

[REPORT - 실습 6]



과 목 명 : 컴퓨터보안 - 02

교 수 : 김영부 교수님

학번 : 2020112119

이름 : 강동희

제출일 : 2023.05.02

목 차

1. 서론

1.1. 문제 분석

2. 본론

2.1. 실습 1 : 전용백신 개발하기

2.2. 실습 2 : 다양한 악성코드 진단 및 치료하기

2.3. 실습 3 : 악성코드 패턴 분리하기

2.4. 실습 4 : 악성코드 진단 및 치료 모듈 분리하기

2.5. 실습 5 : 전용백신 배포본 만들기

3. 결론

3.1. 실습 결과 및 분석

3.2. 느낀 점

4. 소스 코드

1 서론

1.1 문제 분석

주제 1. 전용백신 개발하기

EICAR Test 파일을 악성코드 진단 문자열과 md5 해시를 사용하여 악성코드를 진단하는 전용백신을 테스트한다.

해당 텍스트 파일을 통해 [파일 읽기]-[파일의 내용과 악성코드 진단문자 비교하기]-[악성코드 치료(삭제)하기]-[파이썬의 hashlib을 이용하여 MD5해시 구하기] -[EICAR Test 파일 검사]의 과정을 통해 전용 백신의 동작을 확인할 수 있다.

주제 2. 다양한 악성코드 진단 및 치료하기

Dummy Test(새로운 악성코드)를 정의하여 다양한 악성코드를 진단 및 치료할 수 있는 백신을 만들어본다.

[Dummy Test 악성코드 생성] - [악성코드 진단 파이썬 코드 구현] - [악성코드 DB 정의] - [악성코드 DB를 이용한 악성코드 검사 파이썬 코드 구현] - [VirusDB에 파일크기 추가]의 과정을 통해 악성코드를 검사하는 전용백신 프로그램을 구현해본다.

주제 3. 악성코드 패턴 분리하기

악성코드 패턴과 실행파일을 분리하여 다양한 악성코드의 패턴이 추가될 때마다 업데이트할 수 있도록 악성코드 패턴을 분리한다.

[Viruns DB 코드 구현]-[DB파일에 저장한 악성코드 패턴 생성]-[악성코드 패턴 로딩 코드]의 과정을 통해 악성코드 패턴 파일을 수정하고 배포할 수 있음을 확인한다.

주제 4. 악성코드 진단 및 모듈 분리하기

악성코드 중 기존의 진단 방식으로 진단되지 않는 악성코드에 대비해 새로운 진단 방식과 치료 방식을 설계하여 백신코드에 반영해본다.

[악성코드 검사 부분 분리]-[백신의 스캔과 탐색 모듈 합치기]-[탐색(전체위치, 특정위치) 진단 구현]-[치료 모듈 분리]의 과정을 통해서 새로운 진단 방식과 치료 방식에 대한 업데이트를 진행할 수 있도록 구현해볼 수 있다.

주제 5. 전용백신 배포본 만들기

지금까지 진행한 전용 백신을 배포본으로 생성한다. 실행 파일로 변환하는 도구를 통해 백신을 하나의 실행파일로 만들어 본다.

PyInstaller를 사용하여 백신 소스코드를 실행파일로 변환하여 배포본을 생성해볼 수 있다.

