

악성코드 탐지, 암호화 및 패키징

파일 프로텍터 개발 명세서

문서번호 : DS-102

VER1.1

```

1. from pymongo import MongoClient
2. import gridfs
3. import downloadDB_Nor # 파일 저장 모듈
4. from logging_Utils import setup_logger # 공통 로그 설정 함수
5. from dotenv import load_dotenv
6. import os
7.
8. # 환경 변수 로드
9. load_dotenv()
10. MONGO_URI = os.getenv("MONGO_URI")
11.
12. # 로그 설정
13. log_file = r'C:\pymodules\log\db_monitor.log'
14. logger = setup_logger(log_file)
15.
16. def monitor_files():
17.     """MongoDB Change Stream을 사용하여 normal_files.filedata 테이블을 모니터링"""
18.     try:
19.         # MongoDB 에 연결
20.         client = MongoClient(MONGO_URI)
21.         db = client['normal_files']
22.         fs = gridfs.GridFS(db) # GridFS 인스턴스 생성
23.
24.         # Change Streams로 normal_files에서 파일 업로드를 감시
25.         with db.filedata.watch() as stream:
26.             logger.info("Started monitoring new files in MongoDB using Change Streams.")
27.             for change in stream:
28.                 if change["operationType"] == "insert":
29.                     file_doc = change["fullDocument"]
30.                     logger.info(f"New file detected: {file_doc['filename']}")
31.
32.                     # GridFS에서 파일 가져오기 (gridfs_file_id를 사용)
33.                     gridfs_file_id = file_doc.get('gridfs_file_id')
34.                     if gridfs_file_id:
35.                         try:
36.                             # GridFS에서 파일 데이터를 가져옴
37.                             grid_out = fs.get(gridfs_file_id)
38.                             file_data = grid_out.read()
39.
40.                             # 로컬에 파일 저장 (파일 저장 모듈에 맞게 수정 필요)
41.                             local_file_path = downloadDB_Nor.store_file(file_doc, file_data)
42.                             if local_file_path:
43.                                 logger.info(f"File {file_doc['filename']} has been saved to {local_file_path}")
44.                             else:
45.                                 logger.error(f"Failed to save file {file_doc['filename']} to local storage.")
46.                         except gridfs.errors.NoFile:
47.                             logger.error(f"File not found in GridFS for file ID: {gridfs_file_id}")
48.                         else:
49.                             logger.error(f"No gridfs_file_id found in the document for file: {file_doc['filename']}")
50.                     else:
51.                         logger.info(f"Ignored operation: {change['operationType']}")
52.             except Exception as e:
53.                 logger.error(f"Error occurred during file monitoring: {e}", exc_info=True)
54.
55. if __name__ == "__main__":
56.     monitor_files()

```

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. import gridfs
10. load_dotenv()
11.
12. #env 에서 로드
13. MONGO_URI=os.getenv("MONGO_URI")
14. # 로그 파일 설정
15. log_file = r'C:\pymodules\log\db_file_store.log'
16. logger = setup_logger(log_file)
17.
18. # MongoDB 연결 설정
19. client = MongoClient(MONGO_URI)
20. db = client['normal_files']
21. collection = db['filedata']
22. fs = gridfs.GridFS(db)
23. def store_file(file_doc, file_data):
24.     """
25.     DB 에서 파일을 다운로드하고 signature_id 로 로컬에 저장 (확장자 필드 사용).
26.     :param file_doc: 파일의 메타데이터 문서
27.     :param file_data: GridFS 에서 가져온 파일의 바이너리 데이터
28.     """
29.     try:
30.         signature_id = file_doc["signature_id"]
31.         file_extension = file_doc.get("file_extension", "")
32.
33.         # 확장자가 없거나 잘못된 형식일 경우 수정
34.         if not file_extension.startswith("."):
35.             file_extension = f".{file_extension}"
36.
37.         # 오늘 날짜를 기준으로 폴더 생성
38.         today_date = datetime.today().strftime('%Y-%m-%d')
39.         local_folder_path = os.path.join(r'C:\DBFiles', today_date, 'origin_files')
40.
41.         # 로컬 폴더가 존재하지 않으면 생성
42.         if not os.path.exists(local_folder_path):
43.             os.makedirs(local_folder_path)
44.
45.         # 파일명을 signature_id 로 하고, 확장자를 적용
46.         local_file_path = os.path.join(local_folder_path,
47. f"{signature_id}{file_extension}")
48.
49.         # 파일 데이터를 로컬에 저장
50.         with open(local_file_path, "wb") as f:
51.             f.write(file_data) # 파일 데이터를 바이너리로 저장
52.         logger.info(f"File with signature_id {signature_id} has been saved to
53. {local_file_path}")
54.
55.         # 3-1: 파일 PE 섹션 분석
56.         pe_info = check_Enc.analyze_pe_sections(file_data, is_path=False)
57.
58.         # 3-2: 섹션 암호화 여부 확인

```

