



제출일	2023.05.03	학과	컴퓨터공학전공
과목	컴퓨터보안	학번	2018112007
담당교수	김영부 교수님	이름	이승현



## 1) 실습 환경

(1)

운영 체제: Microsoft Windows 11 Home 64bit

프로세서 : Intel(R) Core(TM) i7-10510U @ 1.80GHz (8 CPUs), ~ 2.3GHz

메모리 : DDR4 16GB 2,667MHz

그래픽 카드 : Intel UHD Graphics

(2)

운영 체제: Microsoft Windows 10 Home 64bit

프로세서 : Intel(R) Core(TM) i7-7700HQ @ 2.80GHz (8 CPUs), ~ 2.8GHz

메모리 : DDR4 8GB 2,133MHz

그래픽 카드 : Intel HD Graphics 630, NVIDIA GeForce GTX 1050

## 2) 실습 진행

### 1. 문제 분석

#### 1) 전용 백신 개발하기

- 전용 백신은 특정한 악성코드 하나를 진단 및 치료하는 백신을 의미한다.
- 전용 백신은 일반적인 백신보다 속도 측면에서 빠르고 진단 측면에서 정확하다는 장점이 있다.
- 이번 실습에서는 EICAR Text 파일을 진단 및 치료할 수 있는 EICAR 전용 백신을 만들어본다.
- 백신 개발에는 분할과 정복 기법을 적용하여 작은 프로젝트부터 공략하여 큰 프로젝트로 확장한다.

#### 2) 다양한 악성코드 진단 및 치료하기

- 전용 백신은 사회적으로 이슈가 되는 특정 악성코드가 자신의 시스템에 존재하는지 확인하기에는 편리하지만, 시스템 전반의 일반적인 보안 점검에는 사용하기 어렵다.
- 이번 실습에서는 새로운 악성코드 Dummy Test를 정의하고, 전용 백신을 수정해서 다양한 악성코드를 진단 및 치료할 수 있는 백신을 만들어본다.

#### 3) 악성코드 패턴 분리하기

- 매번 악성코드의 패턴이 추가될 때마다 악성코드 DB 파일을 수정하는 것은 무리이다.
- 백신 프로그램을 앞으로 실행파일로 변환하는데, 실행파일을 수정하고 배포하는 것은 엄격한 테스트가 필요하다.
- 악성코드를 빨리 퇴치하기 위해서는 될 수 있으면 테스트를 줄여야 하는데 실행파일을 수정하는 것은 좋은 접근법이 아니다.
- 따라서 이번 실습에서는 악성코드 패턴을 실행파일과 분리하여 별도의 파일로 배포한다.

#### 4) 악성코드 진단 및 치료 모듈 분리하기

- 다양한 악성코드를 분석하다 보면 기존의 진단 방식으로 진단되지 않는 악성코드가 존재할 수 있고, 기존의 치료 방식으로 치료되지 않는 악성코드도 존재할 수 있다.
- 이런 경우에는 새로운 진단 방식과 치료 방식을 설계해서 백신 코드에 반영해야 하는데, 백신의 소스코드를 수정한 다음 다시 배포하는 것은 비효율적이다.
- 따라서 이번 실습에서는 악성코드 진단 및 치료 모듈을 백신 코드에서 분리한다.

#### 5) 전용 백신 배포본 만들기

- 지금까지 만들어본 전용 백신을 윈도우 실행 파일로 변환하여 배포본으로 만들어본다.

## 2. 프로그램 설계 / 알고리즘

### 1) python 개발

- 리스트나 해싱 함수, 파일 입출력 등등 전용 백신을 구현하는데 필요한 함수, 라이브러리가 존재하는 python 을 사용하여 전용 백신을 구현한다.

### 2) 리스트 사용

- 바이러스 정보를 구성하기 위한 자료구조로 리스트를 사용한다. 다만, 이 실습은 예전 python 언어를 사용하는 것을 전제로 하고 있기에 바이러스 정보를 리스트에 저장하고 '.'를 기준으로 split 하여 사용한다. 따라서 최신 python을 사용하는 경우 split 할 필요 없이 <key, value>로 저장할 수 있는 딕셔너리를 사용하면 번거로움 없이 구현할 수 있다.

### 3) MD5

- 이번 실습에서는 파일 해싱 함수로 MD5를 사용한다. 이 해싱 함수는 hashlib 라이브러리에 정의되어 있으므로, 이 라이브러리를 import하고 MD5를 사용한다.

### 4) 악성코드 패턴 파일 분리

- 악성코드 패턴 파일을 소스 코드에서 추출하여 별도의 파일로 저장한다.
- 악성코드 패턴 파일에 암호/복호화를 진행하여 외부에서 파일을 조작할 수 없도록 한다.

### 5) 악성코드 진단/치료 모듈 분리

- MD5 진단 모듈을 별도의 파일로 분리하고, 백신의 소스 코드를 수정한다.
- 해당 위치에 악성코드 진단 문자열이 존재하는지 체크 하는 특정 위치 진단 모듈을 구현한다.
- 악성코드 치료 모듈을 별도의 파일로 분리한다.
- 특정 위치 검색법에 해당하는 악성코드 패턴을 악성코드 패턴 파일에 추가한다.

### 6) 전용 백신 배포본 제작

- pyinstaller를 사용하여 제작한 전용 백신을 윈도우 실행파일로 변환한다.
- exit 오류를 해결하기 위해 sys.exit(0)을 명시한다.
- 악성코드 진단 및 치료 모듈을 외부에서 로딩할 수 있도록 imp 라이브러리를 사용하여 모듈을 찾는다.
- 소스를 공개하지 않고 배포하기 위해 pyc 파일을 함께 배포할 수 있도록 한다.

## 3. 소스 코드 / 주석

### 1) 전용 백신 개발하기

#### (1) 리스트 4-1

```
fp = open('eicar.txt', 'rb') #eicar.txt를 이진 파일로 open
fbuf = fp.read()           #eicar.txt 파일의 내용을 저장
fp.close()                 #파일 close
```

#### (2) 리스트 4-2

```
if fbuf[0:3] == b'X5O':    #[0:3]의 내용이 X5O인지 판별
    print("Virus")         #바이러스 임을 출력
else:
    print("No Virus")      #바이러스가 아님을 출력
```

#### (3) 리스트 4-3

```
#!/usr/bin/perl -w
# coding:utf-8 -*-
import os
fp = open('eicar.txt', 'rb') #eicar.txt를 이진 파일로 open
fbuf = fp.read()           #eicar.txt 파일의 내용을 저장
fp.close()                 #파일 close
if fbuf[0:3] == b'X5O':    #[0:3]의 내용이 X5O인지 판별
    print("Virus")         #바이러스 임을 출력
    os.remove('eicar.txt') #eicar.txt 파일 삭제
else:
    print("No Virus")      #바이러스가 아님을 출력
```

#### (4) 리스트 4-4

```
#!/usr/bin/perl -w
# coding:utf-8 -*-
import hashlib
```

```

m = hashlib.md5() #md5 해시함수 불러옴
m.update('hello'.encode('utf-8')) #문자열 hello 해싱
fmd5 = m.hexdigest() #해싱된 문자열 저장

print(fmd5)