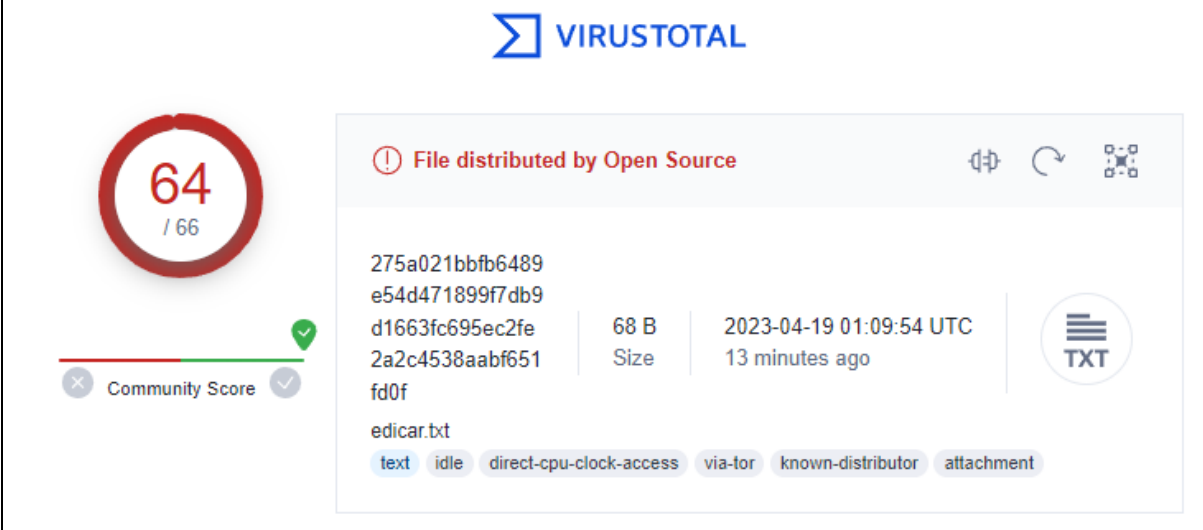
2 본 론

2.1 실습 1 전용백신 개발하기

(1) eicar.txt 파일 생성하기

Task1 > `≡ eicar.txt`

```
1 X50!P%@AP[4\PZX54(P^)7CC)7]$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```



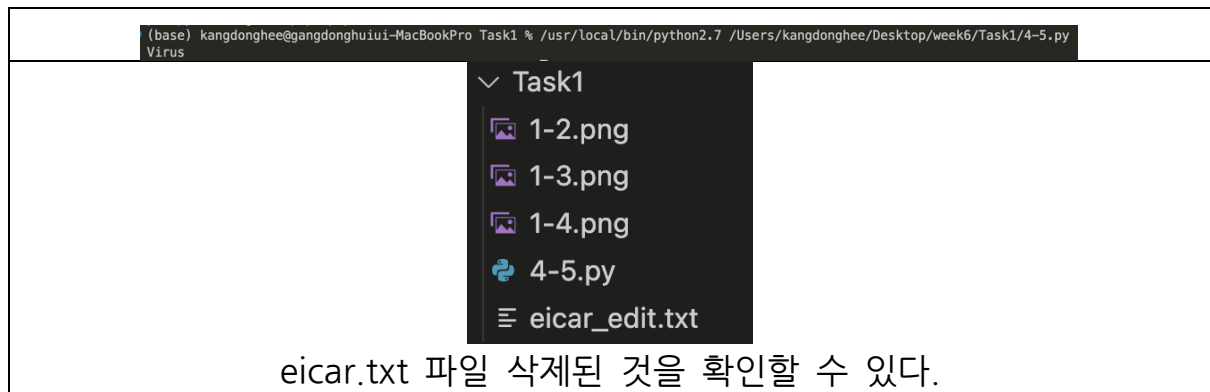
The image shows the VirusTotal interface for a file named 'eicar.txt'. On the left, there is a large red circle with the number '64' inside, indicating a community score of 64 out of 66. Below this is a green checkmark and the text 'Community Score'. On the right, there is a box with the text 'File distributed by Open Source'. Below this, there is a table with the following information: MD5 hash '275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabbf651fd0f', file size '68 B', and upload time '2023-04-19 01:09:54 UTC' (13 minutes ago). The file type is 'TXT'. At the bottom, there are tags for 'text', 'idle', 'direct-cpu-clock-access', 'via-tor', 'known-distributor', and 'attachment'.

(2) 악성코드 진단 문자열과 md5 해시를 이용하여 EICAR Test(악성코드)을 진단하는 전용백신을 테스트해 보아라 (리스트 4-5 수행)

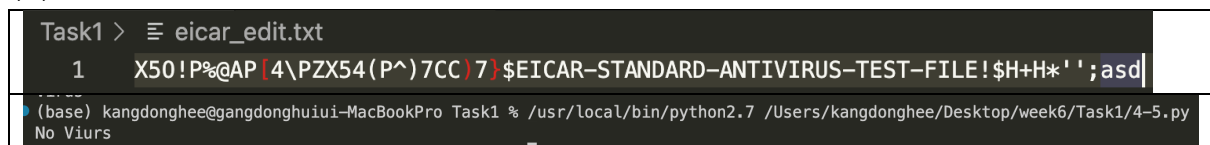
```
(base) kangdonghee@gangdonghuiui-MacBookPro Python-2.7.14 % python
Python 2.7.14 (default, May 2 2023, 21:56:03)
