file\_data = download\_file(session, link, ...)  result\_with\_extraction = processor.process\_file(wrapped)  return {  'title': title,  'author\_id': author\_id,  'files': files\_info,  driver.get() 게시글 본문에 접근하여 WebDriver로 페이지 로딩을 시작합니다.  extract\_author\_id() 게시글 내 작성자 정보를 CSS 구조별로 다르게 탐색하여 ID 또는 이름을 추출합니다.  collect\_download\_links() 해당 게시글에서 다운로드 가능한 파일의 링크들을 모두 수집합니다.  download\_file() 파일을 다운로드하고, SHA-256 해시와 파일 메타 정보를 수집합니다.  ArchiveProcessor.process\_file() 압축파일 내에 포함된 파일들을 다층적으로 해제하고 해시값 및 VT URL까지 수집합니다. | 내부 압축 구조는 최대 depth 3까지 지원되며, 중첩된 파일도 모두 분석됩니다. |
| 요구 인자 | post\_url: 게시글 URL (str)  driver: Selenium WebDriver 객체  session: 요청용 requests.Session 객체 |  |
| OUTPUT | 성공 시: 게시글 정보를 포함한 정제된 딕셔너리 (dict) 반환  실패 시: None 반환 및 오류 로그 기록 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-008** | **비고** |
| 기능 목적 | run\_crawler() 함수는 전체 자동화 크롤링 프로세스의 진입점으로, 웹사이트의 게시글 URL 수집부터 게시글별 데이터 처리, 결과 저장, WebDriver 종료까지의 전체 흐름을 제어합니다. 각 단계는 서로 유기적으로 연결되어 있어, 오류 발생 시에도 로그를 남기고 안정적으로 다음 단계로 진행할 수 있도록 설계되어 있습니다. |  |
| 요구사항 바인딩 | REQ-008 |  |
| 관련 기능 사항 | def run\_crawler():  post\_urls = collect\_post\_urls(driver)  data = process\_post(post\_url, driver, session)  JsonHandler.save\_result[data]  logger.info(f"크롤링 완료: 총 {count}/{len(post\_urls)}개 게시글 처리 성공")  return True  create\_webdriver\_options() 크롬 드라이버 옵션을 설정하여 자동화에 최적화된 WebDriver 인스턴스를 생성합니다.  collect\_post\_urls() 대상 웹페이지에서 다운로드 가능한 게시글 URL 목록을 추출하여 JSON 파일로 저장합니다.  process\_post() 각 게시글에 접근하여 제목, 작성자, 비밀번호, 다운로드 링크, 내부 파일 해시 등을 수집합니다.  JsonHandler.save\_result() 수집된 데이터를 지정된 JSON 파일에 append 모드로 저장합니다.  driver.quit() 게시글 처리 후 WebDriver를 정상 종료하며, 예외 발생 시에도 finally 블록 내에서 종료가 보장됩니다. | 전체 게시글 중 처리된 개수와 성공 여부는 로그로 상세 기록됩니다.  게시글 처리 중간 실패가 있어도 전체 크롤링이 중단되지 않도록 견고하게 설계되어 있습니다. |
| 요구 인자 |  |  |
| OUTPUT | 성공 시: True 반환  실패 시: False 반환 (오류 로그 기록 포함) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-009** | **비고** |
| 기능 목적 | FileUtils.calculate\_sha256() 함수는 주어진 파일의 내용을 8192바이트 단위로  메모리에 로딩하며 SHA-256 해시값을 메모리 효율적으로 계산합니다. |  |
| 요구사항 바인딩 | REQ-009 |  |
| 관련 기능 사항 | @staticmethod  def calculate\_sha256(path):  hash\_obj = hashlib.sha256()  with open(path, "rb") as file:  for chunk in iter(lambda: file.read(8192), b""):  hash\_obj.update(chunk)  return hash\_obj.hexdigest()  open(path, "rb") 바이너리 모드로 파일을 열어 내용을 읽습니다.  iter(lambda: file.read(8192), b"") 파일을 8192바이트씩 반복해서 읽어들입니다.  hash\_obj.update(...) 읽은 데이터를 순차적으로 해시 처리합니다.  hash\_obj.hexdigest() 최종적으로 SHA-256 문자열을 반환합니다. | 대용량 파일도 안정적으로 처리할 수 있으며, 내부 파일 해시 분석 및 VirusTotal 연동 기반이 되는 핵심 함수입니다. |
