악성코드 탐지, 암호화 및 패키징

파일 프로텍터 개발 명세서

문서번호 : DS-102

VER1.0

|  |  |
| --- | --- |
| 1. from pymongo import MongoClient  2. import downloadDB\_Nor # 파일 저장 모듈  3. from logging\_Utils import setup\_logger # 공통 로그 설정 함수  4. from dotenv import load\_dotenv  5. import os  6.  7. load\_dotenv()  8. #env에서 로드  9. MONGO\_URI=os.getenv("MONGO\_URI")  10. # 로그 설정  11. log\_file = r'C:\pymodules\log\db\_monitor.log'  12. logger = setup\_logger(log\_file)  13.  14. def monitor\_files():  15. """MongoDB Change Stream을 사용하여 normal\_files.filedata 테이블을 모니터링"""  16. try:  17. # MongoDB에 연결  18. client = MongoClient(MONGO\_URI)  19. db = client['normal\_files']  20.  21. # Change Streams로 normal\_files에서 파일 업로드를 감시  22. with db.filedata.watch() as stream:  23. logger.info("Started monitoring new files in MongoDB using Change Streams.")  24. for change in stream:  25. if change["operationType"] == "insert":  26. file\_doc = change["fullDocument"]  27. logger.info(f"New file detected: {file\_doc['filename']}")  28.  29. # 2번째 단계: 파일을 로컬에 저장하는 모듈 호출  30. local\_file\_path = downloadDB\_Nor.store\_file(file\_doc)  31. if local\_file\_path:  32. logger.info(f"File {file\_doc['filename']} has been saved to {local\_file\_path}")  33. else:  34. logger.error(f"Failed to save file {file\_doc['filename']} to local storage.")  35.  36. else:  37. logger.info(f"Ignored operation: {change['operationType']}")  38.  39. except Exception as e:  40. logger.error(f"Error occurred during file monitoring: {e}", exc\_info=True)  41.  42. if \_\_name\_\_ == "\_\_main\_\_":  43. monitor\_files()  44. | |
| 목적 | Mongo DB에 올라온 유저가 올린 파일 모니터링 모듈 |
| 파일명 | monitorDB.py |

|  |  |
| --- | --- |
| 1. import os  2. from pymongo import MongoClient  3. from datetime import datetime  4. import convert\_NorToEnc # 암호화 모듈  5. import uploadDB\_NorPE # PE 섹션 분석 및 저장 모듈  6. import check\_Enc # 암호화 확인 모듈  7. from logging\_Utils import setup\_logger # 로그 설정 함수  8. from dotenv import load\_dotenv  9. load\_dotenv()  10.  11. #env에서 로드  12. MONGO\_URI=os.getenv("MONGO\_URI")  13. # 로그 파일 설정  14. log\_file = r'C:\pymodules\log\db\_file\_store.log'  15. logger = setup\_logger(log\_file)  16.  17. # MongoDB 연결 설정  18. client = MongoClient(MONGO\_URI)  19. db = client['normal\_files']  20. collection = db['filedata']  21.  22. def store\_file(file\_doc):  23. """DB에서 파일을 다운로드하고 signature\_id로 로컬에 저장 (확장자 필드 사용)"""  24. try:  25. signature\_id = file\_doc["signature\_id"]  26. file\_extension = file\_doc.get("file\_extension", "")  27.  28. # 확장자가 없거나 잘못된 형식일 경우 수정  29. if not file\_extension.startswith("."):  30. file\_extension = f".{file\_extension}"  31.  32. # 오늘 날짜를 기준으로 폴더 생성  33. today\_date = datetime.today().strftime('%Y-%m-%d')  34. local\_folder\_path = os.path.join(r'C:\DBFiles', today\_date, 'origin\_files')  35.  36. # 로컬 폴더가 존재하지 않으면 생성  37. if not os.path.exists(local\_folder\_path):  38. os.makedirs(local\_folder\_path)  39.  40. # 파일명을 signature\_id로 하고, 확장자를 적용  41. local\_file\_path = os.path.join(local\_folder\_path, f"{signature\_id}{file\_extension}")  42.  43. # MongoDB에서 파일 데이터 다운로드 및 로컬에 저장  44. with open(local\_file\_path, "wb") as f:  45. f.write(file\_doc["file\_data"]) # 파일 데이터를 바이너리로 저장  46.  47. logger.info(f"File with signature\_id {signature\_id} has been saved to {local\_file\_path}")  48.  49. # 3-1: 파일 PE 섹션 분석  50. pe\_info = check\_Enc.analyze\_pe\_sections(file\_doc["file\_data"], is\_path=False)  51.  52. # 3-2: 섹션 암호화 여부 확인  53. encryption\_status = check\_Enc.check\_file\_encryption(pe\_info)  54.  55. # 3-3: PE 섹션 정보를 MongoDB에 저장  56. uploadDB\_NorPE.store\_normal\_file\_pe\_info(signature\_id, file\_doc["filename"], file\_doc["upload\_ip"], pe\_info, encryption\_status)  57.  58. # 3-4: 파일 암호화 작업 실행  59. encrypted\_file\_path = convert\_NorToEnc.encrypt\_with\_themida(local\_file\_path, file\_doc)  60. if encrypted\_file\_path:  61. logger.info(f"File {local\_file\_path} has been successfully encrypted.")  62. else:  63. logger.error(f"Encryption failed for file {local\_file\_path}.")  64.  65. except Exception as e:  66. logger.error(f"Error storing file with signature\_id {file\_doc.get('signature\_id', 'unknown')}: {e}", exc\_info=True)  67. print(f"Error storing file with signature\_id {file\_doc.get('signature\_id', 'unknown')}. Check log for details.")  68. | |
| 목적 | DB에 올라온 사용자 파일 다운로드 모듈 |
| 파일명 | downlaodDB.py |

