

악성코드 탐지, 암호화 및 패키징

# 악성코드 탐지 개발 명세서

문서번호 : DS-101

VER1.1

```

1. import time
2. import json
3. import re
4. from hash_handler import read_hashes_from_csv, load_processed_hashes,
save_processed_hashes
5. from virustotal_API import search_file_by_hash
6. from data_converter import convert_data
7. from DB_handler import upload_to_mongodb
8. from logger_manager import setup_logger
9.
10. # 로그 설정
11. logger = setup_logger(r'C:\VTAPImodules\log\process_hash',
r'C:\VTAPImodules\log\process_hash.log')
12.
13. # 설정
14. MAX_EXECUTIONS_PER_MINUTE = 2 # 분당 최대 2 번 실행
15. MAX_EXECUTIONS_PER_DAY = 250 # 하루 최대 250 번 실행
16.
17. # 분당 5 번 실행을 위한 제한 함수
18. def rate_limiter():
19.     time.sleep(30) # 30 초마다 1 번 실행 -> 분당 2 번 실행 가능
20.     logger.info("Rate limiter: 30 초 대기")
21.
22. # MD5 해시가 유효한지 확인하는 함수
23. def is_valid_md5(md5_hash):
24.     """MD5 해시가 32 자리의 16 진수인지 확인"""
25.     valid = re.match(r"^[a-fA-F0-9]{32}$", md5_hash) is not None
26.     if not valid:
27.         logger.warning(f"유효하지 않은 MD5 해시: {md5_hash}")
28.     return valid
29.
30. def process_hash(hash_value, processed_hashes):
31.     if hash_value in processed_hashes:
32.         logger.info(f"{hash_value} 이미 처리됨. 스킵합니다.")
33.         return False # 이미 처리된 경우 False 반환
34.
35.     # 유효한 MD5 해시인지 확인
36.     if not is_valid_md5(hash_value):
37.         logger.warning(f"{hash_value} 유효하지 않은 MD5 해시입니다. 스킵합니다.")
38.         return False # 유효하지 않은 해시는 처리하지 않음
39.
40.     # VirusTotal API 호출
41.     details = search_file_by_hash(hash_value)
42.     if details is None:
43.         logger.error(f"{hash_value} 처리 실패. 파일을 찾을 수 없습니다.")
44.         return False
45.
46.     behavior = search_file_by_hash(hash_value, "behaviour_summary")
47.     if behavior is None:
48.         logger.warning(f"{hash_value} 행동 분석 정보를 찾을 수 없습니다.")
49.
50.     if details:
51.         # 데이터 변환
52.         logger.info(f"{hash_value} 데이터 변환 중...")

```

```

53.     converted_data = convert_data(details, behavior)
54.
55.     # MongoDB 에 업로드
56.     upload_to_mongodb(converted_data, "info")
57.     logger.info(f"{hash_value} MongoDB 에 저장 완료.")
58.
59.     # 처리된 해시 기록
60.     processed_hashes.append(hash_value)
61.     save_processed_hashes(processed_hashes)
62.     logger.info(f"{hash_value} 처리된 해시 기록에 추가.")
63.     return True # 성공적으로 처리된 경우 True 반환
64. else:
65.     logger.error(f"{hash_value} 처리 실패.")
66.     return False # 처리 실패 시 False 반환
67.
68.     rate_limiter()
69.
70. if __name__ == "__main__":
71.     logger.info("처리된 해시 로드 시작")
72.
73.     # 처리된 해시 로드
74.     processed_hashes = load_processed_hashes()
75.
76.     # 해시값 처리
77.     execution_count = 0
78.     hashes = read_hashes_from_csv()
79.
80.     for hash_value in hashes:
81.         if execution_count >= MAX_EXECUTIONS_PER_DAY:
82.             logger.info("오늘의 최대 실행 횟수에 도달했습니다.")
83.             break
84.
85.         if process_hash(hash_value, processed_hashes):
86.             execution_count += 1 # 실제로 처리된 경우에만 증가
87.