```

## (5) 리스트 4-5

```

#-*- coding:utf-8 -*-
import hashlib
import os

fp = open('eicar.txt', 'rb') #eicar.txt를 이진 파일로 open
fbuf = fp.read() #eicar.txt 파일의 내용을 저장
fp.close() #파일 close
m = hashlib.md5() #md5 해시함수 불러옴
m.update(fbuf) #문자열 hello 해싱
fmd5 = m.hexdigest() #해싱된 문자열 저장
if fmd5 == '44d88612fea8a8f36de82e1278abb02f': #해시값 비교
    print("Virus") #같으면 바이러스
    os.remove('eicar.txt') #바이러스 파일 삭제
else:
    print("No Virus") #바이러스 아님

```

## 2) 다양한 악성코드 진단 및 치료하기

### (1) 리스트 5-6

```

#-*- coding:utf-8 -*-
import hashlib
import os
import sys
VirusDB = ['44d88612fea8a8f36de82e1278abb02f:EICAR Test',
           '77bffb0b143e4840ae73d4582a8914a43:Dummy Test'] #악성코드 DB 저장

vdb = []
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':') #'-'를 delimiter로 하여 분리
        t.append(v[0]) #해시 값 저장
        t.append(v[1]) #악성코드 이름 저장
        vdb.append(t) #vdb에 최종적으로 저장
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5: #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
        return False, '' #존재하지 않으면 False와 함께 빈 string 반환
if __name__ == '__main__':
    MakeVirusDB() #vdb 구성
    if len(sys.argv) != 2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    fp = open(fname, 'rb') #파일을 이진 파일로 불러옴
    fbuf = fp.read() #파일 내용 저장
    fp.close() #파일 닫음
    m = hashlib.md5() #해시함수 불러옴
    m.update(fbuf) #해싱 진행
    fmd5 = m.hexdigest() #해싱값 저장
    ret, vname = SearchVDB(fmd5) #vdb에서 악성코드 검색
    if ret == True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
        print("%s : %s" % (fname, vname))
        os.remove(fname) #악성코드 파일 삭제
    else:
        print("%s : ok" % (fname)) #파일 이름 출력하며 안전함을 출력

```

## (2) 리스트 5-9

```

#-*- coding:utf-8 -*-
import hashlib
import os
import sys
VirusDB = ['68:44d88612fea8a8f36de82e1278abb02f:EICAR Test',
           '65:77bffb0b143e4840ae73d4582a8914a43:Dummy Test']
#바이러스 DB에 파일 크기 추가

vdb = []
vsize = []
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':') #'-'를 delimiter로 하여 분리
        t.append(v[1]) #해시 값 저장
        t.append(v[2]) #악성코드 이름 저장
        vdb.append(t) #vdb에 최종적으로 저장
        size = int(v[0]) #파일 크기 저장
        if vsize.count(size) == 0: #vsize에 해당 파일 크기가 없으면 추가
            vsize.append(size):
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5: #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
        return False, '' #존재하지 않으면 False와 함께 빈 string 반환
if __name__ == '__main__':
    MakeVirusDB() #vdb 구성
    if len(sys.argv) != 2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp = open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해시함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest() #해싱값 저장
        ret, vname = SearchVDB(fmd5) #vdb에서 악성코드 검색
        if ret == True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
            print("%s : %s" % (fname, vname))
            os.remove(fname) #악성코드 파일 삭제
        else:
            print("%s : ok" % (fname)) #파일 이름 출력하며 안전함을 출력

```

## 3) 악성코드 패턴 분리하기

### (1) 리스트 6-4

```

#-*- coding:utf-8 -*-
import hashlib
import os
import sys
VirusDB = []
vdb = []
vsize = []
def LoadVirusDB():
    fp =open('virus.db', 'rb') #virus.db를 이진 파일로 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #일을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split('.') # '.'를 delimiter로 하여 분리
        t.append(v[1]) #해시 값 저장
        t.append(v[2]) #악성코드 이름 저장
        vdb.append(t) #vdb에 최종적으로 저장
        size =int(v[0]) #파일 크기 저장
        if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
            vsize.append(size)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5: #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
            #존재하지 않으면 False와 함께 빈 string 반환
    return False, ''
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print("Usage : antivir.py [file]")
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해시함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest() #해싱값 저장
        ret, vname = SearchVDB(fmd5) #vdb에서 악성코드 검색
        if ret ==True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
            print("%s : %s" % (fname, vname))
            os.remove(fname) #악성코드 파일 삭제
        else:
            print("%s : ok" % (fname)) #파일 이름 출력하며 안전함을 출력

```

## (2) 리스트 6-5

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib
import os
def main():
    if len(sys.argv) !=2: #파일 입력이 없는 경우 메시지 출력 및 함수 종료
        print("Usage : kmake.py [file]")
        return

    fname = sys.argv[1] #파일 이름 저장
    tname = fname
    fp =open(tname, 'rb') #파일을 이진 파일로 불러옴
    buf = fp.read() #파일 내용 저장
    fp.close() #파일 닫음
    buf2 = zlib.compress(buf) #파일 내용 압축
    buf3 = ""
    for c in buf2:
        buf3 += chr(ord(c) ^ 0xFF) #압축된 내용을 0xFF로 xor
    buf4 = 'KAVM'+ buf3 #헤더 생성
    f = buf4
    for i in range(3):
        md5 = hashlib.md5() #해시함수 불러옴
        md5.update(f) #해시값 구함
        f = md5.hexdigest() #해시값 저장
    buf4 += f #암호화된 내용에 해시값 저장
    kmd_name = fname.split('.')[0] + '.kmd' #파일 이름에 .kmd 확장자 추가
    fp =open(kmd_name, 'wb') #kmd 확장자로 암호 파일 생성
    fp.write(buf4) #파일 출력
    fp.close() #파일 닫음
    print("%s -> %s" % (fname, kmd_name)) #파일 이름과 암호화된 파일 이름 출력
if __name__ == '__main__':
    main()

```

## (3) 리스트 6-6

```

def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해시함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #불리함 MD5 값과 같은지 확인
            raise SystemError

        buf3 = ""
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환

```

## (4) 리스트 6-7

```

#-*- coding:utf-8 -*-
import sys
import zlib

```

```

import hashlib
import os
import StringIO
VirusDB = []
vdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #읽을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split('.') #'. '를 delimiter로 하여 분리
        t.append(v[1]) #해시 값 저장
        t.append(v[2]) #악성코드 이름 저장
        vdb.append(t) #vdb에 최종적으로 저장
        size =int(v[0]) #파일 크기 저장
        if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
            vsize.append(size)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5: #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
    return False, '' #존재하지 않으면 False와 함께 빈 string 반환
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해싱함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest() #해시값 저장
        ret, vname = SearchVDB(fmd5) #vdb에서 악성코드 검색
        if ret ==True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
            print("%s : %s" % (fname, vname))
            os.remove(fname) #악성코드 파일 삭제
        else:
            print("%s : ok" % (fname)) #파일 이름 출력하며 안전함을 출력

```

## 4) 악성코드 진단 및 치료 모듈 분리하기

### (1) 리스트 7-2

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib
import os
import StringIO
VirusDB = []
vdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #읽을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []

```

```

v = pattern.split(',') #','를 delimiter로 하여 분리
t.append(v[1])          #해시 값 저장
t.append(v[2])          #악성코드 이름 저장
vdb.append(t)           #vdb에 최종적으로 저장
size =int(v[0])          #파일 크기 저장
if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
    vsize.append(size):
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5:
            #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
        return False, ' '    #존재하지 않으면 False와 함께 빈 string 반환
def ScanMD5(fname):
    ret =False
    vname =''
    size = os.path.getsize(fname) #파일의 사이즈 저장
    if vsize.count(size):         #파일의 사이즈가 vsize에 존재한다면
        fp =open(fname, 'rb') #파일 불러옴
        fbuf = fp.read() #파일의 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해싱함수 불러옴
        m.update(fbuf) #해시값 계산
        fmd5 = m.hexdigest() #해시값 저장
        ret, vname = SearchVDB(fmd5) #해시값 검색
    return ret, vname
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    ret, vname = ScanMD5(fname) #MD5 해시로 바이러스 스캔
    if ret ==True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
        print('%s : %s' % (fname, vname))
        os.remove(fname) #악성코드 파일 삭제
    else:
        print('%s : ok' % (fname)) #파일 이름 출력하며 안전함을 출력

```

## (2) 리스트 7-4

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib
import os
import StringIO
import scanmod
VirusDB = []
vdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #있을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(',') #','를 delimiter로 하여 분리
        t.append(v[1])          #해시 값 저장
        t.append(v[2])          #악성코드 이름 저장
        vdb.append(t)           #vdb에 최종적으로 저장
        size =int(v[0])          #파일 크기 저장
        if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
            vsize.append(size):
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size):         #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해싱함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest() #해싱값 저장
        ret, vname = scanmod.ScanMD5(vdb, vsize, fname) #scanmod의 SearchVDB 함수 실행
        if ret ==True: #악성코드 존재하면 파일 이름과 악성코드 이름 출력
            print('%s : %s' % (fname, vname))
            os.remove(fname) #악성코드 파일 삭제
        else:
            print('%s : ok' % (fname)) #파일 이름 출력하며 안전함을 출력

```

## (3) 리스트 7-9

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib

```

```

import os
from io import StringIO
import scanmod #
import curemod #
VirusDB = []
vdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #읽을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(',') #'.'를 delimiter로 하여 분리
        t.append(v[1]) #해시 값 저장
        t.append(v[2]) #악성코드 이름 저장
        vdb.append(t) #vdb에 최종적으로 저장
        size =int(v[0]) #파일 크기 저장
        if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
            vsize.append(size)
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해싱함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest()#해시값 저장
        ret, vname = scanmod.SearchVDB(vdb, vsize, fmd5) #scanmod의 SearchVDB 함수 실행
        if ret ==True:
            print("%s : %s" % (fname, vname))
            curemod.CureDelete(fname)
        else:
            print("%s : ok" % (fname))

```

## (4) 리스트 7-15

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib
import os
import StringIO
import scanmod
import curemod
VirusDB = []
vdb = []
sdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #읽을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(',') #'.'를 delimiter로 하여 분리
        scan_func = v[0] #악성코드 검사 함수
        #cure_func = v[1] #악성코드 치료 함수

```



```

if scan_func == "ScanMD5" : #검사 함수가 MD5라면
    t.append(v[3]) #해시값 저장
    t.append(v[4]) #악성코드 이름값 저장
    vdb.append(t) #vdb에 최종적으로 저장
    size =int(v[2]) #파일 크기 저장
    if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
        vsize.append(size):
    elif scan_func == "ScanStr": #검사 함수가 특정 위치 검색법이라면
        t.append(int(v[2])) #문자열의 위치 저장
        t.append(v[3]) #진단 문자열 저장
        t.append(v[4]) #악성코드 이름 저장
        sdb.append(t) #sdb에 최종적으로 저장
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivir.py [file]')
        exit(0)
    fname = sys.argv[1] #파일 이름 저장
    ret, vname = scanmod.ScanVirus(vdb, vsize, sdb, fname) #scanmod의 SearchVDB 함수 실행
    if ret ==True:
        print("%s : %s" % (fname, vname)) #파일의 이름과 악성코드 이름 출력
        curemod.CureDelete(fname) #curemod의 CureDelete 함수 실행
    else:
        print("%s : ok" % (fname)) #파일이 안전함을 출력

```

## (5) scanmod.py

```

#-*- coding:utf-8 -*-
import os
import hashlib
def SearchVDB(vdb, fmd5):
    for t in vdb:
        if t[0] == fmd5: #vdb에서 해시를 검색해서 존재여부 확인
            return True, t[1] #존재하면 True와 함께 악성코드 이름 반환
        #존재하지 않으면 False와 함께 빈 string 반환
    return False, ''
def ScanMD5(vdb, vsize, fname):
    ret =False
    vname =''
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해시함수 불러옴
        m.update(fbuf) #해시 진행
        fmd5 = m.hexdigest() #해시값 저장
        ret, vname = SearchVDB(vdb, fmd5) #vdb에서 악성코드 검색
    return ret, vname #검사 결과와 악성코드 이름 반환
def ScanStr(fp, offset, mal_str):
    size =len(mal_str) #악성코드 진단 문자열의 길이 저장
    fp.seek(offset) #문자열이 존재할 것으로 예상되는 위치로 이동
    buf = fp.read(size) #길이만큼 읽기
    if buf == mal_str:
        return True #악성코드 발견
    else:
        return False #악성코드 미발견
def ScanVirus(vdb, vsize, sdb, fname):
    ret, vname = ScanMD5(vdb, vsize, fname) #vdb에서 MD5해싱을 이용하여 악성코드 존재하는지 확인
    if ret ==True : #존재하면 존재 여부와 악성코드 이름 반환
        return ret, vname

    fp =open(fname, 'rb') #파일을 이진파일로 불러옴
    for t in sdb :
        if ScanStr(fp, t[0], t[1]) ==True: #특정 위치 검색법을 이용해서 찾은 경우
            ret =True #True 저장
            vname = t[2] #악성코드 이름 저장
            break #for문 탈출
    fp.close() #파일 닫음
    return ret, vname #악성코드 존재 여부와 이름 반환
...
def ScanVirus(vdb, vsize, sdb, fname):
    print("[*] New ScanVirus") #새로운 스캔 함수임을 출력
    ret, vname = ScanMD5(vdb, vsize, fname) #vdb에서 MD5해싱을 이용하여 악성코드 존재하는지 확인
    if ret == True : #존재하면 존재 여부와 악성코드 이름 반환
        return ret, vname

    fp = open(fname, 'rb') #파일을 이진파일로 불러옴
    for t in sdb :
        if ScanStr(fp, t[0], t[1]) == True: #특정 위치 검색법을 이용해서 찾은 경우
            ret = True #True 저장
            vname = t[2] #악성코드 이름 저장
            break #for문 탈출
    fp.close() #파일 닫음
    return ret, vname #악성코드 존재 여부와 이름 반환
...

```

## (6) scan\_str.py