```

58.         encryption_status = check_Enc.check_file_encryption(pe_info)
59.
60.         # 3-3: PE 섹션 정보를 MongoDB 에 저장
61.         uploadDB_NorPE.store_normal_file_pe_info(signature_id, file_doc["filename"],
62.         file_doc["upload_ip"], pe_info, encryption_status)
63.
64.         # 3-4: 파일 암호화 작업 실행
65.         encrypted_file_path = convert_NorToEnc.encrypt_with_themida(local_file_path,
66.         file_doc)
67.         if encrypted_file_path:
68.             logger.info(f"File {local_file_path} has been successfully encrypted.")
69.         else:
70.             logger.error(f"Encryption failed for file {local_file_path}.")
71.         return local_file_path
72.     except Exception as e:
73.         logger.error(f"Error storing file with signature_id {file_doc.get('signature_id',
74.         'unknown')}: {e}", exc_info=True)
75.         print(f"Error storing file with signature_id {file_doc.get('signature_id',
76.         'unknown')}. Check log for details.")
77.         return None

```

목적	DB에 올라온 사용자 파일 다운로드 모듈
파일명	downlaodDB_Nor.py

```

1. import subprocess
2. import os
3. from logging_Utils import setup_logger
4. from datetime import datetime
5. import uploadDB_Enc # MongoDB 에 암호화된 파일을 저장하는 모듈
6. import uploadDB_EncPE # MongoDB 에 암호화된 파일 PE 를 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.     Themida 로 파일 암호화하고, Themida 작업 완료 후 암호화된 파일의 PE 를 분석하여
MongoDB 에 저장
16.     :param input_file: 원본 파일 경로
17.     :param original_file_doc: 원본 파일의 메타데이터 (DB 에서 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, stdout 과 stderr 를 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 MongoDB 에 저장
47.         uploadDB_Enc.store_encrypted_file(output_file, original_file_doc) #
MongoDB 에 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 PE 를 분석하고, MongoDB 에 저장
66.     :param encrypted_file: 암호화된 파일 경로
67.     :param original_file_doc: 원본 파일의 메타데이터 (DB 에서 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: MongoDB 에 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. #env 에서 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) # GridFS 를 이용해 파일을 저장
19.
20. def store_encrypted_file(encrypted_file_path, original_file_doc):
21.     """
22.     Themida 로 보호된 파일을 GridFS 에 저장하고, 관련 메타데이터를 MongoDB 의
    encrypted_files.filedata 컬렉션에 저장.
23.
24.     :param encrypted_file_path: 보호된 파일 경로
25.     :param original_file_doc: 원본 파일의 메타데이터 (MongoDB 에서 가져온 정보)
26.     """
27.     try:
28.         # 암호화된 파일을 읽어서 MongoDB 에 저장
29.         with open(encrypted_file_path, "rb") as f:
30.             encrypted_file_data = f.read()
31.
32.         # GridFS 에 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 # GridFS 에 저장된 파일의 ID
51.     }
52.
53.     # encrypted_files 의 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_Utils 에서 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_utils 의 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: True 면 파일 경로로 처리, False 면 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
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
47.                 "section_alignment": pe.OPTIONAL_HEADER.SectionAlignment # 섹션의
48.                 # 메모리 정렬 값