| 요구 인자 | path: 해시값을 계산할 파일의 전체 경로 |  |
| OUTPUT | 성공 시: SHA-256 해시값 문자열 반환  실패 시: 예외 발생 가능 (호출부에서 try-except 처리 필요) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-010** | **비고** |
| 기능 목적 | FileUtils.get\_file\_extension() 함수는 파일명 또는 URL에서 .tar.gz와 같은 복합 확장자를 포함해 정확한 파일 확장자를 식별하여 반환합니다. |  |
| 요구사항 바인딩 | REQ-010 |  |
| 관련 기능 사항 | @staticmethod  def get\_file\_extension(filename):  if filename.lower().endswith(".tar.gz"):  return ".tar.gz"  ext = os.path.splitext(filename)[1]  return ext.lower() if ext else "N/A"  filename.lower().endswith(".tar.gz") tar.gz 파일 여부를 별도 조건으로 우선  체크합니다.  os.path.splitext() 확장자 분리를 위해 사용하며 .gz 등의 일반 확장자도  포함합니다. | .tar.gz는 일반 확장자 분리로는 .gz만 추출되므로, 별도 조건을 우선 처리합니다. |
| 요구 인자 | filename: 파일명 또는 경로 문자열 |  |
| OUTPUT | 성공 시: 실제 확장자 문자열 (예: .zip, .rar, .tar.gz)  확장자 없을 시: "N/A" 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-011** | **비고** |
| 기능 목적 | FileUtils.generate\_vt\_url() 함수는 SHA-256 해시값을 입력받아 해당 파일의 VirusTotal 검사 URL을 생성합니다. |  |
| 요구사항 바인딩 | REQ-011 |  |
| 관련 기능 사항 | @staticmethod  def generate\_vt\_url(sha256\_hash):  if AUTOMATION\_CONFIG['VT\_LOOKUP\_ENABLED']:  return f"https://www.virustotal.com/gui/file/{sha256\_hash}"  return "N/A"  AUTOMATION\_CONFIG['VT\_LOOKUP\_ENABLED'] 설정값을 기준으로  VT URL 생성 여부를 결정합니다.  f"https://www.virustotal.com/gui/file/{sha256\_hash}" 해시 기반의 고유  URL을 생성합니다. | 후속 보안 분석을 위한 외부 연계 URL로, JSON 결과에 포함되어 사람이 직접 확인할 수 있도록 합니다. |
| 요구 인자 | sha256\_hash: 대상 파일의 SHA-256 해시값 문자열 |  |
| OUTPUT | 설정값 활성화 시: VT URL 반환  비활성화 시: "N/A" 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-012** | **비고** |
| 기능 목적 | ArchiveProcessor.process\_file() 함수는 다운로드된 파일이 압축파일일 경우 이를 해제하고 내부 파일들의 메타데이터(해시값, 확장자, 크기 등)를 추출하여 최종 분석 결과를 구성합니다. |  |
| 요구사항 바인딩 | REQ-012 |  |
| 관련 기능 사항 | def process\_file(self, file\_info):  for depth in range(1, self.config['UNZIP\_DEPTH'] + 1):  extracted\_files, err = self.extract\_archive(filepath, password, depth)  file\_entry['extracted\_files'] = extraction\_results  self.extract\_archive() 다중 압축 구조를 깊이(depth)별로 반복 해제합니다.  FileUtils.calculate\_sha256() 내부 파일들의 해시값을 개별적으로 계산합니다.  FileUtils.generate\_vt\_url() 각 내부 파일에 대한 VirusTotal URL을 생성합니다. | 압축 대상이 아닌 파일은 자동 스킵되며, 압축 대상은 최대 depth만큼 반복 해제됩니다. |