[GCC Apple LLVM 14.0.3 (clang-1403.0.22.14.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Task1 > 4-5.py > ...
1 import hashlib
2 import os
3
4 #text 파일 바이너리 읽기 모드(rb)로 오픈
5 fp=open('/Users/kangdonghee/Desktop/week6/Task1/eicar.txt', 'rb')
6 fbuf=fp.read()
7 fp.close()
8
9
10 m=hashlib.md5()
11 #md5 해쉬 구하기
12 m.update(fbuf)
13 #file의 md5값을 저장
14 fmd5=m.hexdigest()
15
16 #파일의 md5 해시 값이 eicar 바이러스 해시값과 일치할 경우
17 if fmd5 == "44d88612fea8a8f36de82e1278abb02f":
18     #Virus 출력 및 파일 삭제
19     print("Virus")
20     os.remove('eicar.txt')
21 else:
22     #아닐 경우 No Virus 출력
23     print("No Viurs")
24
```

(3) EICAR TEST 파일 검사 수행하기

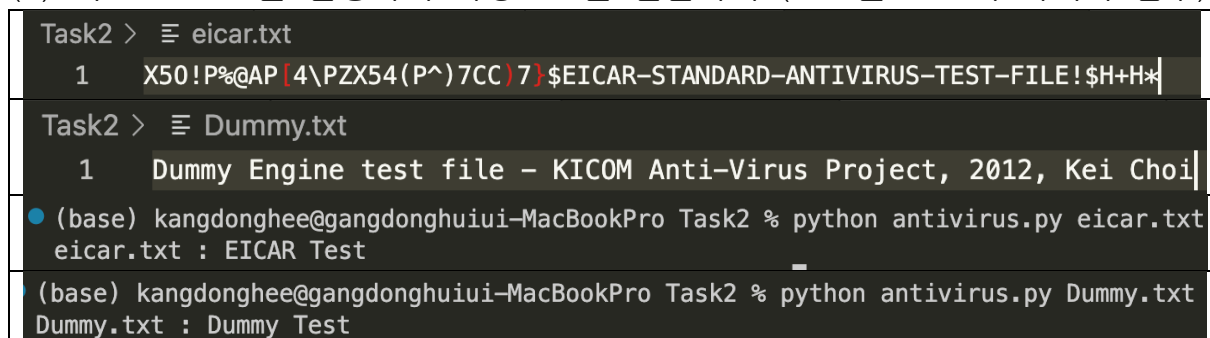


(4) eicar.txt 를 수정하여 바이러스가 아닌 경우 코드 실행

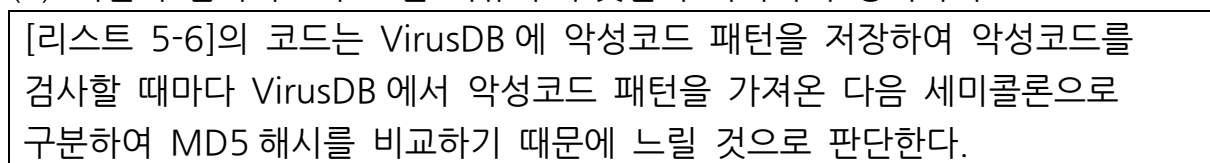


2.2 실습 2 다양한 악성코드 진단 및 치료하기

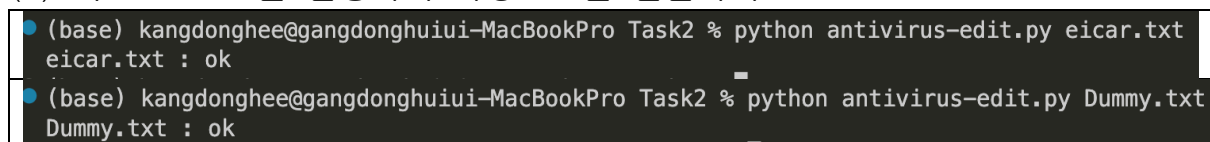
(1) 리스트 5-6 을 실행하여 악성코드를 진단하기 (코드는 보고서 마지막 첨부)



(2) 백신의 검사속도가 느린 이유가 무엇인지 파악하여 정리하기



(3) 리스트 5-9 을 실행하여 악성코드를 진단하기



(4) 리스트 5-6 와 리스트 5-9 의 코드를 각각 실행하였을 때 속도 차이를 확인

VirusDB 의 악성코드 패턴에 악성코드의 파일크기를 추가하여 검사 대상 파일의 크기가 악성코드 DB 에 등록된 악성코드 파일의 크기와 일치한다면 불필요하게 악성코드를 검사할 필요가 없기 때문에 리스트 5-9, 즉 악성코드의 파일크기를 추가한 코드가 실행속도가 더 빠른 것을 확인할 수 있었다.

2.3 실습 3 악성코드 패턴 분리하기

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