|  |  |
| --- | --- |
| 1. import subprocess  2. import os  3. from logging\_Utils import setup\_logger  4. from datetime import datetime  5. import uploadDB\_Enc # MongoDB에 암호화된 파일을 저장하는 모듈  6. import uploadDB\_EncPE # MongoDB에 암호화된 파일 PE를 저장하는 모듈  7. import check\_Enc  8.  9. # 로그 파일 설정  10. log\_file = r'C:\pymodules\log\file\_encryption.log'  11. logger = setup\_logger(log\_file)  12.  13. def encrypt\_with\_themida(input\_file, original\_file\_doc):  14. """  15. Themida로 파일 암호화하고, Themida 작업 완료 후 암호화된 파일의 PE를 분석하여 MongoDB에 저장  16. :param input\_file: 원본 파일 경로  17. :param original\_file\_doc: 원본 파일의 메타데이터 (DB에서 가져온 정보)  18. :return: 암호화된 파일 경로 또는 None  19. """  20. try:  21. # 암호화된 파일을 저장할 경로 설정  22. today\_date = datetime.today().strftime('%Y-%m-%d')  23. protected\_folder\_path = os.path.join(r'C:\DBFiles', today\_date, 'protected')  24.  25. # 폴더가 존재하지 않으면 생성  26. if not os.path.exists(protected\_folder\_path):  27. os.makedirs(protected\_folder\_path)  28.  29. # 암호화된 파일명을 설정  30. file\_name, file\_extension = os.path.splitext(os.path.basename(input\_file))  31. output\_file = os.path.join(protected\_folder\_path, f"{file\_name}\_protected{file\_extension}")  32.  33. logger.info(f"Starting encryption for file: {input\_file}")  34.  35. # 배치 파일 경로 (환경에 맞게 수정)  36. bat\_file\_path = r"C:\pymodules\bat\themida\_encrypt.bat"  37.  38. # 배치 파일 실행, stdout과 stderr를 로그로 기록  39. result = subprocess.run([bat\_file\_path, input\_file, output\_file], stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)  40.  41. # 배치 파일 실행 후 결과 확인  42. if result.returncode == 0:  43. logger.info(f"File {input\_file} has been encrypted successfully to {output\_file}")  44. logger.info(f"Batch file output: {result.stdout.decode()}") # 배치 파일 출력 로그  45.  46. # 암호화가 완료된 파일을 MongoDB에 저장  47. uploadDB\_Enc.store\_encrypted\_file(output\_file, original\_file\_doc) # MongoDB에 저장  48. logger.info(f"Encrypted file {output\_file} has been saved to MongoDB.")  49.  50. # Themida 암호화 작업이 완료된 후, PE 섹션 분석 진행  51. analyze\_and\_store\_pe(output\_file, original\_file\_doc)  52.  53. return output\_file  54. else:  55. logger.error(f"Error in Themida encryption for {input\_file}: Return code {result.returncode}")  56. logger.error(f"Error message: {result.stderr.decode()}") # 배치 파일 오류 로그  57. return None  58.  59. except Exception as e:  60. logger.error(f"Error during encryption for {input\_file}: {e}", exc\_info=True)  61. return None  62.  63. def analyze\_and\_store\_pe(encrypted\_file, original\_file\_doc):  64. """  65. Themida 암호화 작업이 끝난 후 암호화된 파일의 PE를 분석하고, MongoDB에 저장  66. :param encrypted\_file: 암호화된 파일 경로  67. :param original\_file\_doc: 원본 파일의 메타데이터 (DB에서 가져온 정보)  68. """  69. try:  70. # 1: 파일 PE 섹션 분석  71. pe\_info = check\_Enc.analyze\_pe\_sections(encrypted\_file, is\_path=True)  72.  73. # 2: 섹션 암호화 여부 확인  74. encryption\_status = check\_Enc.check\_file\_encryption(pe\_info)  75.  76. # 3: MongoDB에 PE 정보와 암호화 여부 저장  77. signature\_id = original\_file\_doc["signature\_id"]  78. uploadDB\_EncPE.store\_encrypted\_file\_pe\_info(signature\_id, encrypted\_file, original\_file\_doc["upload\_ip"], pe\_info, encryption\_status)  79.  80. logger.info(f"PE information and encryption status for encrypted file {encrypted\_file} successfully stored in MongoDB.")  81.  82. except Exception as e:  83. logger.error(f"Error during PE analysis for encrypted file {encrypted\_file}: {e}", exc\_info=True)  84. | |
| 목적 | 사용자가 올린 파일을 프로텍터로 암호화 모듈 |
| 파일명 | conver\_NorToEnc.py |