| 요구 인자 | file\_info: 비밀번호 및 다운로드 파일 목록이 포함된 딕셔너리 |  |
| OUTPUT | 성공 시: 내부 파일 정보 및 요약을 포함한 딕셔너리 반환  실패 시: "error" 키가 포함된 오류 정보 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-013** | **비고** |
| 기능 목적 | ArchiveProcessor.extract\_archive() 함수는 zip, rar, 7z, tar.gz 등 다양한 압축 포맷의 파일을 해제하고, 내부 파일의 해시값, 크기, 확장자 등의 메타데이터를 수집합니다. |  |
| 요구사항 바인딩 | REQ-013 |  |
| 관련 기능 사항 | def extract\_archive(self, filepath, password, depth):  if extension == '.zip':  with ZipFile(filepath) as zf:  zf.extractall(path=inner\_dir, pwd=password.encode())  elif extension == '.rar':  with rarfile.RarFile(filepath) as rf:  rf.extractall(path=inner\_dir, pwd=password)  elif extension == '.7z':  with py7zr.SevenZipFile(filepath, mode='r', password=password) as sz:  sz.extractall(path=inner\_dir)  elif extension == '.tar.gz':  with tarfile.open(filepath, mode='r:gz') as tf:  tf.extract(member, path=inner\_dir)  os.path.exists(filepath) 압축 파일 존재 여부 확인  ZipFile, RarFile, SevenZipFile, tarfile 각각의 확장자 유형에 맞는 해제 방식 수행  FileUtils.calculate\_sha256() 내부 파일 해시 계산  FileUtils.generate\_vt\_url() VirusTotal URL 생성 | 내부 폴더 구조는 압축 깊이별로 \_depthN 디렉토리에 구분 저장됩니다.  압축 해제 실패 시 7z CLI를 통한 폴백 로직으로 이어집니다. |
| 요구 인자 | filepath: 압축파일 경로  password: 압축 해제 비밀번호  depth: 현재 압축 해제 깊이 (1부터 시작) |  |
| OUTPUT | 성공 시: 내부 파일 목록 (list[dict])과 오류 코드 None  실패 시: 빈 리스트와 오류 코드 문자열 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-014** | **비고** |
| 기능 목적 | extract\_with\_7z() 함수는 기본 라이브러리 기반 압축 해제에 실패했을 때, 7z CLI를 호출하여 압축 해제를 시도하는 폴백(fallback) 기능을 수행합니다. |  |
| 요구사항 바인딩 | REQ-014 |  |
| 관련 기능 사항 | def extract\_with\_7z(zip\_path, password, output\_dir, depth=1):  cmd = [  "7z", "x", zip\_path,  f"-o{output\_dir}",  f"-p{password}" if password != 'N/A' else "-p",  "-y"  ]  result = subprocess.run(cmd,)  "7z", "x", zip\_path 7z CLI를 통해 압축 해제 명령 실행  -p{password} 비밀번호 지정  scan\_extracted\_files() 해제된 디렉토리 내 파일 메타데이터 수집 | 내부적으로 시스템에 7z가 설치되어 있어야 동작하며, subprocess 기반으로 실행됩니다.  CLI 기반이라 라이브러리 해제가 실패하는 특수 파일에도 대응 가능성이 높습니다. |
| 요구 인자 | zip\_path: 압축파일 경로  password: 비밀번호  output\_dir: 압축 해제 디렉토리  depth: 현재 압축 깊이 |  |
| OUTPUT | 성공 시: 내부 파일 목록과 오류 코드 None  실패 시: 빈 리스트와 오류 메시지 반환 |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-015** | **비고** |
| 기능 목적 | scan\_extracted\_files() 함수는 압축 해제된 디렉토리 내 파일들을 재귀적으로 순회하며, 파일 이름, 해시값, 크기, 확장자, 압축 깊이 등의 메타데이터를 수집합니다. |  |