```
Task3 > ≡ virus.db
1 68:44d88612fea8a8f36de82e1278abb02f:EICAR Test
2 65:77bff0b143e4840ae73d4582a8914a43:Dummy Test

(base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py eicar.txt
eicar.txt : EICAR Test
(base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py Dummy.txt
Dummy.txt : Dummy Test
```

(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 코드이다. 이를 실행하여 virus.db 를 암호화한 virus.kmd 를 생성해보자. (소스코드는 보고서 마지막에 첨부)

```
≡ virus.db
≡ virus.kmd
```

(3) [리스트6-6]의 악성코드 패턴 파일을 복호화하는 파이썬 코드(DecodeKMD 함수)를 분석해보자.

KMD 파일을 디코딩하여 원본 데이터를 가져오는 함수를 구현한 것이다. KMD 파일의 내용을 읽어와 MD5 해시 값을 검증한 뒤, 내용을 XOR 연산으로 복호화하고 zlib 으로 압축 해제하여 반환하는 함수이다.

(4) antivirus.py 에 DecodeKMD 함수를 적용하여 virus.kmd 로부터 악성코드 패턴을 로딩 하여 보자.

```
(base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py Dummy.txt
Dummy.txt : ok
(base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py eicar.txt
eicar.txt : ok
```

2.4 실습4 악성코드 진단 및 치료 모듈 분리하기

(1) SearchVDB 와 ScanMD5 를 추출하고 scanmod 로 모듈화한 백신 프로그램 (antivirus.py) 인 [리스트 7-4]를 실행하여 보아라. 결과는 [그림 7-1]과 같아야 한다.

```
● (base) kangdonghee@gangdonghuiui-MacBookPro Task4 % python antivirus.py eicar.txt  
eicar.txt : ok
```

(2) [리스트7-5]의 특정위치 검색법(scan_str.py)을 사용하여 악성코드를 검사하여 보아라. 결과는 [그림7-2]와 같아야 한다.