```

목적	데이터셋에서 Virustotal API를 통해 나온 정보로 DB에 업로드 하기 위한 모듈
파일명	process_hash.py

```

1. import json
2. from logger_manager import setup_logger
3. import csv
4.
5. # 로그 설정
6. logger = setup_logger(r'C:\VTAPImodules\log\hash_handler',
r'C:\VTAPImodules\log\hash_handler.log')
7.
8. PROCESSED_HASHES_FILE = r'C:\VTAPImodules\processed_hashes.json'
9.
10. def read_hashes_from_csv():
11.     """CSV 파일에서 해시 값을 읽어온다."""
12.     csv_file = r"C:\VTAPImodules\dataset.csv"
13.     try:
14.         with open(csv_file, 'r', encoding='ISO-8859-1') as file:
15.             reader = csv.DictReader(file)
16.             hashes = [row['md5_hash'].strip() for row in reader] # CSV 에서 'md5_hash'
열 읽기
17.             logger.info(f"CSV 파일 {csv_file}에서 해시 값 로드 성공")
18.             return hashes
19.     except FileNotFoundError:
20.         logger.error(f"CSV 파일 {csv_file}을 찾을 수 없음")
21.         return []
22.     except Exception as e:
23.         logger.error(f"CSV 파일 읽기 중 오류 발생: {e}")
24.         return []
25.
26. def load_processed_hashes():
27.     try:
28.         with open(PROCESSED_HASHES_FILE, 'r') as f:
29.             logger.info("처리된 해시 로드 성공")
30.             return json.load(f)
31.     except FileNotFoundError:
32.         logger.error("처리된 해시 파일을 찾을 수 없음")
33.         return []
34.     except json.JSONDecodeError:
35.         logger.error("처리된 해시 파일을 읽는 중 JSON 오류 발생")
36.         return []
37.
38. def save_processed_hashes(processed_hashes):
39.     try:
40.         with open(PROCESSED_HASHES_FILE, 'w') as f:
41.             json.dump(processed_hashes, f)
42.             logger.info("처리된 해시 기록 저장 성공")
43.     except Exception as e:
44.         logger.error(f"처리된 해시 저장 중 오류 발생: {e}")
45.

```

목적	csv와 json에서 해시를 읽거나 저장하는 모듈
파일명	hash_handler.py

```

1. import time
2. from pymongo import MongoClient
3. import os
4. from dotenv import load_dotenv
5. import requests
6. import json
7. from logger_manager import setup_logger
8. import gridfs
9.
10. # 로그 설정
11. logger = setup_logger(r'C:\VTAPImodules\log\DB_handler',
r'C:\VTAPImodules\log\DB_handler.log')
12.
13. load_dotenv()
14. MONGO_URI = os.getenv('MONGO_URI')
15. DB_NAME = "vsapi"
16. FILES_COLLECTION = "file"
17. INFO_COLLECTION = "info"
18. client = MongoClient(MONGO_URI)
19. db = client[DB_NAME]
20. fs = gridfs.GridFS(db)
21.
22. def watch_for_file_uploads():
23.     collection = db[FILES_COLLECTION]
24.     logger.info("파일 업로드 감시 시작")
25.     with collection.watch() as stream:
26.         for change in stream:
27.             if change["operationType"] == "insert":
28.                 file_data = change["fullDocument"]
29.                 file_hash = file_data.get('filehash') # MD5 해시 가져오기
30.                 gridfs_file_id = file_data.get('gridfs_file_id') # GridFS 파일 ID 가져오기
31.
32.                 if file_hash and gridfs_file_id:
33.                     logger.info(f"새로운 파일 업로드 감지: {file_hash}, GridFS 파일 ID:
{gridfs_file_id}")
34.
35.                     # GridFS 에서 파일 가져오기 (gridfs_file_id 를 사용하여 파일 검색)
36.                     try:
37.                         grid_out = fs.get(gridfs_file_id)
38.                         logger.info(f"GridFS 에서 파일 {file_hash} 로드 완료")
39.                         return file_hash, grid_out.read() # 파일의 바이너리 데이터 반환
40.                     except gridfs.errors.NoFile:
41.                         logger.error(f"GridFS 에 {gridfs_file_id}에 해당하는 파일이 없습니다.
(file_hash: {file_hash})")
42.                         return None, None
43.                     else:
44.                         logger.error(f"file_hash 또는 gridfs_file_id 가 존재하지 않습니다.
file_hash: {file_hash}, gridfs_file_id: {gridfs_file_id}")
45.                         return None, None
46.
47.
48.
49.
50. # MongoDB 업로드
51. def upload_to_mongodb(data, collection_name=FILES_COLLECTION):
52.     collection = db[collection_name]
53.     try:
54.         collection.insert_one(data)
55.         logger.info(f"Data uploaded to MongoDB: {collection_name}")
56.     except Exception as e:
57.         logger.error(f"MongoDB 업로드 실패: {e}")
58.