|  |  |
| --- | --- |
| 1. import os  2. from pymongo import MongoClient  3. import gridfs  4. from logging\_Utils import setup\_logger  5. from datetime import datetime  6. from dotenv import load\_dotenv  7.  8. load\_dotenv()  9. #env에서 로드  10. MONGO\_URI=os.getenv("MONGO\_URI")  11. # 로그 설정  12. log\_file = r'C:\pymodules\log\file\_encryption.log'  13. logger = setup\_logger(log\_file)  14.  15. # MongoDB 연결 설정  16. client = MongoClient(MONGO\_URI)  17. db = client['encrypted\_files'] # encrypted\_files DB 사용  18. fs = gridfs.GridFS(db) # GridFS를 이용해 파일을 저장  19.  20. def store\_encrypted\_file(encrypted\_file\_path, original\_file\_doc):  21. """  22. Themida로 보호된 파일을 GridFS에 저장하고, 관련 메타데이터를 MongoDB의 encrypted\_files.filedata 컬렉션에 저장.  23.  24. :param encrypted\_file\_path: 보호된 파일 경로  25. :param original\_file\_doc: 원본 파일의 메타데이터 (MongoDB에서 가져온 정보)  26. """  27. try:  28. # 암호화된 파일을 읽어서 MongoDB에 저장  29. with open(encrypted\_file\_path, "rb") as f:  30. encrypted\_file\_data = f.read()  31.  32. # GridFS에 암호화된 파일 저장  33. encrypted\_file\_id = fs.put(  34. encrypted\_file\_data,  35. filename=os.path.basename(encrypted\_file\_path),  36. original\_filename=original\_file\_doc["filename"],  37. original\_signature\_id=original\_file\_doc["signature\_id"],  38. file\_extension=os.path.splitext(encrypted\_file\_path)[1],  39. encrypted=True # 파일이 암호화되었음을 표시  40. )  41.  42. # 메타데이터 저장: 암호화된 파일 정보  43. encrypted\_file\_doc = {  44. "signature\_id": original\_file\_doc["signature\_id"], # 원본 파일의 signature\_id 유지  45. "original\_filename": original\_file\_doc["filename"], # 원본 파일명  46. "encrypted\_filename": os.path.basename(encrypted\_file\_path), # 암호화된 파일명  47. "original\_upload\_time": original\_file\_doc["upload\_time"], # 원본 파일 업로드 시간  48. "encrypted\_upload\_time": datetime.now(), # 암호화된 파일 저장 시간  49. "upload\_ip": original\_file\_doc["upload\_ip"], # 업로드한 IP 정보  50. "gridfs\_file\_id": encrypted\_file\_id # GridFS에 저장된 파일의 ID  51. }  52.  53. # encrypted\_files의 filedata 컬렉션에 메타데이터 저장  54. db.filedata.insert\_one(encrypted\_file\_doc)  55.  56. logger.info(f"Encrypted file {encrypted\_file\_path} has been stored in MongoDB successfully.")  57. return True  58.  59. except Exception as e:  60. logger.error(f"Error storing encrypted file {encrypted\_file\_path} in MongoDB: {e}", exc\_info=True)  61. return False  62. | |
| 목적 | 프로텍터로 암호화된 파일 DB에 업로드 모듈 |
| 파일명 | uploadDB\_Enc.py |