```
● (base) kangdonghee@gangdonghuiui-MacBookPro Task4 % python scan_str.py  
False
```

(3) MD5해쉬와 특정위치 검색법을 이용할 수 있도록 악성코드 패턴 파일 (virus.db)을 수정하고 이를 주제3에서 다루었던 kmake.py를 이용하여 암호화한다. 결과는 [그림7-3]과 같아야 한다.

(4) 7.4절을 포함하여 7장에서 언급된 모든 내용을 백신에 반영하여 수정하고 악성코드를 진단하여 보아라. 결과는 [그림7-4]와 같아야 한다. Dummy Test 악성코드는 md5 해쉬를 이용해서 검사했고 EICAR Test 악성코드는 특정위치 검색법을 이용하여 검사했는지 확인하여라.

2.5 실습 5 전용백신 배포본 만들기

3 결 론

3.1 실습 결과 및 분석

실습 1 : 전용백신 개발하기

해당 텍스트 파일을 통해 [파일 읽기]-[파일의 내용과 악성코드 진단문자 비교하기]-[악성코드 치료(삭제)하기]-[파이썬의 hashlib 을 이용하여 MD5 해시 구하기]-[EICAR Test 파일 검사]의 과정을 통해 전용 백신의 동작을 확인할 수 있다.

```
(base) kangdonghee@gangdonghuiui-MacBookPro Task1 % /usr/local/bin/python2.7 /Users/kangdonghee/Desktop/week6/Task1/4-5.Virus
```

실습 2 : 다양한 악성코드 진단 및 치료하기

[Dummy Test 악성코드 생성] - [악성코드 진단 파이썬 코드 구현] - [악성코드 DB 정의] - [악성코드 DB 를 이용한 악성코드 검사 파이썬 코드 구현] - [VirusDB 에 파일크기 추가]의 과정을 통해 악성코드를 검사하는 전용백신 프로그램을 구현해볼 수 있었다.

[리스트 5-6]의 코드는 VirusDB 에 악성코드 패턴을 저장하여 악성코드를 검사할 때마다 VirusDB 에서 악성코드 패턴을 가져온 다음 세미콜론으로 구분하여 MD5 해시를 비교하기 때문에 느릴 것으로 판단한다.

VirusDB 의 악성코드 패턴에 악성코드의 파일크기를 추가하여 검사 대상 파일의 크기가 악성코드 DB 에 등록된 악성코드 파일의 크기와 일치한다면 불필요하게 악성코드를 검사할 필요가 없기 때문에 리스트 5-9, 즉 악성코드의 파일크기를 추가한 코드가 실행속도가 더 빠른 것을 확인할 수 있었다.

실습 3 : 악성코드 패턴 분리하기

[Viruns DB 코드 구현]-[DB 파일에 저장한 악성코드 패턴 생성]-[악성코드 패턴 로딩 코드]의 과정을 통해 악성코드 패턴 파일을 수정하고 배포할 수 있음을 확인할 수 있었다.

```
• (base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py Dummy.txt  
Dummy.txt : ok  
• (base) kangdonghee@gangdonghuiui-MacBookPro Task3 % python antivirus.py eicar.txt  
eicar.txt : ok
```

실습 4 : 악성코드 진단 및 치료 모듈 분리하기

[악성코드 검사 부분 분리]-[백신의 스캔과 탐색 모듈 합치기]-[탐색(전체위치, 특정위치) 진단 구현]-[치료 모듈 분리]의 과정을 통해서 새로운 진단 방식과 치료 방식에 대한 업데이트를 진행할 수 있도록 구현해볼 수 있었다.

실습 5 : 전용백신 배포본 만들기

PyInstaller 를 사용하여 백신 소스코드를 실행파일로 변환하여 배포본을 생성해볼 수 있었다.

3.2 느낀 점

6주차 실습을 진행하면서 악성 코드의 백신을 직접 구현해보면서 백신이 어떤 구성으로 이루어져 있는지 모듈을 분리해보고 구현을 통해서 알 수 있었다. 특히 백신의 검사 속도를 높이기 위해 파일의 크기를 통해서 판단하여 검사하는 방식을 통해서 MD5 해시 값의 Signature로 인해서 파일 크기가 고정되어 가능하다는 점을 알 수 있었다. 또한 악성 코드 패턴을 실행파일과 분리하여 수정 및 배포에 용이하도록 하여 관리 및 업데이트를 위한 작업도 알 수 있었다. 그리고 악성코드가 새로운 진단 방식, 치료 방식을 설계하기 위해서 해당 모듈도 분리해보고 파이썬 파일을 실행코드로 변환해보면서 프로그램을 설계 및 배포할 수 있는 좋은 경험이었다.

4. 실습 코드

[1] 실습 1