```

#-*- coding:utf-8 -*-
def ScanStr(fp, offset, mal_str):
    size =len(mal_str) #악성코드 진단 문자열의 길이 저장
    fp.seek(offset) #문자열이 존재할 것으로 예상되는 위치로 이동
    buf = fp.read(size) #길이만큼 읽기
    if buf == mal_str:
        return True #악성코드 발견
    else:
        return False #악성코드 미발견

fp =open('eicar.txt', 'rb') #파일을 이진파일로 불러옴
print(ScanStr(fp, 0, b'X5O')) #0번째 위치에 X5O가 존재하는지 확인
fp.close() #파일 닫음

```

### (7) curemod.py

```

#-*- coding:utf-8 -*-
import os
def CureDelete(fname):
    return os.remove(fname) #인자로 받은 파일 이름에 해당하는 파일 삭제

```

## 5) 전용 백신 배포본 만들기

### (1) 리스트 8-1

```

#-*- coding:utf-8 -*-

```

```

import sys
import zlib
import hashlib
import os
import StringIO
import scanmod #
import curemod #
VirusDB = []
vdb = []
sdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해시함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화
    fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
    while True:
        line = fp.readline() #라인 별로 저장
        if not line : break #읽을 라인 없을 경우 while 탈출
        line = line.strip() #선행과 후행 문자 제거
        VirusDB.append(line) #VirusDB에 저장
    fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(',') #'-'를 delimiter로 하여 분리
        scan_func = v[0] #악성코드 검사 함수
        cure_func = v[1] #악성코드 치료 함수
        if scan_func == "ScanMD5" : #검사 함수가 MD5라면
            t.append(v[3]) #해시값 저장
            t.append(v[4]) #악성코드 이름값 저장
            vdb.append(t) #vdb에 최종적으로 저장
            size =int(v[2]) #파일 크기 저장
            if vsize.count(size) ==0: #vsize에 해당 파일 크기가 없으면 추가
                vsize.append(size);
        elif scan_func == "ScanStr": #검사 함수가 특정 위치 검색법이라면
            t.append(int(v[2])) #문자열의 위치 저장
            t.append(v[3]) #진단 문자열 저장
            t.append(v[4]) #악성코드 이름 저장
            sdb.append(t) #sdb에 최종적으로 저장
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) !=2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        sys.exit(0) #실행파일로 변환했을때 오류 방지하기 위해 sys 명시
    fname = sys.argv[1] #파일 이름 저장
    size = os.path.getsize(fname) #입력받은 파일의 크기 저장
    if vsize.count(size): #vsize에 입력 받은 파일의 크기가 저장되어 있다면
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        fbuf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        m = hashlib.md5() #해시함수 불러옴
        m.update(fbuf) #해싱 진행
        fmd5 = m.hexdigest()#해시값 저장
        ret, vname = scanmod.SearchVDB(vdb, vsize, fmd5) #scanmod의 SearchVDB 함수 실행
        if ret ==True:
            print("%s : %s" % (fname, vname)) #파일의 이름과 악성코드 이름 출력
            curemod.CureDelete(fname) #curemod의 CureDelete 함수 실행
        else:
            print("%s : ok" % (fname)) #파일이 안전함을 출력

```

## (2) 리스트 8-2

```

#-*- coding:utf-8 -*-
import sys
import zlib
import hashlib
import os
import StringIO
import scanmod #
import curemod #
import imp
VirusDB = []
vdb = []
sdb = []
vsize = []
def DecodeKMD(fname):
    try :
        fp =open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음
        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리
        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해시함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장
        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 =''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass
    return None #오류가 있으면 None 반환
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd') #virus.kmd를 복호화

```

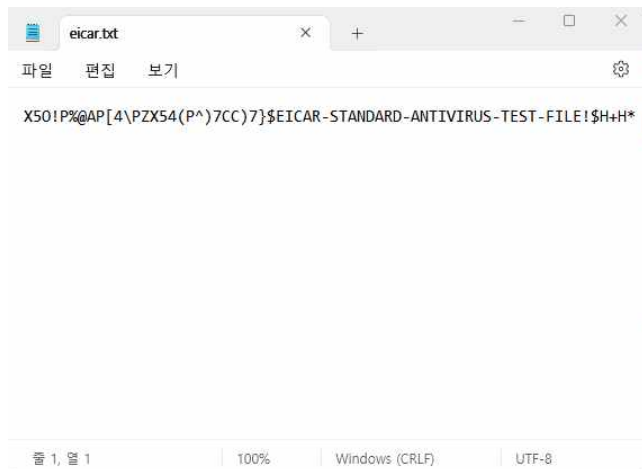
```

fp = StringIO.StringIO(buf) #복호화한 파일을 불러옴
while True:
    line = fp.readline() #라인 별로 저장
    if not line : break #일을 라인 없을 경우 while 탈출
    line = line.strip() #선행과 후행 문자 제거
    VirusDB.append(line) #VirusDB에 저장
fp.close() #파일 종료
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':') #'-'를 delimiter로 하여 분리
        scan_func = v[0] #악성코드 검사 함수
        #cure_func = v[1] #악성코드 치료 함수
        if scan_func == "ScanMD5": #검사 함수가 MD5라면
            t.append(v[3]) #해시값 저장
            t.append(v[4]) #악성코드 이름값 저장
            vdb.append(t) #vdb에 최종적으로 저장
            size = int(v[2]) #파일 크기 저장
            if vsize.count(size) == 0: #vsize에 해당 파일 크기가 없으면 추가
                vsize.append(size):
        elif scan_func == "ScanStr": #검사 함수가 특정 위치 검색법이라면
            t.append(int(v[2])) #문자열의 위치 저장
            t.append(v[3]) #진단 문자열 저장
            t.append(v[4]) #악성코드 이름 저장
            sdb.append(t) #sdb에 최종적으로 저장
if __name__ == '__main__':
    LoadVirusDB() #바이러스 DB 불러옴
    MakeVirusDB() #vdb 구성
    if len(sys.argv) != 2 : #파일을 입력하지 않았다면 메시지 출력 및 종료
        print('Usage : antivirus.py [file]')
        sys.exit(0) #실행파일로 변환했을때 오류 방지하기 위해 sys 명시
    fname = sys.argv[1] #파일 이름 저장
    try :
        m = 'scanmod'
        f, filename, desc = imp.find_module(m, []) #현재 폴더에서 모듈 탐색
        module = imp.load_module(m, f, filename, desc) #모듈 로드
        cmd = 'ret, vname = module.ScanVirus(vdb, vsize, sdb, fname)' #진단 함수 호출 명령어 구성작업
        exec(cmd) #명령어 실행
    except ImportError:
        ret, vname = scanmod.ScanVirus(vdb, vsize, sdb, fname) #동적 import에 실패하면 실행
    if ret == True:
        print("%s : %s" % (fname, vname)) #파일의 이름과 악성코드 이름 출력
        curemod.CureDelete(fname) #curemod의 CureDelete 함수 실행
    else:
        print("%s : ok" % (fname)) #파일이 안전함을 출력

```

## 4. 결과 / 결과 분석

### 1) 전용 백신 개발하기



- eicar.txt에 바이러스 서명을 입력한 모습이다.
- 입력하고 저장하면 안티바이러스 프로그램이 바로 인식하기 때문에 주의해야 한다.

(1) [리스트 4-1] ~ [리스트4-4]의 내용을 익히고 테스트해본다.