|  |  |
| --- | --- |
| 1. from pefile import PE  2. from logging\_Utils import setup\_logger # logging\_Utils에서 로거 설정 함수 가져오기  3. import hashlib  4.  5. # 로그 파일 설정  6. log\_file = r'C:\pymodules\log\check\_enc.log'  7. logger = setup\_logger(log\_file) # logging\_utils의 setup\_logger 사용  8.  9. def analyze\_pe\_sections(file\_input, is\_path=False):  10. """  11. PE 파일의 섹션을 분석하고, 필요한 섹션 정보를 추출  12. :param file\_input: 파일 경로 또는 파일 데이터  13. :param is\_path: True면 파일 경로로 처리, False면 파일 데이터로 처리  14. :return: 섹션 정보가 담긴 딕셔너리  15. """  16. try:  17. # 파일 경로로 분석할지, 파일 데이터로 분석할지 결정  18. if is\_path:  19. pe = PE(file\_input) # 파일 경로를 사용한 경우  20. else:  21. pe = PE(data=file\_input) # 파일 데이터를 사용한 경우  22.  23. sections = {}  24.  25. for section in pe.sections:  26. section\_name = section.Name.decode('utf-8').rstrip('\x00')  27.  28. # 섹션의 MD5 해시 계산  29. section\_data = section.get\_data()  30. section\_md5 = hashlib.md5(section\_data).hexdigest()  31.  32. # 추가적인 섹션 정보 추출  33. sections[section\_name] = {  34. "virtual\_address": section.VirtualAddress, # 메모리 상의 가상 주소  35. "virtual\_size": section.Misc\_VirtualSize, # 메모리 로드 시 크기  36. "raw\_size": section.SizeOfRawData, # 실제 파일에서의 크기  37. "entropy": section.get\_entropy(), # 엔트로피  38. "md5\_hash": section\_md5, # 섹션의 MD5 해시  39. "characteristics": section.Characteristics, # 섹션의 권한 정보 (읽기/쓰기/실행)  40. "permissions": {  41. "readable": bool(section.Characteristics & 0x40000000),  42. "writable": bool(section.Characteristics & 0x80000000),  43. "executable": bool(section.Characteristics & 0x20000000),  44. },  45. # PE 값 추가  46. "pointer\_to\_raw\_data": section.PointerToRawData, # 파일 내에서 섹션의 시작 위치  47. "section\_alignment": pe.OPTIONAL\_HEADER.SectionAlignment # 섹션의 메모리 정렬 값  48. }  49.  50. logger.info("PE section analysis completed successfully.")  51. return sections  52.  53. except Exception as e:  54. logger.error(f"Error during PE section analysis: {e}", exc\_info=True)  55. return {}  56.  57. # 섹션이 암호화되었는지 확인하는 함수  58. def is\_section\_encrypted(section):  59. entropy\_threshold = 7.0 # 예시 임계값 (일반적으로 7.0 이상이 암호화된 섹션으로 간주됨)  60. return section['entropy'] > entropy\_threshold  61.  62. # 파일의 섹션 암호화 상태를 확인하는 함수  63. def check\_file\_encryption(pe\_info):  64. encryption\_status = {}  65.  66. try:  67. for section\_name, section\_data in pe\_info.items():  68. encrypted = is\_section\_encrypted(section\_data)  69. encryption\_status[section\_name] = encrypted  70. logger.info(f"Section {section\_name}: {'Encrypted' if encrypted else 'Not Encrypted'} (Entropy: {section\_data['entropy']})")  71.  72. logger.info("File encryption status check completed successfully.")  73. return encryption\_status  74.  75. except Exception as e:  76. logger.error(f"Error during file encryption status check: {e}", exc\_info=True)  77. return {}  78. | |
| 목적 | 파일의 PE 분석 및 암호화 확인 모듈 |
| 파일명 | check\_Enc.py |