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. #env 에서 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 MongoDB 에 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.         # MongoDB 에 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.         # MongoDB 에 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. #env 에서 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 MongoDB 에 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.         # MongoDB 에 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.         # MongoDB 에 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.handlers():
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.exe 로 파일 아키텍처 확인
7. C:\pymodules\sigcheck\sigcheck64.exe -a %input_file% > temp_arch.txt
8.
9. REM temp_arch.txt 에서 아키텍처 정보를 읽어옴
10. findstr /i "32-bit" temp_arch.txt > nul
11. if %errorlevel% == 0 (
12.     echo 32 비트 파일입니다. Themida 32 비트 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 64 비트 파일입니다. Themida 64 비트 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. import gridfs
5. from dotenv import load_dotenv
6.
7. # 환경 변수 로드
8. load_dotenv()
9. MONGO_URI = os.getenv("MONGO_URI")
10.
11. # MongoDB 연결
12. client = MongoClient(MONGO_URI)
13.
14. # 데이터베이스 및 GridFS 설정
15. db = client['normal_files']
16. fs = gridfs.GridFS(db) # GridFS 인스턴스 생성
17. collection = db['filedata']
18.
19. # signature_id 생성 함수
20. def generate_signature_id():
21.     # 오늘의 날짜 (YYYYMMDD 형식)
22.     today = datetime.now().strftime("%Y%m%d")
23.
24.     # 오늘 날짜로 시작하는 signature_id의 개수를 확인하여 번호를 매김
25.     count = collection.count_documents({"signature_id": {"$regex": f"^{today}-"}})
26.     next_id = count + 1
27.
28.     # signature_id를 YYYYMMDD-번호 형식으로 생성
29.     signature_id = f"{today}-{next_id:03d}"
30.     return signature_id
31.
32. # 파일 경로 지정
33. file_path = r'C:\Users\Administrator\Desktop\sample_data\Bandizip.exe'
34.
35. # 파일명과 확장자 추출
36. filename = os.path.basename(file_path) # 파일명 추출
37. file_extension = os.path.splitext(filename)[1] # 확장자 추출 (예: .exe)
38.
39. # 파일을 읽어서 바이너리 데이터로 변환
40. with open(file_path, 'rb') as file:
41.     file_data = file.read()
42.
43. # signature_id 생성
44. signature_id = generate_signature_id()
45.
46. # GridFS에 파일 저장
47. gridfs_file_id = fs.put(file_data, filename=filename)
48.
49. # 파일 정보를 MongoDB의 메타데이터 컬렉션에 삽입
50. file_document = {
51.     "signature_id": signature_id,
52.     "filename": filename,
53.     "file_extension": file_extension, # 확장자 추가
54.     "gridfs_file_id": gridfs_file_id, # GridFS 파일 ID
55.     "upload_time": datetime.now(),
56.     "upload_ip": "192.168.0.1" # 적절한 IP를 입력하세요

```



```

57. }
58.
59. # 메타데이터를 MongoDB 컬렉션에 삽입
60. collection.insert_one(file_document)
61.
62. print(f"파일이 GridFS 와 MongoDB 메타데이터 컬렉션에 성공적으로 업로드되었습니다. Signature ID:
{signature_id}")
63.

```

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. #env 에서 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.     GridFS 에서 파일 ID 를 사용해 파일을 다운로드합니다.
23.     :param file_id: GridFS 에 저장된 파일의 ID (ObjectId)
24.     :return: 파일 데이터 (바이너리) 또는 None
25.     """
26.     try:
27.         # GridFS 에서 파일 ID 를 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 GridFS 에 저장된 파일의 ObjectId)
52.     file_id_str = input("Enter the GridFS file ID (ObjectId format): ")
53.
54.     try:
55.         # 입력된 문자열을 ObjectId 로 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.     # GridFS 에서 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