```

fp = open('eicar.txt', 'rb') #eicar.txt를 이진 파일로 open
fbuf = fp.read() #eicar.txt 파일의 내용을 저장
fp.close() #파일 close

if fbuf[0:3] == b'X50':
    print("Virus") #[0:3]의 내용이 x50인지 판별
    #바이러스 임을 출력
else:
    print("No Virus") #바이러스가 아님을 출력

```

```

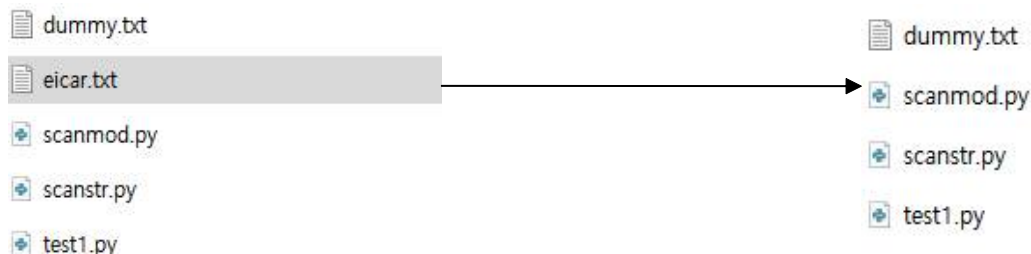
/python.exe "c:/Users/kocan/OneDrive - dongguk.edu/컴공/컴퓨터보안/과제/6주차/test1.py"
Virus

```

- 리스트 4-1, 4-2를 합쳐서 테스트한 모습이다.
- eicar.txt를 이진 파일로 불러와 앞부분이 X50 이라면 바이러스로 판단한다.

```
/python.exe "c:/Users/kocan/OneDrive - dongguk.edu/컴공/컴퓨터보안/과제/6주차/4-3.py"
Virus
```

- 리스트 4-3을 테스트한 모습이다.
- eicar.txt를 이진 파일로 불러와 앞부분이 X50 이라면 바이러스로 판단하고, 그 파일을 삭제한다.



- 실제로 eicar.txt 파일이 삭제된 모습을 확인할 수 있다.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> & C:/Python27/python.exe "c:/Users/kocan/OneDrive - dongguk.edu/컴공/컴퓨터보안/과제/6주차/4-4.py"
5d41402abc4b2a76b9719d911017c592
```

- 리스트 4-4를 테스트한 모습이다.
- 'hello'라는 문자열을 MD5로 해싱하여 해시값을 출력한 모습이다.

(2) 악성코드 진단 문자열과 md5해시를 이용하여 EICAR Test파일(악성코드)을 진단하는 전용백신을 테스트해보아라 (리스트4-5 수행).

```
/python.exe "c:/Users/kocan/OneDrive - dongguk.edu/컴공/컴퓨터보안/과제/6주차/4-5.py"
Virus
```

- 리스트 4-5를 테스트한 모습이다.
- eicar.txt 파일을 불러와 MD5 해시함수를 이용하여 해싱을 진행하고, 해시값이 바이러스로 미리 정의된 해시값과 같은 경우에는 바이러스로 판단한다.

## 2) 다양한 악성코드 진단 및 치료하기

(1) [리스트 5-6]을 실행하여 악성코드를 진단해보자.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "5-6.py" eicar.txt
eicar.txt : EICAR Test

PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "5-6.py" dummy.txt
dummy.txt : Dummy Test
```

- eicar.txt와 dummy.txt 파일을 이용해 악성코드를 진단한 모습이다. 둘 다 악성코드로 분류되었다.
- 이미 저장된 악성코드 DB에서 해시값을 가져와 파일의 해시값과 비교해서 같으면 악성코드로 분류하고, 악성코드의 이름과 파일의 이름을 출력한다.

(2) 백신의 검사속도가 느린 이유가 무엇인지 파악하여 정리하여 보자.

- 악성코드의 존재 여부와 상관없이 파일을 열고 해싱을 진행한 다음, 비교 과정을 수행한다. 현재 연 파일이 악성코드 파일이면 상관이 없으나, 만약 악성코드에 감염된 파일이 아니라면 해싱을 진행하고, 비교를 수행하는 행위는 필요하지 않게 된다. 그러나 리스트 5-6은 악성코드 파일인지 아닌지 구분하지 못하기 때문에 검사가 필요 없는 파일임에도 검사를 수행하고, 이는 속도 감소로 이어진다.

(3) [리스트 5-9]을 실행하여 악성코드를 진단해보자.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python 5-9.py eicar.txt
eicar.txt : EICAR Test
```

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "5-9.py" dummy.txt
dummy.txt : Dummy Test
```

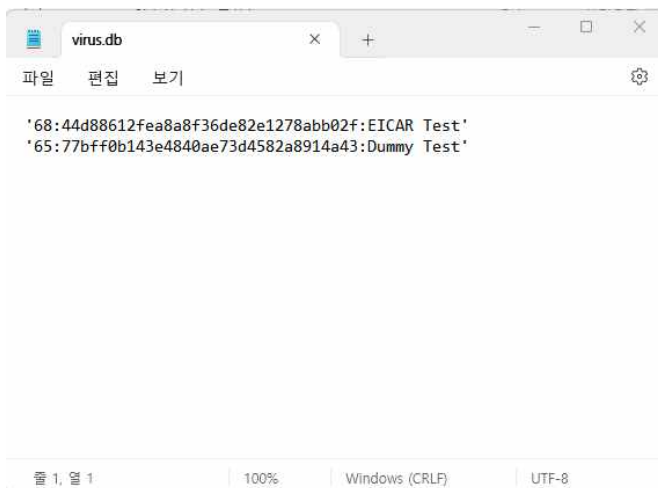
- 리스트 5-6에서 악성코드 DB에 악성코드 파일 크기 속성을 추가했다.
- 입력 파일의 크기와 악성코드로 판별된 파일의 크기를 비교해서 같으면 그때야 파일을 입력받는다.
- eicar.txt와 dummy.txt 파일을 대상으로 테스트했을 때 둘 다 악성코드로 판별되었다.

(4) [리스트 5-6] 와 [리스트 5-9]의 코드를 각각 실행 하였을 때 속도 차이를 확인할 수 있는가? Yes or No?  
그 이유는 무엇인가?

- 속도 차이를 확인할 수 있었다.
- 그 이유는 입력받은 파일이 악성코드 파일이면 두 소스 코드 모두 속도 차이가 거의 없을 것이다. 그러나 악성코드 파일이 아닌 경우에는 리스트 5-6의 경우에는 파일을 열어서 해싱을 진행한 다음, DB에 저장된 값과 비교하지만, 리스트 5-9는 파일을 열기 전에 파일 크기를 비교하여 DB에 저장된 악성코드 파일 크기와 다르다면 파일을 열지 않고, 해싱을 진행하지 않으며, 비교도 하지 않는다. 따라서 리스트 5-9가 리스트 5-6보다 속도가 빠르다고 할 수 있다.

### 3) 악성코드 패턴 분리하기

(1) [리스트 6-4]에 제시된 별도의 파일로 분리된 악성코드 패턴을 로딩하여 악성코드를 진단 및 치료하는 antivirus.py 코드를 실행시켜보자.



- 악성코드 DB를 별도의 파일로 추출한 모습이다.
- EICAR TEST와 Dummy Test 악성코드의 파일 크기와 해시값과 이름이 저장되어 있다.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "6-4.py" eicar.txt
eicar.txt : EICAR Test
```