|  |  |
| --- | --- |
| 1. from pymongo import MongoClient  2. import datetime  3. from logging\_Utils import setup\_logger # 로깅 설정 함수 가져오기  4. from dotenv import load\_dotenv  5. import os  6.  7. load\_dotenv()  8. #env에서 로드  9. MONGO\_URI=os.getenv("MONGO\_URI")  10. # 로그 파일 설정  11. log\_file = r'C:\pymodules\log\upload\_db\_norpe.log'  12. logger = setup\_logger(log\_file)  13.  14. # 일반 파일의 PE 정보를 MongoDB에 저장하는 함수  15. def store\_normal\_file\_pe\_info(signature\_id, filename, upload\_ip, pe\_info, encryption\_status):  16. try:  17. # MongoDB 연결  18. client = MongoClient(MONGO\_URI)  19. db = client['normal\_files']  20. normal\_file\_pe\_info\_collection = db['pe\_info']  21.  22. # 업로드 시간을 현재 시각으로 설정  23. upload\_time = datetime.datetime.now()  24.  25. # MongoDB에 저장할 PE 데이터 생성  26. pe\_data = {  27. "signature\_id": signature\_id,  28. "filename": filename,  29. "upload\_time": upload\_time,  30. "upload\_ip": upload\_ip,  31. "pe\_info": pe\_info,  32. "encrypted": encryption\_status  33. }  34.  35. # MongoDB에 데이터 삽입  36. normal\_file\_pe\_info\_collection.insert\_one(pe\_data)  37. logger.info(f"PE information for file {filename} (signature\_id: {signature\_id}) successfully stored in MongoDB.")  38.  39. except Exception as e:  40. logger.error(f"Error storing PE information for file {filename} (signature\_id: {signature\_id}): {e}", exc\_info=True)  41. | |
| 목적 | 사용자가 업로드 한 파일의 PE를 DB에 업로드 하는 모듈 |
| 파일명 | uploadDB\_NorPE.py |

|  |  |
| --- | --- |
| 1. from pymongo import MongoClient  2. import datetime  3. from logging\_Utils import setup\_logger # 로깅 설정 함수 가져오기  4. from dotenv import load\_dotenv  5. import os  6.  7. load\_dotenv()  8. #env에서 로드  9. MONGO\_URI=os.getenv("MONGO\_URI")  10. # 로그 파일 설정  11. log\_file = r'C:\pymodules\log\upload\_db\_encpe.log'  12. logger = setup\_logger(log\_file)  13.  14. # 암호화된 파일의 PE 정보를 MongoDB에 저장하는 함수  15. def store\_encrypted\_file\_pe\_info(signature\_id, encrypted\_filename, upload\_ip, pe\_info, encryption\_status):  16. try:  17. # MongoDB 연결 설정  18. client = MongoClient(MONGO\_URI)  19. db = client['encrypted\_files']  20. encrypted\_file\_pe\_info\_collection = db['pe\_info']  21.  22. # 업로드 시간을 현재 시간으로 설정  23. upload\_time = datetime.datetime.now()  24.  25. # MongoDB에 저장할 PE 데이터 생성  26. pe\_data = {  27. "signature\_id": signature\_id,  28. "encrypted\_filename": encrypted\_filename,  29. "encrypted\_upload\_time": upload\_time,  30. "upload\_ip": upload\_ip,  31. "pe\_info": pe\_info, # PE 정보 저장  32. "encrypted": encryption\_status # 섹션 암호화 여부 저장  33. }  34.  35. # MongoDB에 데이터 삽입  36. encrypted\_file\_pe\_info\_collection.insert\_one(pe\_data)  37. logger.info(f"PE information and encryption status for encrypted file {encrypted\_filename} (signature\_id: {signature\_id}) successfully stored in MongoDB.")  38.  39. except Exception as e:  40. logger.error(f"Error storing PE information for encrypted file {encrypted\_filename} (signature\_id: {signature\_id}): {e}", exc\_info=True)  41. | |
| 목적 | 프로텍터로 암호화된 파일 PE를 DB에 업로드 하는 모듈 |
| 파일명 | uploadDB\_EncPE.py |