```

```

59. # 해시가 MongoDB 'info' 컬렉션에 있는지 확인하는 함수
60. def check_hash_in_mongodb(file_hash):
61.     collection = db[INFO_COLLECTION]
62.     try:
63.         existing_data = collection.find_one({"md5": file_hash})
64.         if existing_data:
65.             logger.info(f"{file_hash} 이미 MongoDB 'info' 컬렉션에 존재합니다.")
66.             return existing_data
67.         else:
68.             logger.info(f"{file_hash} MongoDB 'info' 컬렉션에 존재하지 않음.")
69.     except Exception as e:
70.         logger.error(f"MongoDB 해시 검사 실패: {e}")
71.     return None
72.

```

목적	DB에 올라온 파일을 감시, DB에 업로드, DB에 해당 해시 값이 있는지 확인하는 모듈
파일명	DB_handler.py

```

1. import json
2. import os
3. from logger_manager import setup_logger
4.
5. # 로그 설정
6. logger = setup_logger(r'C:\VTAPImodules\log\data_converter',
r'C:\VTAPImodules\log\data_converter.log')
7.
8. def convert_data(details, behavior):
9.     logger.info("데이터 변환 시작")
10.
11.     try:
12.         template_file_path = r"C:\VTAPImodules\template.json"
13.         with open(template_file_path, 'r', encoding='utf-8') as template_file:
14.             json_data = json.load(template_file)
15.
16.             detail_data_attribute = details.get("data", {}).get("attributes", {})
17.             json_data["md5"] = detail_data_attribute.get("md5", None)
18.
19.             # 해시 정보
20.             json_data["details"]["hash"]["md5"] = detail_data_attribute.get("md5", None)
21.             json_data["details"]["hash"]["sha1"] = detail_data_attribute.get("sha1",
None)
22.             json_data["details"]["hash"]["sha256"] = detail_data_attribute.get("sha256",
None)
23.             json_data["details"]["hash"]["vhash"] = detail_data_attribute.get("vhash",
None)
24.             json_data["details"]["hash"]["auth_hash"] =
detail_data_attribute.get("authentihash", None)
25.             json_data["details"]["hash"]["imphash"] =
detail_data_attribute.get("pe_info", {}).get("imphash", None)
26.             json_data["details"]["hash"]["ssdeep"] = detail_data_attribute.get("ssdeep",
None)
27.             json_data["details"]["hash"]["tlsh"] = detail_data_attribute.get("tlsh",
None)
28.
29.             # 파일 정보
30.             json_data["details"]["file_info"]["md5"] = detail_data_attribute.get("md5",
None)
31.             json_data["details"]["file_info"]["file_type"] =
detail_data_attribute.get("type_tags", None)
32.             json_data["details"]["file_info"]["magic"] =
detail_data_attribute.get("magic", None)
33.             json_data["details"]["file_info"]["file_size"] =
detail_data_attribute.get("size", None)
34.             json_data["details"]["file_info"]["PEID_packer"] =
detail_data_attribute.get("packers", {}).get("PEiD", None)
35.             json_data["details"]["file_info"]["first_seen_time"] =
detail_data_attribute.get("first_submission_date", None)
36.
37.             # 이름 정보
38.             json_data["details"]["file_info"]["name"] =
detail_data_attribute.get("names", None)
39.
40.             # 시그니처 정보
41.             json_data["details"]["signature"] =
detail_data_attribute.get("signature_info", {})
42.
43.             json_data["details"]["pe_info"] = detail_data_attribute.get("pe_info", {})
44.             json_data["details"]["dot_net_assembly"] =
detail_data_attribute.get("dot_net_assembly", {})