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "6-4.py" dummy.txt
dummy.txt : Dummy Test
```

- 리스트 6-4를 테스트한 모습이다.
- 외부의 악성코드 DB 파일을 불러와 악성코드 DB를 내부에서 구성하고, 입력받은 파일을 MD5로 해싱한 다음 값을 비교하여 악성코드를 판별한다.
- eicar.txt와 dummy.txt 파일 둘 다 악성코드로 판별된 것을 볼 수 있다.

(2) 악성코드 패턴 파일(virus.db)을 암호화하는 도구는 백신업체만 가지고 있어야 한다. 따라서 별도의 도구로

만들어야 한다. [리스트 6-5]는 악성코드 패턴 파일을 암호화하는 파이썬 코드(kmake.py)이다. 이를 실행하여 virus.db를 암호화한 virus.kmd를 생성하여 보자.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "6-5.py" virus.db
virus.db -> virus.kmd
```

- 악성코드 DB 파일이 암호화된 모습을 볼 수 있다.
- 파일 내용을 zlib 라이브러리를 이용해 압축한 다음 0xFF로 xor 연산한다.
- 파일의 앞에 'KAVM' 헤더를 붙인다
- MD5 해싱 함수를 3번 적용한 다음에 .kmd 확장자로 파일이 저장되었다.

(3) antivirus.py에서는 암호화된 virus.kmd를 바로 로딩 할 수 없다. 따라서 이를 복호화한 다음 로딩 할 수 있도록 복호화 모듈이 필요하다. [리스트 6-6]의 악성코드 패턴 파일을 복호화하는 파이썬 코드(DecodeKMD 함수)를 분석해보자.

```
def DecodeKMD(fname):
    try :
        fp = open(fname, 'rb') #파일을 이진 파일로 불러옴
        buf = fp.read() #파일 내용 저장
        fp.close() #파일 닫음

        buf2 = buf[:-32] #암호화 내용 분리
        fmd5 = buf[-32:] #MD5 분리

        f = buf2
        for i in range(3):
            md5 = hashlib.md5() #해싱함수 불러옴
            md5.update(f) #해시값 구함
            f = md5.hexdigest() #해시값 저장

        if f != fmd5: #분리한 MD5 값과 같은지 확인
            raise SystemError

        buf3 = ''
        for c in buf2[4:]:
            buf3 += chr(ord(c) ^ 0xFF) #0xFF로 xor 한다
        buf4 = zlib.decompress(buf3) #압축을 해제한다
        return buf4 #복호화된 내용 반환
    except :
        pass

    return None #오류가 있으면 None 반환
```

- 암호화된 악성코드 DB 파일을 불러와 암호화 내용과 MD5 해시값을 분리한다.
- 암호화 내용에 MD5 해싱 함수를 3번 적용한 다음에, 이 해시값이 이전에 분리한 MD5 해시값과 같은지 확인한다.
- 암호화 내용에 0xFF를 xor 연산하고, zlib 라이브러리를 사용해 압축을 해제한다.

(4) 이제 antivirus.py에 DecodeKMD 함수를 적용하여 virus.kmd로부터 악성코드 패턴을 로딩하여 보자 (이때 StringIO 모듈의 fp.readline 함수가 사용된다.). 즉, [리스트 6-7]을 실행시켜보자.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "6-7.py" eicar.txt
eicar.txt : EICAR Test
```

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "6-7.py" dummy.txt
dummy.txt : Dummy Test
```

- 리스트 6-7를 테스트한 모습이다.
- 외부에서 암호화된 악성코드 DB 파일을 불러와 복호화를 진행한 다음, 내부에서 악성코드 DB를 생성한다.



- DB에 저장된 파일 크기와 입력 파일의 크기를 비교하여 같으면 파일을 연다.
- 파일에 MD5 해싱을 진행한 다음, DB에 파일의 해시값이 존재하는지 확인하고, 악성코드 여부를 판별한다.
- eicar.txt와 dummy.txt 모두 악성코드로 판별된 모습을 볼 수 있다.

#### 4) 악성코드 진단 및 치료 모듈 분리하기

(1) SearchVDB와 ScanMD5를 추출하고 scanmod로 모듈화한 백신 프로그램 (antivirus.py) 인 [리스트 7-4]를 실행하여 보아라. 결과는 [그림 7-1] 과 같아야 한다.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "7-4.py" eicar.txt
eicar.txt : EICAR Test

PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "7-4.py" dummy.txt
dummy.txt : Dummy Test
```

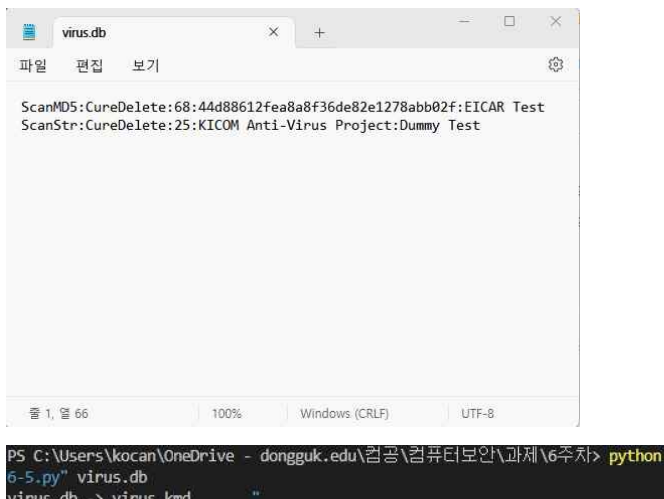
- SearchVDB와 ScanMD5가 scanmod.py로 추출되어 모듈화되었다.
- 이 함수를 이용하기 위해 scanmod를 import 했다.
- 함수가 모듈화되어 분리했기 때문에, 메인 프로그램의 변수에 직접 접근할 수 없게 되어 파라미터를 통해 값을 가져와 계산하게 된다. 따라서 파라미터의 개수가 늘었다.
- 함수를 사용하려면 scanmod.함수 형태로 사용하게 된다.
- 악성코드 탐지의 경우에는 위 방식과는 크게 달라지지 않았으며, 외부에서 저장된 DB 파일을 불러와 복호화를 진행하고 DB를 재구성한 다음, 입력 파일을 해싱하여 DB에 저장된 파일 크기와 같은지 확인하고 악성코드 여부를 판별한다.
- eicar.txt와 dummy.txt 모두 악성코드로 판별된 모습을 볼 수 있다.

(2) [리스트 7-5]의 특정위치 검색법(scan\_str.py)을 사용하여 악성코드를 검사하여 보아라. 결과는 [그림 7-2]와 같아야 한다.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> & C:/Users/kocan/anaconda3/python.exe "c:/Users/kocan/OneDrive - dongguk.edu/컴공/컴퓨터보안/과제/6주차/scanstr.py"
True
```

- 파일의 문자열과 오프셋, 악성코드 파일 문자열의 일부분을 파라미터로 받는다.
- 파일의 문자열에서 오프셋만큼 위치를 이동한 다음, 악성코드 문자열과 비교해서 같으면 악성코드로 판별한다.
- eicar.txt 파일에서 0번째 위치에 'X50'가 존재하는지 ScanStr 함수를 이용해 판별하였더니, 악성코드로 분류되었다.