```
# -*- coding: utf-8 -*-
import sys
import os
import hashlib
VirusDB = [
    '44d88612fea8a8f36de82e1278abb02f:EICAR Test',
    '77bffa0b143e4840ae73d4582a8914a43:Dummy Test'
]
vdb = []
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':')
        t.append(v[0])
        t.append(v[1])
        vdb.append(t)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5:
            return True, t[1]
    return False, ''
if __name__ == '__main__':
    MakeVirusDB()
    if len(sys.argv) != 2:
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1]
    fp = open(fname, 'rb')
    buf = fp.read()
    fp.close()
    m = hashlib.md5()
    m.update(buf)
    fmd5 = m.hexdigest()
    ret, vname = SearchVDB(fmd5)
    if ret == True:
        print('%s : %s' %(fname, vname))
        os.remove(fname)
    else:
        print('%s : ok' %(fname))
```

[2] 실습 2

antivirus.py

```
# -*- coding: utf-8 -*-
import sys
import os
import hashlib
VirusDB = [
    '44d88612fea8a8f36de82e1278abb02f:EICAR Test',
    '77bffa0b143e4840ae73d4582a8914a43:Dummy Test'
]
vdb = []
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':')
```

```

        t.append(v[0])
        t.append(v[1])
        vdb.append(t)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5:
            return True, t[1]
    return False, ''
if __name__ == '__main__':
    MakeVirusDB()
    if len(sys.argv) != 2:
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1]
    fp = open(fname, 'rb')
    buf = fp.read()
    fp.close()
    m = hashlib.md5()
    m.update(buf)
    fmd5 = m.hexdigest()
    ret, vname = SearchVDB(fmd5)
    if ret == True:
        print('%s : %s' %(fname, vname))
        os.remove(fname)
    else:
        print('%s : ok' %(fname))

```

antivirus-edit.py

```

# -*- coding: utf-8 -*-
import sys
import os
import hashlib
VirusDB = [
    '68:44d88612fea8a8f36de82e1278abb02f:EICAR Test',
    '65:77bff0b143e4840ae73d4582a8914a43:Dummy Test'
]
vdb = []
vsize = []
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':')
        t.append(v[0])
        t.append(v[1])
        vdb.append(t)
        size = int(v[0])
        if vsize.count(size) == 0:
            vsize.append(size)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5:
            return True, t[1]
    return False, ''
if __name__ == '__main__':
    MakeVirusDB()
    if len(sys.argv) != 2:
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1]
    fp = open(fname, 'rb')
    buf = fp.read()
    fp.close()
    m = hashlib.md5()
    m.update(buf)

```

```

fmd5 = m.hexdigest()
ret, vname = SearchVDB(fmd5)
if ret == True:
    print('%s : %s' %(fname, vname))
    os.remove(fname)
else:
    print('%s : ok' %(fname))

```

[3] 실습 3

antivirus.py