| 요구사항 바인딩 | REQ-015 |  |
| 관련 기능 사항 | def scan\_extracted\_files(directory, depth\_level):  for root, \_, files in os.walk(directory):  for fname in files:  full\_path = os.path.join(root, fname)  sha = FileUtils.calculate\_sha256(full\_path)  ext = FileUtils.get\_file\_extension(fname)  os.walk(directory) 압축 해제된 디렉토리 내 모든 파일 순회  calculate\_sha256() 파일 해시값 계산  get\_file\_extension() 확장자 판별  generate\_vt\_url() VirusTotal 링크 생성 | scan 결과는 extracted\_files 필드에 저장되며, 다단계 압축 구조 내 파일 분석에 필수적인 역할을 합니다. |
| 요구 인자 | directory: 압축 해제된 디렉토리 경로  depth\_level: 현재 압축 깊이 |  |
| OUTPUT | 성공 시: 파일 메타데이터가 담긴 리스트 (list[dict])  실패 시: 예외 발생 시 개별 파일 스킵 (전체 흐름은 유지) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-016** | **비고** |
| 기능 목적 | JsonHandler.save\_result() 함수는 크롤링 및 분석 결과를 JSON 파일에 저장하며, append 모드일 경우 기존 파일에 안전하게 누적합니다. |  |
| 요구사항 바인딩 | REQ-016 |  |
| 관련 기능 사항 | @staticmethod  def save\_result(data, output\_file, append=False):  if append and os.path.exists(output\_file):  with open(output\_file, 'r', encoding='utf-8') as f:  existing\_data = json.load(f)  existing\_data.append(data)  with open(output\_file, 'w', encoding='utf-8') as f:  json.dump(existing\_data, f, indent=2, ensure\_ascii=False)  else:  with open(output\_file, 'w', encoding='utf-8') as f:  json.dump([data], f, indent=2, ensure\_ascii=False)  append=True 기존 JSON 데이터를 불러와 병합 저장  json.load(), json.dump() JSON 입출력 처리  ensure\_ascii=False, indent=2 가독성을 고려한 저장 옵션 적용 | JSON 형식이 손상된 경우 새로 초기화하여 저장하며, 데이터 누락 방지를 위한 방어적 설계가 적용돼 있습니다. |
| 요구 인자 | data: 저장할 결과 데이터 (dict 또는 list)  output\_file: 저장 대상 JSON 파일 경로  append: 기존 파일에 추가 여부 (bool, 기본값 False) |  |
| OUTPUT | 성공 시: True 반환  실패 시: False 반환 (오류 로그 기록) |  |

|  |  |  |
| --- | --- | --- |
| **기능 사항 번호** | **FUNC-017** | **비고** |
| 기능 목적 | JsonHandler.load\_data() 함수는 지정된 JSON 파일을 불러와 리스트 형태의 데이터를 반환하며, 파일이 존재하지 않거나 파싱 실패 시 빈 리스트를 반환합니다. |  |
| 요구사항 바인딩 | REQ-017 |  |
| 관련 기능 사항 | @staticmethod  def load\_data(input\_file):  with open(input\_file, 'r', encoding='utf-8') as f:  return json.load(f)  open(..., 'r') 읽기 모드로 파일 열기  json.load() JSON 파싱 수행  except Exception 오류 발생 시 빈 리스트 반환 | 크롤링 결과 후처리나 재가공을 위한 재활용이 가능하며, JSON 구조 손상 대비 설계가 포함돼 있습니다. |
| 요구 인자 | input\_file: 불러올 JSON 파일 경로 |  |
| OUTPUT | 성공 시: JSON 데이터 객체 (list 또는 dict)  실패 시: 빈 리스트 반환 |  |