악성코드 탐지, 암호화 및 패키징

악성코드 탐지 개발 명세서

문서번호 : DS-101

VER1.0

|  |  |
| --- | --- |
| 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 2번 실행  15. MAX\_EXECUTIONS\_PER\_DAY = 250 # 하루 최대 250번 실행  16.  17. # 분당 5번 실행을 위한 제한 함수  18. def rate\_limiter():  19. time.sleep(30) # 30초마다 1번 실행 -> 분당 2번 실행 가능  20. logger.info("Rate limiter: 30초 대기")  21.  22. # MD5 해시가 유효한지 확인하는 함수  23. def is\_valid\_md5(md5\_hash):  24. """MD5 해시가 32자리의 16진수인지 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. # MongoDB에 업로드  56. upload\_to\_mongodb(converted\_data, "info")  57. logger.info(f"{hash\_value} MongoDB에 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 Virustoatal 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] # CSV에서 '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. # GridFS에서 파일 가져오기 (gridfs\_file\_id를 사용하여 파일 검색)  36. try:  37. grid\_out = fs.get(gridfs\_file\_id)  38. logger.info(f"GridFS에서 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"GridFS에 {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\_id가 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. for technique in mitre\_techniques:  53. technique\_id = technique.get("id")  54. description = technique.get("signature\_description")  55. severity = technique.get("severity", "")  56.  57. json\_data["behavior"]["mitre"][technique\_id] = {  58. "description": description,  59. "severity": severity  60. }  61.  62. # 행동 정보 추가  63. json\_data["behavior"]["modules\_loaded"] = behavior.get("data", {}).get("modules\_loaded", {})  64. json\_data["behavior"]["tags"] = behavior.get("data", {}).get("tags", {})  65.  66. # Capabilities 처리  67. capabilities\_comms = behavior.get("data", {}).get("signature\_matches", {})  68. for comm in capabilities\_comms:  69. if comm.get("format") == "SIG\_FORMAT\_CAPA":  70. capa\_name = comm.get("name")  71. description = comm.get("description")  72. authors = comm.get("authors")  73. rule\_src = comm.get("rule\_src")  74. refs = comm.get("refs", {})  75. json\_data["behavior"]["Capabilities"][capa\_name] = {  76. "authors": authors,  77. "description": description,  78. "rule": rule\_src,  79. "refs": refs  80. }  81.  82. # 네트워크 통신 정보  83. network\_communications = ["ja3\_digests", "http\_conversations", "memory\_pattern\_domains", "memory\_pattern\_urls",  84. "memory\_pattern\_ips", "tls"]  85. for net\_comm in network\_communications:  86. col = behavior.get("data", {}).get(net\_comm, {})  87. json\_data["behavior"]["network\_communications"][net\_comm] = col  88.  89. # 파일 작업 정보  90. file\_actions = ["files\_opened", "files\_written", "files\_deleted", "files\_attribute\_changed", "files\_dropped"]  91. for file\_comm in file\_actions:  92. col = behavior.get("data", {}).get(file\_comm, {})  93. json\_data["behavior"]["file\_system\_actions"][file\_comm] = col  94.  95. # 레지스트리 작업 정보  96. registry\_actions = ["registry\_keys\_opened", "registry\_keys\_set", "registry\_keys\_deleted"]  97. for reg\_comm in registry\_actions:  98. col = behavior.get("data", {}).get(reg\_comm, {})  99. json\_data["behavior"]["registry\_actions"][reg\_comm] = col  100.  101. # 프로세스 및 서비스 작업 정보  102. process\_and\_service\_actions = ["processes\_created", "command\_executions", "processes\_injected",  103. "processes\_terminated", "services\_opened", "processes\_tree"]  104. for pas\_comm in process\_and\_service\_actions:  105. col = behavior.get("data", {}).get(pas\_comm, {})  106. json\_data["behavior"]["process\_and\_service\_actions"][pas\_comm] = col  107.  108. # 동기화 메커니즘 정보  109. synchronization\_mechanisms\_signals = ["mutexes\_created", "mutexes\_opened"]  110. for sms\_comm in synchronization\_mechanisms\_signals:  111. col = behavior.get("data", {}).get(sms\_comm, {})  112. json\_data["behavior"]["synchronization\_mechanisms\_signals"][sms\_comm] = col  113.  114. # 강조된 작업 정보  115. highlighted\_actions = ["calls\_highlighted", "text\_decoded"]  116. for high\_comm in highlighted\_actions:  117. col = behavior.get("data", {}).get(high\_comm, {})  118. json\_data["behavior"]["highlighted\_actions"][high\_comm] = col  119.  120. # 시스템 속성 조회  121. system\_property\_lookups = behavior.get("data", {}).get("system\_property\_lookups", {})  122. json\_data["behavior"]["system\_property\_lookups"] = system\_property\_lookups  123.  124. logger.info("데이터 변환 완료")  125. return json\_data  126.  127. except Exception as e:  128. logger.error(f"데이터 변환 중 오류 발생: {e}")  129. return None  130. | |
| 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. # VirusTotal에서 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 VirusTotal에서 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. # DB에 hash가 있는 경우 종료  15. if check\_hash\_in\_mongodb(file\_hash):  16. logger.info(f"{file\_hash} 이미 DB에 존재함. 처리 종료.")  17. return  18.  19. # VirusTotal에서 해시 검색  20. details = search\_file\_by\_hash(file\_hash)  21. if not details:  22. logger.info(f"{file\_hash} VirusTotal에 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 VirusTotal에 존재함. 데이터 가져오는 중...")  32. behavior = search\_file\_by\_hash(file\_hash, "behaviour\_summary")  33.  34. # 데이터 변환 후 DB에 추가  35. if details:  36. logger.info(f"{file\_hash} 데이터 변환 중...")  37. converted\_data = convert\_data(details, behavior)  38.  39. # MongoDB에 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 MongoDB에 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.hasHandlers():  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. # GridFS에 파일 데이터 저장  53. gridfs\_file\_id = fs.put(file\_data, filename=filename, filehash=filehash)  54.  55. # 파일 메타데이터 저장 (GridFS에서 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. # MongoDB의 file 컬렉션에 메타데이터 저장  66. collection.insert\_one(file\_metadata)  67. print(f"파일 {filename}가 MongoDB에 GridFS로 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 MongoDB에 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 |