```

```

45.
46.     if behavior is None or behavior.get("data") is None:
47.         logger.info("Behavior 데이터가 없음")
48.         return json_data
49.
50.     # MITRE 공격 기법 정보 추가
51.     mitre_techniques = behavior.get("data", {}).get("mitre_attack_techniques",
52.     {})
53.     for technique in mitre_techniques:
54.         technique_id = technique.get("id")
55.         description = technique.get("signature_description")
56.         severity = technique.get("severity", "")
57.
58.         json_data["behavior"]["mitre"][technique_id] = {
59.             "description": description,
60.             "severity": severity
61.         }
62.
63.     # 행동 정보 추가
64.     json_data["behavior"]["modules_loaded"] = behavior.get("data",
65.     {}).get("modules_loaded", {})
66.     json_data["behavior"]["tags"] = behavior.get("data", {}).get("tags", {})
67.
68.     # Capabilities 처리
69.     capabilities_comms = behavior.get("data", {}).get("signature_matches", {})
70.     for comm in capabilities_comms:
71.         if comm.get("format") == "SIG_FORMAT_CAPA":
72.             capa_name = comm.get("name")
73.             description = comm.get("description")
74.             authors = comm.get("authors")
75.             rule_src = comm.get("rule_src")
76.             refs = comm.get("refs", {})
77.             json_data["behavior"]["Capabilities"][capa_name] = {
78.                 "authors": authors,
79.                 "description": description,
80.                 "rule": rule_src,
81.                 "refs": refs
82.             }
83.
84.     # 네트워크 통신 정보
85.     network_communications = ["ja3_digests", "http_conversations",
86.     "memory_pattern_domains", "memory_pattern_urls",
87.     "memory_pattern_ips", "tls"]
88.     for net_comm in network_communications:
89.         col = behavior.get("data", {}).get(net_comm, {})
90.         json_data["behavior"]["network_communications"][net_comm] = col
91.
92.     # 파일 작업 정보
93.     file_actions = ["files_opened", "files_written", "files_deleted",
94.     "files_attribute_changed", "files_dropped"]
95.     for file_comm in file_actions:
96.         col = behavior.get("data", {}).get(file_comm, {})
97.         json_data["behavior"]["file_system_actions"][file_comm] = col
98.
99.     # 레지스트리 작업 정보
100.     registry_actions = ["registry_keys_opened", "registry_keys_set",
    "registry_keys_deleted"]
101.     for reg_comm in registry_actions:
102.         col = behavior.get("data", {}).get(reg_comm, {})
103.         json_data["behavior"]["registry_actions"][reg_comm] = col

```



```

101.         # 프로세스 및 서비스 작업 정보
102.         process_and_service_actions = ["processes_created", "command_executions",
103.         "processes_injected",
104.         "processes_terminated", "services_opened",
105.         "processes_tree"]
106.         for pas_comm in process_and_service_actions:
107.             col = behavior.get("data", {}).get(pas_comm, {})
108.             json_data["behavior"]["process_and_service_actions"][pas_comm] = col
109.
110.         # 동기화 메커니즘 정보
111.         synchronization_mechanisms_signals = ["mutexes_created", "mutexes_opened"]
112.         for sms_comm in synchronization_mechanisms_signals:
113.             col = behavior.get("data", {}).get(sms_comm, {})
114.             json_data["behavior"]["synchronization_mechanisms_signals"][sms_comm] =
115.             col
116.
117.         # 강조된 작업 정보
118.         highlighted_actions = ["calls_highlighted", "text_decoded"]
119.         for high_comm in highlighted_actions:
120.             col = behavior.get("data", {}).get(high_comm, {})
121.             json_data["behavior"]["highlighted_actions"][high_comm] = col
122.
123.         # 시스템 속성 조회
124.         system_property_lookups = behavior.get("data",
125.         {}).get("system_property_lookups", {})
126.         json_data["behavior"]["system_property_lookups"] = system_property_lookups
127.
128.         logger.info("데이터 변환 완료")
129.         return json_data
130.
131.     except Exception as e:
132.         logger.error(f"데이터 변환 중 오류 발생: {e}")
133.         return None