|  |  |
| --- | --- |
| 1. import logging  2. import os  3.  4. def setup\_logger(log\_file\_path):  5. """로그 파일 경로를 받아 로거를 설정"""  6. logger = logging.getLogger(log\_file\_path)  7. logger.setLevel(logging.INFO)  8.  9. # 중복 로그 방지  10. if not logger.hasHandlers():  11. # 로그 핸들러 설정  12. file\_handler = logging.FileHandler(log\_file\_path)  13. formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')  14. file\_handler.setFormatter(formatter)  15. logger.addHandler(file\_handler)  16.  17. return logger  18. | |
| 목적 | 각 모듈 로그 관리하는 모듈 |
| 파일명 | logging\_Utils.py |

|  |  |
| --- | --- |
| 1. @echo off  2. echo Starting Themida encryption...  3. set input\_file=%1  4. set output\_file=%2  5.  6. REM sigcheck.exe로 파일 아키텍처 확인  7. C:\pymodules\sigcheck\sigcheck64.exe -a %input\_file% > temp\_arch.txt  8.  9. REM temp\_arch.txt에서 아키텍처 정보를 읽어옴  10. findstr /i "32-bit" temp\_arch.txt > nul  11. if %errorlevel% == 0 (  12. echo 32비트 파일입니다. Themida 32비트 버전으로 보호 작업을 실행합니다.  13. "C:\Program Files (x86)\Themida Full Activated\Themida.exe" /protect "C:\Program Files (x86)\Themida Full Activated\Prevision.tmd" /inputfile %input\_file% /outputfile %output\_file%  14. goto done  15. )  16.  17. findstr /i "64-bit" temp\_arch.txt > nul  18. if %errorlevel% == 0 (  19. echo 64비트 파일입니다. Themida 64비트 버전으로 보호 작업을 실행합니다.  20. "C:\Program Files (x86)\Themida Full Activated\Themida64.exe" /protect "C:\Program Files (x86)\Themida Full Activated\Prevision64.tmd" /inputfile %input\_file% /outputfile %output\_file%  21. goto done  22. )  23.  24. echo 파일 아키텍처를 확인할 수 없습니다. sigcheck 결과를 확인하세요.  25.  26. :done  27. REM sigcheck 프로세스 종료 후 임시 파일 삭제  28.  29. del temp\_arch.txt  30. echo 작업 완료.  31. | |
| 목적 | Themida Protector 암호화 작동 배치 파일 |
| 파일명 | themida\_encrypt.bat |

|  |  |
| --- | --- |
| 1. from pymongo import MongoClient  2. from datetime import datetime  3. import os  4. from dotenv import load\_dotenv  5.  6. load\_dotenv()  7. #env에서 로드  8. MONGO\_URI=os.getenv("MONGO\_URI")  9. # MongoDB 연결  10. client = MongoClient(MONGO\_URI)  11.  12. # 데이터베이스 선택  13. db = client['normal\_files']  14. collection = db['filedata']  15.  16. # signature\_id 생성 함수  17. def generate\_signature\_id():  18. # 오늘의 날짜 (YYYYMMDD 형식)  19. today = datetime.now().strftime("%Y%m%d")  20.  21. # 오늘 날짜로 시작하는 signature\_id의 개수를 확인하여 번호를 매김  22. count = collection.count\_documents({"signature\_id": {"$regex": f"^{today}-"}})  23. next\_id = count + 1  24.  25. # signature\_id를 YYYYMMDD-번호 형식으로 생성  26. signature\_id = f"{today}-{next\_id:03d}"  27. return signature\_id  28.  29. # 파일 경로 지정  30. #file\_path = r'C:\Users\Administrator\Desktop\sample\_data\PEview.exe'  31. #file\_path = r'C:\Users\Administrator\Desktop\sample\_data\notepad.exe'  32. file\_path = r'C:\Users\Administrator\Desktop\sample\_data\PEview.exe'  33. # 파일명과 확장자 추출  34. filename = os.path.basename(file\_path) # 파일명 추출  35. file\_extension = os.path.splitext(filename)[1] # 확장자 추출 (예: .exe)  36.  37. # 파일을 읽어서 바이너리 데이터로 변환  38. with open(file\_path, 'rb') as file:  39. file\_data = file.read()  40.  41. # signature\_id 생성  42. signature\_id = generate\_signature\_id()  43.  44. # 파일 정보를 MongoDB에 삽입  45. file\_document = {  46. "signature\_id": signature\_id,  47. "filename": filename,  48. "file\_extension": file\_extension, # 확장자 추가  49. "file\_data": file\_data,  50. "upload\_time": datetime.now(),  51. "upload\_ip": "192.168.0.1" # 적절한 IP를 입력하세요  52. }  53.  54. # 데이터 삽입  55. collection.insert\_one(file\_document)  56.  57. print(f"파일이 MongoDB에 성공적으로 업로드되었습니다. Signature ID: {signature\_id}")  58. | |
| 목적 | 모듈 테스트를 위해 DB에 샘플 파일 업로드 하는 모듈 |
| 파일명 | uploadDB\_Sam.py |