(3) MD5해쉬와 특정위치 검색법을 이용할 수 있도록 악성코드 패턴 파일(virus.db)을 수정하고 이를 주제3에서 다루었던 kmake.py를 이용하여 암호화한다. 결과는 [그림 7-3]과 같아야 한다.



The image shows a text editor window titled 'virus.db' with the following content:

```
ScanMD5:CureDelete:68:44d88612fea8a8f36de82e1278abb02f:EICAR Test
ScanStr:CureDelete:25:KICOM Anti-Virus Project:Dummy Test
```

Below the editor, a terminal window shows the command and output:

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python 6-5.py" virus.db
virus.db -> virus.kmd
```

- 악성코드별로 MD5와 특정 위치 검색법을 적용할 수 있도록 DB 파일을 수정한 모습이다.
- MD5일 때 파일의 크기를 저장하지만, 특정 위치 검색법의 경우에는 파일에서 문자열을 비교할 위치를 지정하는 오프셋을 저장하게 된다.
- 수정된 DB 파일은 암호화 프로그램을 이용하여 virus.kmd 파일로 암호화하게 된다.

(4) 7.4절을 포함하여 7장에서 언급된 모든 내용을 백신에 반영하여 수정하고 악성코드를 진단하여 보아라. 결과는 [그림7-4]와 같아야 한다. Dummy Test 악성코드는 MD5 해시를 이용해서 검사했고 EICAR Test 악성코드는 특정 위치 검색법을 이용하여 검사했는지 확인하여라.

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python 7-15.py eicar.txt
eicar.txt : EICAR Test
```

```
PS C:\Users\kocan\OneDrive - dongguk.edu\컴공\컴퓨터보안\과제\6주차> python "7-15.py" dummy.txt
dummy.txt : Dummy Test
```

- eicar.txt는 MD5 해시를 이용하여 검사했다. DB에는 MD5 해시를 이용하여 검사하게 되어있기 때문에, 파일의 크기와 해시값, 악성코드 이름을 가져와서 eicar.txt의 해시값과 비교해서 같으면 악성코드임을 판별한다. 이 경우에는 해시값이 같기에 파일의 이름과 악성코드 이름이 같이 출력하게 된다.
- dummy.txt는 특정 위치 검색법을 이용하여 검사했다. DB에 특정 위치 검색법을 이용하여 검사하게 되어있기 때문에, 오프셋과 악성코드 문자열, 악성코드 이름을 가져와서 dummy.txt에서 오프셋만큼 이동한 위치에 악성코드 문자열이 존재하는 경우 악성코드임을 판별하게 된다. 이 경우에는 문자열이 존재하기에 파일의 이름과 악성코드 이름이 같이 출력하게 된다.

## 5) 전용 백신 배포본 만들기

(1) PyInstaller를 사용하여 주제4에서 작성한 파이썬으로 된 백신을 윈도우 실행파일로 변환한다. [그림8-2] 및 [그림8-3]과 같은 결과를 보이는지 확인한다.

```
PS C:\Users\kocan\Desktop\asdf> pyinstaller -F 7-15.py
84 INFO: PyInstaller: 3.2
84 INFO: Python: 2.7.18
85 INFO: Platform: Windows-10-10.0.22621
86 INFO: wrote C:\Users\kocan\Desktop\asdf\7-15.spec
89 INFO: UPX is not available.
91 INFO: Extending PYTHONPATH with paths
['C:\\Users\\kocan\\Desktop\\asdf', 'C:\\Users\\kocan\\Desktop\\asdf']
91 INFO: checking Analysis
```

- pyinstaller로 실행파일을 만드는 모습이다.
- pyinstaller와 python의 버전, 운영 체제 정보가 출력되며 그 후에는 파일 생성 과정을 보여준다.
- 원래는 python 폴더에 pyinstaller를 압축 풀어서 하려고 했으나, 실행이 제대로 되지 않아 pip install pyinstaller==3.2로 pyinstaller를 다운 받아 사용했다.
- -F 옵션은 모듈을 하나로 합쳐 하나의 실행파일로 만드는 옵션이다. --onefile 옵션과 동일한 옵션이다.

```
PS C:\Users\kocan\Desktop\asdf> cd dist
PS C:\Users\kocan\Desktop\asdf\dist> dir

디렉터리: C:\Users\kocan\Desktop\asdf\dist

Mode                LastWriteTime         Length Name
----                -
-a----          2023-04-21 오전 12:40        3362271 7-15.exe
-a----          2023-04-19 오전 10:40           65 dummy.txt
-a----          2023-04-20 오후 3:44           68 eicar.txt
```

- dist 디렉터리로 이동하여 실행파일이 생성되었는지 확인하였다.
- dir 명령어를 통해 디렉터리 내 파일 리스트를 출력하였으며, Mode와 최종 수정 날짜, 파일 길이와 이름이 같이 출력된다.
- 실행파일이 정상적으로 생성된 모습을 볼 수 있다.



(2) 변환된 백신의 윈도우 실행파일을 테스트해본다. exit오류가 발생하면 8.3.1절의 내용에 따라 수정한다. [그림8-6]과 같은지 확인한다.

```
PS C:\Users\kocan\Desktop\asdf\dist> ./7-15.exe
Usage : antivirus.py [file]
Traceback (most recent call last):
  File "7-15.py", line 84, in <module>
    NameError: name 'exit' is not defined
Failed to execute script 7-15
```

- 생성한 백신 파일을 실행한 모습이다.
- 실행파일 내 exit 함수가 정의되지 않았다는 오류 메시지를 출력하며 종료되었다.

```
PS C:\Users\kocan\Desktop\asdf> pyinstaller -F 8-1.py
90 INFO: PyInstaller: 3.2
90 INFO: Python: 2.7.18
92 INFO: Platform: Windows-10-10.0.22621
92 INFO: wrote C:\Users\kocan\Desktop\asdf\8-1.spec
96 INFO: UPX is not available.
97 INFO: Extending PYTHONPATH with paths
['C:\\Users\\kocan\\Desktop\\asdf', 'C:\\Users\\kocan\\Desktop\\asdf']
97 INFO: checking Analysis
125 INFO: checking PYZ
142 INFO: checking PKG
```

- exit를 sys.exit로 수정한 파일을 실행파일로 생성하고 있다.
- pyinstaller과 python의 버전, 운영 체제 정보가 출력되며 그 후에는 파일 생성 과정을 보여준다.
- 원래는 python 폴더에 pyinstaller를 압축 풀어서 하려고 했으나, 실행이 제대로 되지 않아 pip install pyinstaller==3.2로 pyinstaller를 다운 받아 사용했다.
- -F 옵션은 모듈을 하나로 합쳐 하나의 실행파일로 만드는 옵션이다. --onefile 옵션과 동일한 옵션이다.

```
PS C:\Users\kocan\Desktop\asdf> cd dist
PS C:\Users\kocan\Desktop\asdf\dist> dir

디렉터리: C:\Users\kocan\Desktop\asdf\dist

Mode                LastWriteTime         Length Name
----                -
-a----            2023-04-21 오전 12:40       3362271 7-15.exe
-a----            2023-04-21 오전 12:41       3362397 8-1.exe
-a----            2023-04-19 오전 10:40           65 dummy.txt
-a----            2023-04-20 오후 3:44           68 eicar.txt
```