```

목적	VirusTotal API를 돌려나온 Details, Behavior를 template에 맞게 변환하는 모듈
파일명	data_converter.py

```

1. import requests
2. import time
3. import os
4. from dotenv import load_dotenv
5. from logger_manager import setup_logger
6.
7. # 로그 설정
8. logger = setup_logger(r'C:\VTAPImodules\log\virustotal_api',
r'C:\VTAPImodules\log\virustotal_api.log')
9.
10. load_dotenv()
11. API_KEY = os.getenv('VT_API_KEY')
12.
13. # VirusTotal 에서 해시로 검색하는 함수 (재시도 포함)
14. def search_file_by_hash_with_retry(file_hash, retries=5, wait_time=60):
15.     for i in range(retries):
16.         logger.info(f"해시 검색 시도 {i + 1}/{retries}: {file_hash}")
17.
18.         response_json = search_file_by_hash(file_hash)
19.         if response_json:
20.             logger.info(f"{file_hash} 해시 검색 성공")
21.             return response_json
22.         else:
23.             logger.info(f"{file_hash} 파일이 아직 처리되지 않았습니다. {wait_time}초 대기
후 다시 시도합니다...")
24.             time.sleep(wait_time)
25.         logger.error(f"{file_hash} 최대 재시도 횟수 초과")
26.     return None
27.
28. # VirusTotal 해시 검색
29. def search_file_by_hash(hash_value, endpoint=""):
30.     url = f"https://www.virustotal.com/api/v3/files/{hash_value}"
31.     if endpoint:
32.         url = f"https://www.virustotal.com/api/v3/files/{hash_value}/{endpoint}"
33.
34.     headers = {"accept": "application/json", "x-apikey": API_KEY}
35.     response = requests.get(url, headers=headers)
36.
37.     if response.status_code == 200: # 성공 시
38.         logger.info(f"{hash_value} 해시 검색 성공")
39.         return response.json()
40.     else:
41.         logger.error(f"{hash_value} 해시 검색 실패: {response.status_code}")
42.         return None
43.
44. # VirusTotal 파일 업로드
45. def upload_file_to_virustotal(file_data):
46.     logger.info("VirusTotal 파일 업로드 시작")
47.
48.     upload_url_response =
requests.get("https://www.virustotal.com/api/v3/files/upload_url", headers={"x-apikey":
API_KEY})
49.
50.     if upload_url_response.status_code == 200:
51.         upload_url = upload_url_response.json().get('data', None)
52.
53.         if upload_url:
54.             files = {'file': file_data}

```

```

55.         upload_response = requests.post(upload_url, files=files, headers={"x-
apikey": API_KEY})
56.
57.         if upload_response.status_code == 200:
58.             logger.info("파일 업로드 성공")
59.             return True
60.         else:
61.             logger.error(f"파일 업로드 실패: {upload_response.status_code} -
{upload_response.text}")
62.         else:
63.             logger.error("업로드 URL 획득 실패")
64.     else:
65.         logger.error(f"업로드 URL 요청 실패: {upload_url_response.status_code} -
{upload_url_response.text}")
66.
67.     return None
68.