```

# -*- coding: utf-8 -*-
import sys
import os
import hashlib
import zlib
from io import StringIO
VirusDB = [] #악성코드 패턴은 모두 virus.db 에 존재함
vdb = [] #가공된 악성코드 DB 가 저장된다.
vsize = [] #악성코드의 파일 크기만 저장한다.
def DecodeKMD(fname):
    try:
        fp = open(fname, 'rb')
        buf = fp.read()
        fp.close()
        buf2 = buf[:-32]
        fmd5 = buf[-32:]
        f = buf2
        for i in range(3) :
            md5 = hashlib.md5()
            md5.update(f)
            f = md5.hexdigest()
            if f != md5:
                raise SystemError

        buf3 = ''
        for c in buf2[4:] :
            buf3 += chr(ord(c) ^ 0xFF)
        buf4 = zlib.decompress(buf3)
        return buf4
    except:
        pass
    return None
# virus.db 파일에서 악성코드 패턴을 읽는다.
def LoadVirusDB():
    buf = DecodeKMD('virus.kmd')
    fp = StringIO(buf)
    while True:
        line = fp.readline()
        if not line : break

        line = line.strip()
        line = line.decode('utf-8')
        VirusDB.append(line)
    fp.close()
def MakeVirusDB():
    for pattern in VirusDB:
        t = []
        v = pattern.split(':')
        t.append(v[1])

```

```

        t.append(v[2])
        vdb.append(t)
        size = int(v[0])
        if vsize.count(size) == 0:
            vsize.append(size)
def SearchVDB(fmd5):
    for t in vdb:
        if t[0] == fmd5:
            return True, t[1]
    return False, ''
if __name__ == '__main__':
    LoadVirusDB()
    MakeVirusDB()
    if len(sys.argv) != 2:
        print('Usage : antivirus.py [file]')
        exit(0)
    fname = sys.argv[1]
    size = os.path.getsize(fname)
    if vsize.count(size):
        fp = open(fname, 'rb')
        buf = fp.read()
        fp.close()
        m = hashlib.md5()
        m.update(buf)
        fmd5 = m.hexdigest()
        ret, vname = SearchVDB(fmd5)
        if ret == True:
            print('%s : %s' %(fname, vname))
            os.remove(fname)
        else:
            print('%s : ok' %(fname))
    else:
        print('%s : ok' %(fname))

```

kamke.py

```

# -*- coding: utf-8 -*-
import sys
import zlib
import hashlib
import os
def main():
    if len(sys.argv) != 2:
        print('Usage : kmake.py [file]')
        return

    fname = sys.argv[1]
    tname = fname
    fp = open(tname, 'rb')
    buf = fp.read()
    fp.close()
    buf2 = zlib.compress(buf)
    buf3 = ''
    print(buf2)
    for c in buf2:
        buf3 += chr(ord(c) ^ 0xFF)
    buf4 = 'KAVM' + buf3
    f = buf4
    for i in range(3):
        md5 = hashlib.md5()
        md5.update(f)
        f = md5.hexdigest()
    buf4 += f
    kmd_name = fname.split('.')[0] + '.kmd'
    fp = open(kmd_name, 'wb')

```

```

fp.write(buf4)
fp.close()
print('%s -> %s' %(fname, kmd_name))
if __name__ == '__main__':
    main()

```

[4] 실습 4

Scan_str.py

```

# -*-coding:utf-8 -*-
def ScanStr(fp, offset, mal_str) :
    size = len(mal_str)
    fp.seek(offset)
    buf = fp.read(size)
    if buf == mal_str :
        return True
    else :
        return False
fp = open('/Users/kangdonghee/Desktop/week6/Task4/eicar.txt', 'rb')
print(ScanStr(fp, 0, 'X50'))
fp.close()

```

Scanmod.py

```

# -*- coding:utf8 -*-
import os
import hashlib
def SearchVDB(vdb, fmd5):
    for t in vdb:
        if t[0] == fmd5:
            return True, t[1]
    return False, ''
def ScanMD5(vdb, vsize, fname):
    ret = False
    vname = ''
    size = os.path.getsize(fname)
    if vsize.count(size):
        fp = open(fname, 'rb')
        buf = fp.read()
        fp.close()
        m = hashlib.md5()
        m.update(buf)
        fmd5 = m.hexdigest()
        ret, vname = SearchVDB(vdb, fmd5)
    return ret, vname

```

[5] 실습 5