- dist 디렉터리로 이동하여 실행파일이 생성되었는지 확인하였다.
- dir 명령어를 통해 디렉터리 내 파일 리스트를 출력하였으며, Mode와 최종 수정 날짜, 파일 길이와 이름이 같이 출력된다.
- 실행파일이 정상적으로 생성된 모습을 볼 수 있다.

```
PS C:\Users\kocan\Desktop\asdf\dist> ./8-1.exe
Usage : antivirus.py [file]
```

- 수정한 백신 파일을 실행한 모습이다.
- exit 함수가 sys.exit 함수로 정의되었기 때문에, 오류 메시지의 출력 없이 종료된 모습을 볼 수 있다.

(3) [그림8-9]와 [그림8-11]에 보여지는 것과 같이 악성코드 진단 및 치료 모듈을 antivirus.exe의 외부에서 로딩 할 수 있도록 scanmod.py의 ScanVirus함수를 수정해본다. scanmod.py가 잘 반영되는지 백신을 테스트한다. 또한 소스를 공개하지 않고 배포하기 위하여 scanmod.py를 컴파일한 scanmod.pyc를 함께 배포하

는 방법도 테스트해 본다.

```
PS C:\Users\kocan\Desktop\asdf\dist> dir
```

디렉터리: C:\Users\kocan\Desktop\asdf\dist

Mode	LastWriteTime	Length	Name
-a----	2023-04-21 오전 12:40	3362271	7-15.exe
-a----	2023-04-21 오전 12:41	3362397	8-1.exe
-a----	2023-04-21 오전 1:00	3362442	8-2.exe
-a----	2023-04-19 오전 10:40	65	dummy.txt
-a----	2023-04-20 오후 3:44	68	eicar.txt
-a----	2023-04-21 오전 12:55	2856	scanmod.py
-a----	2023-04-20 오후 10:08	215	virus.kmd

- 동적으로 모듈을 import 할 수 있도록 수정한 백신 파일을 실행파일로 생성하였다.
- 모듈을 정상적으로 import 하는지 scanmod 파일에서 scanVirus 함수를 약간 수정하였다.
- scanmod 파일을 실행파일이 있는 디렉터리에 저장한 다음에 실행파일을 실행해본다.

```
PS C:\Users\kocan\Desktop\asdf\dist> ./8-2.exe eicar.txt
[*] New ScanVirus
eicar.txt : EICAR Test
```

```
PS C:\Users\kocan\Desktop\asdf\dist> ./8-2.exe dummy.txt
[*] New ScanVirus
dummy.txt : Dummy Test
```

- eicar.txt와 dummy.txt 모두 악성코드로 잘 판별되는 모습을 볼 수 있다.
- 수정된 scanmod 파일이 동적으로 import 되는 모습을 New ScanVirus 메시지를 통해 확인할 수 있다.

```
PS C:\Users\kocan\Desktop\asdf\dist> c:\Python27\python.exe -m compileall scanmod.py
Compiling scanmod.py ...
PS C:\Users\kocan\Desktop\asdf\dist> dir
```

디렉터리: C:\Users\kocan\Desktop\asdf\dist

Mode	LastWriteTime	Length	Name
-a----	2023-04-21 오전 12:40	3362271	7-15.exe
-a----	2023-04-21 오전 12:41	3362397	8-1.exe
-a----	2023-04-21 오전 1:00	3362442	8-2.exe
-a----	2023-04-19 오전 10:40	65	dummy.txt
-a----	2023-04-20 오후 3:44	68	eicar.txt
-a----	2023-04-21 오전 12:55	2856	scanmod.py
-a----	2023-04-21 오전 1:02	1492	scanmod.pyc
-a----	2023-04-20 오후 10:08	215	virus.kmd

- scanmod 파일을 소스 공개하지 않고 배포하기 위해 컴파일을 진행한다.
- python 폴더에 있는 python 실행파일에 옵션을 입력하여 컴파일을 진행한다.
- 컴파일 진행 후, dir 명령어를 통해 파일 리스트를 살펴보면 컴파일된 scanmod 파일을 확인할 수 있다.

### 3) 느낀 점

- 이번 프로젝트를 통해 악성코드 백신을 탐지하고 치료하는 실습을 진행할 수 있었다. 평소에 안티바이러스 프로그램을 사용하면서 악성코드를 어떻게 탐지하고 치료하는지 궁금했었는데 이번 프로젝트를 통해 대강 알게 되었다. 물론 현재 상용하는 프로그램들은 탐지 알고리즘이 훨씬 복잡하겠지만 탐지에 대한 원리는 알 수 있어서 큰 이득이었다. 악성코드 DB의 암호화와 복호화를 위해 해싱 함수가 제공되는 python을 이용하여 코드를 작성하게 되었는데 평소에 C, C++을 자주 사용하다 보니 세미콜론을 붙이게 된다. 습관이 참 무섭다. 실습에 있는 교안이 아무래도 예전 내용이라 그런지 이전 버전의 python을 요구하는데, 이것 최신 python으로 작성하려니 버전별로 사용하는 함수가 조금씩 달라져서 결국 이전 버전의 python을 사용하게 되었다. 또한 pyinstaller 또한 교안에 있는 대로 파일을 다운받고 압축 풀어서 사용하려고 했으나 원인 모를 에러가 계속

출력되어 결국 python 내의 pip install 명령어를 이용해서 pyinstall의 여러 버전을 설치하고 테스트를 진행했다. 이렇게 중간에 사소하게 문제를 일으키는 부분이 존재하여 이걸 해결하느라 시간이 좀 소요되었으나 결국 무사히 이번 프로젝트를 끝낼 수 있었다. 이번 프로젝트에서는 비교적 간단한 악성코드 탐지에 대해 실습을 진행했는데 현대에 존재하는 바이러스가 이렇게 간단하게 생기지는 않았을 것으로 생각한다. 그러면 이걸 빠르고 효율적으로 탐지하는 방법이 있을 텐데 그 방법이 무엇인지 궁금해졌다. 빠르게 탐지하려면 해싱부터 빠르게 진행하거나 풀어야 하는데 이걸 가능케 하는 방법이 존재할까? 있다면 어떤 방법일까 너무나 궁금하다. 이걸 수업 시간에 배울지는 잘 모르겠지만 나중에 기회가 있다면 한번 알아보고 싶다. 몇 주 만에 windows 환경에서 진행하는 실습이었는데 쾌적하게 진행되었다고 생각한다. 물론 나는 리눅스에서 하는 실습도 재미있기에 몇 번 정도는 리눅스 환경에서도 실습 진행하고 싶다. 다음 실습에도 이렇게 재미있는 주제로 실습을 진행했으면 좋겠다.