|  |  |
| --- | --- |
| 1. import os  2. import gridfs  3. from pymongo import MongoClient  4. from bson import ObjectId  5. from logging\_Utils import setup\_logger  6. from dotenv import load\_dotenv  7.  8. load\_dotenv()  9. #env에서 로드  10. MONGO\_URI=os.getenv("MONGO\_URI")  11. # 로그 설정  12. log\_file = r'C:\pymodules\log\file\_retrieval.log'  13. logger = setup\_logger(log\_file)  14.  15. # MongoDB 연결 설정  16. client = MongoClient(MONGO\_URI)  17. db = client['encrypted\_files'] # 'encrypted\_files' 데이터베이스 사용  18. fs = gridfs.GridFS(db) # GridFS 객체 생성  19.  20. def retrieve\_encrypted\_file(file\_id):  21. """  22. GridFS에서 파일 ID를 사용해 파일을 다운로드합니다.  23. :param file\_id: GridFS에 저장된 파일의 ID (ObjectId)  24. :return: 파일 데이터 (바이너리) 또는 None  25. """  26. try:  27. # GridFS에서 파일 ID를 이용해 파일을 가져옴  28. file\_data = fs.get(file\_id).read() # 파일 데이터를 읽음  29. logger.info(f"File with ID {file\_id} has been retrieved from MongoDB.")  30. return file\_data  31. except Exception as e:  32. logger.error(f"Error retrieving file with ID {file\_id}: {e}", exc\_info=True)  33. return None  34.  35. def save\_file\_to\_local(file\_data, save\_path):  36. """  37. 파일 데이터를 로컬에 저장합니다.  38. :param file\_data: 가져온 파일의 바이너리 데이터  39. :param save\_path: 파일을 저장할 경로  40. """  41. try:  42. # 로컬 경로에 파일 저장  43. with open(save\_path, 'wb') as f:  44. f.write(file\_data)  45. logger.info(f"File has been saved to {save\_path}")  46. print(f"File saved successfully to {save\_path}")  47. except Exception as e:  48. logger.error(f"Error saving file to {save\_path}: {e}", exc\_info=True)  49.  50. if \_\_name\_\_ == "\_\_main\_\_":  51. # 사용자로부터 파일 ID 입력 받음 (실제 MongoDB GridFS에 저장된 파일의 ObjectId)  52. file\_id\_str = input("Enter the GridFS file ID (ObjectId format): ")  53.  54. try:  55. # 입력된 문자열을 ObjectId로 변환  56. file\_id = ObjectId(file\_id\_str)  57. except Exception as e:  58. print(f"Invalid ObjectId format: {e}")  59. logger.error(f"Invalid ObjectId format: {e}")  60. exit(1)  61.  62. # GridFS에서 파일 가져오기  63. file\_data = retrieve\_encrypted\_file(file\_id)  64.  65. if file\_data:  66. # 가져온 파일 데이터를 로컬 파일로 저장  67. save\_path = input("Enter the path where you want to save the file: ")  68. save\_file\_to\_local(file\_data, save\_path)  69. else:  70. print("Failed to retrieve the file from MongoDB.")  71. | |
| 목적 | 모듈 테스트를 위해 DB에서 암호화된 파일 다운로드 하는 모듈 |
| 파일명 | downloadDB\_Enc.py |