```

목적	VirusTotal API를 활용해 파일을 업로드, Details, Behavior 정보를 불러오는 모듈
파일명	virustotal_api.py

```

1. from DB_handler import watch_for_file_uploads, upload_to_mongodb, check_hash_in_mongodb
2. from virustotal_API import search_file_by_hash_with_retry, upload_file_to_virustotal,
search_file_by_hash
3. from hash_handler import save_processed_hashes, load_processed_hashes
4. from data_converter import convert_data
5. from logger_manager import setup_logger
6. import time
7. # 로그 설정
8. logger = setup_logger(r'C:\VTAPImodules\log\process_new_file',
r'C:\VTAPImodules\log\process_new_file.log')
9.
10. # 새 파일이 업로드되었을 때 VirusTotal 에서 처리하는 함수
11. def process_new_file(file_hash, file_data, processed_hashes):
12.     logger.info(f"새로운 파일 업로드 감지: {file_hash}")
13.
14.     # DB 에 hash 가 있는 경우 종료
15.     if check_hash_in_mongodb(file_hash):
16.         logger.info(f"{file_hash} 이미 DB 에 존재함. 처리 종료.")
17.         return
18.
19.     # VirusTotal 에서 해시 검색
20.     details = search_file_by_hash(file_hash)
21.     if not details:
22.         logger.info(f"{file_hash} VirusTotal 에 없음, 파일 업로드 중...")
23.         flag = upload_file_to_virustotal(file_data) # 파일 내용 업로드
24.         if flag:
25.             # 파일 해시로 검색 (없으면 반복)
26.             search_result = search_file_by_hash_with_retry(file_hash)
27.             if search_result:
28.                 details = search_result
29.                 behavior = search_file_by_hash(file_hash, "behaviour_summary")
30.         else:
31.             logger.info(f"{file_hash} 이미 VirusTotal 에 존재함. 데이터 가져오는 중...")
32.             behavior = search_file_by_hash(file_hash, "behaviour_summary")
33.
34.     # 데이터 변환 후 DB 에 추가
35.     if details:
36.         logger.info(f"{file_hash} 데이터 변환 중...")
37.         converted_data = convert_data(details, behavior)
38.
39.         # MongoDB 에 업로드
40.         upload_to_mongodb(converted_data, 'info')
41.         # 처리된 해시 기록 추가
42.         processed_hashes.append(file_hash)
43.         save_processed_hashes(processed_hashes)
44.         logger.info(f"{file_hash} 분석 및 행동 분석 완료 후 MongoDB 에 저장.")
45.     else:
46.         logger.error(f"{file_hash} 처리 실패.")
47.
48. if __name__ == "__main__":
49.     logger.info("파일 업로드 감시 시작")
50.     filehash, filedata = watch_for_file_uploads()
51.     processed_hashes = load_processed_hashes()
52.
53.     if filehash:
54.         process_new_file(filehash, filedata, processed_hashes)
55.         filehash = None
56.

```

목적	사용자가 DB에 파일을 업로드하면 해당 파일 정보를 DB에 저장해주는 모듈
파일명	process_new_file.py

```

1. import logging
2. import os
3.
4. def setup_logger(logger_name, log_file, level=logging.INFO):
5.
6.
7.     logger = logging.getLogger(logger_name)
8.     logger.setLevel(level)
9.
10.    # 파일 핸들러 설정 (UTF-8 인코딩 추가)
11.    file_handler = logging.FileHandler(log_file, encoding='utf-8')
12.    file_handler.setFormatter(logging.Formatter('%(asctime)s - %(levelname)s
- %(message)s'))
13.
14.    if not logger.handlers():
15.        logger.addHandler(file_handler)
16.
17.    return logger
18.

```

목적	모듈 동작 로그를 관리하기 위한 모듈
파일명	logger_manager.py

```

1. {
2.     "md5": "",
3.     "details": {
4.         "hash": {
5.             "md5": "",
6.             "sha1": "",
7.             "sha256": "",
8.             "vhash": "",
9.             "auth_hash": "",
10.            "imphash": "",
11.            "ssdeep": "",
12.            "tlsh": ""
13.        },
14.        "file_info": {
15.            "md5": "",
16.            "file_type": "",
17.            "magic": "",
18.            "file_size": "",
19.            "PEID_packer": "",
20.            "first_seen_time": "",
21.            "name": ""
22.        },
23.        "signature": {
24.        },
25.        "pe_info": {
26.        },
27.        "dot_net_assembly": {
28.        }
29.    },
30.    "behavior": {
31.        "mitre": {
32.
33.        },
34.        "Capabilities": {
35.
36.        },
37.        "tags": {
38.
39.        },
40.        "network_communications": {
41.            "http_conversations": {},
42.            "ja3_digests": {},
43.            "memory_pattern_domains": {},
44.            "memory_pattern_ips": {},
45.            "memory_pattern_urls": {},
46.            "tls": {}
47.        },
48.        "file_system_actions": {
49.            "files_opened": {},
50.            "files_written": {},
51.            "files_deleted": {},
52.            "files_attribute_changed": {},
53.            "files_dropped": {}
54.        },
55.        "registry_actions": {
56.            "registry_keys_opened": {},
57.            "registry_keys_set": {},
58.            "registry_keys_deleted": {}
59.        },
60.        "process_and_service_actions": {
61.            "processes_created": {},
62.            "command_executions": {},
63.            "processes_injected": {},
64.            "processes_terminated": {},

```

```

65.         "services_opened": {},
66.         "processes_tree": {}
67.     },
68.     "synchronization_mechanisms_signals": {
69.         "mutexes_created": {},
70.         "mutexes_opened": {}
71.     },
72.     "modules_loaded": {
73.     },
74.     "highlighted_actions": {
75.         "calls_highlighted": {},
76.         "text_decoded": {}
77.     },
78.     "system_property_lookups": {}
79.
80. }
81. }
82.

```

목적	DB에 저장할 구조 template
파일명	template.json



```

1. import hashlib
2. from pymongo import MongoClient
3. import gridfs
4. import os
5. from dotenv import load_dotenv
6. from datetime import datetime
7.
8. # .env 파일에서 MongoDB URI 로드
9. load_dotenv()
10. MONGO_URI = os.getenv('MONGO_URI')
11. DB_NAME = "vsapi"
12. FILES_COLLECTION = "file"
13.
14. # MongoDB 연결 및 GridFS 인스턴스 생성
15. client = MongoClient(MONGO_URI)
16. db = client[DB_NAME]
17. fs = gridfs.GridFS(db) # GridFS 인스턴스 생성
18. collection = db[FILES_COLLECTION]
19.
20. # MD5 해시 계산 함수
21. def calculate_md5(file_path):
22.     hash_md5 = hashlib.md5()
23.     with open(file_path, "rb") as f:
24.         while chunk := f.read(8192): # 파일을 8192 바이트씩 읽어서 해시 계산
25.             hash_md5.update(chunk)
26.     return hash_md5.hexdigest()
27.
28. # signature_id 생성 함수
29. def generate_signature_id():
30.     # 오늘의 날짜 (YYYYMMDD 형식)
31.     today = datetime.now().strftime("%Y%m%d")
32.
33.     # 오늘 날짜로 시작하는 signature_id의 개수를 확인하여 번호를 매김
34.     count = collection.count_documents({"signature_id": {"$regex": f"^{today}-"}})
35.     next_id = count + 1
36.
37.     # signature_id를 YYYYMMDD-번호 형식으로 생성
38.     signature_id = f"{today}-{next_id:03d}"
39.     return signature_id
40.
41. # 파일 업로드 함수 (GridFS 적용)
42. def upload_file_to_mongodb(file_path, upload_ip):
43.     filehash = calculate_md5(file_path)
44.     filename = os.path.basename(file_path)
45.
46.     with open(file_path, 'rb') as f:
47.         file_data = f.read()
48.
49.     # 고유한 signature_id 생성
50.     signature_id = generate_signature_id()
51.
52.     # GridFS 에 파일 데이터 저장
53.     gridfs_file_id = fs.put(file_data, filename=filename, filehash=filehash)
54.
55.     # 파일 메타데이터 저장 (GridFS 에서 파일 ID 저장)
56.     file_metadata = {
57.         "signature_id": signature_id,
58.         "filehash": filehash,
59.         "filename": filename,
60.         "gridfs_file_id": gridfs_file_id, # GridFS 파일 ID
61.         "upload_time": datetime.now(),

```

```

62.         "upload_ip": upload_ip
63.     }
64.
65.     # MongoDB 의 file 컬렉션에 메타데이터 저장
66.     collection.insert_one(file_metadata)
67.     print(f"파일 {filename}가 MongoDB 에 GridFS 로 업로드되었습니다. MD5: {filehash},
Signature ID: {signature_id}")
68.
69. # 테스트 파일 경로와 IP
70. test_file_path = r"C:\Users\Administrator\Desktop\sample_data\Bandizip_protected.exe"
71. test_upload_ip = "192.168.0.1"
72.
73. # 파일을 MongoDB 에 업로드
74. upload_file_to_mongodb(test_file_path, test_upload_ip)
75.

```

목적	모듈 테스트를 위한 DB 파일 업로드 모듈
파일명	uploadDB_sam.py