

컴퓨터 보안_01

실습 6주차

동국대학교 CSDC Lab.

실습조교 김선규

2022.04.19 (Wed.)

1. 프로젝트: 악성코드 백신 구현

- 주 제 1 : 전용백신 개발하기
- 주 제 2 : 다양한 악성코드 진단 및 치료하기
- 주 제 3 : 악성코드 패턴 분리하기
- 주 제 4 : 악성코드 진단 및 치료 모듈 분리하기
- 주 제 5 : 전용백신 배포본 만들기

2. Q & A

➤ 주 제 1 : 전용백신 개발하기

- 백신은 생각보다 훨씬 규모가 큰 소프트웨어이다. 따라서 분할과 정복(divide and conquer)기법을 통해 작은 프로젝트부터 공략해서 큰 프로젝트로 확장하는 것이 좋다. 전용백신은 특정한 악성코드 하나를 진단 및 치료하는 백신을 의미한다. 즉, 일반적인 백신보다 속도 측면에서 빠르고 진단 측면에서 정확하다는 장점이 있다. 그러나 범용백신에 비해 다양한 악성코드를 진단 및 치료하지 못하기 때문에 매번 해당 악성코드에 대한 백신을 찾아서 실행을 해야 한다.
- 일반적으로 백신 사용자들은 자신이 설치한 백신이 정상적으로 동작하고 있는지 궁금할 때가 많다. 사실 백신은 악성코드가 시스템에 유입되기 전까지는 정확한 동작유무를 확인하기 어렵기 때문이다. 이제 전용 백신을 만들어 보자. 전용백신을 만들려면 일단 해당 악성코드가 먼저 있어야 한다. 그래서 EICAR Test파일을 진단 및 치료할 수 있는 EICAR전용백신을 만들어 보려고 한다. 60여가지의 백신을 모아둔 바이러스토탈(<http://www.virustotal.com>)에서 EICAR Test파일을 진단하는 백신들을 확인해 볼 수 있다. EICAR Test 파일은 참고자료(4장 4.2.2절을 참조)의 내용을 통해서 만들 수 있는데 파일명은 eicar.txt로 저장하면 된다.
- 참고자료 4장의 [리스트4-1] 파일 읽기, [리스트4-2] 파일의 내용과 악성코드 진단문자 비교하기, [리스트4-3] 악성코드 치료(삭제)하기, 그리고 [리스트4-4] 파이썬의 hashlib를 이용하여 MD5해시 구하기를 이해한 후에, 전용백신이 동작을 확인하기 위해 [리스트4-5]를 통해 EICAR Test 파일을 검사할 수 있다.

➤ 주 제 1 : 전용백신 개발하기

- 아래의 각 문제에 대한 파이썬 코드의 수행 결과를 캡처하여 분석하고 설명하는 내용을 개인적인 견해와 함께 레포트에 모두 반영하여야 한다.

1. http://www.eicar.org/anti_virus_test_file.htm을 방문하여 웹페이지 중간 썸에 위치한 문장을 참조하여 eicar.txt 파일을 만든다 (4.2.2.절 참조).

X5O!P%@AP[4\PZX54(P^7CC)7}\$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\$H+H*

1. [리스트 4-1] ~ [리스트4-4]의 내용을 익히고 테스트해본다.
2. 악성코드 진단 문자열과 md5해시를 이용하여 EICAR Test파일(악성코드)을 진단하는 전용백신을 테스트해 보아라 (리스트4-5 수행).

(주의사항) 참고자료에 제시된 파이썬 코드의 실행을 위한 **파이썬 버전은 2.7**(<https://www.python.org/download/releases/2.7/>)이다.

- 악성코드 진단을 목적으로 할때는 반드시 **바이너리 읽기 모드(rb)**로 파일을 엽니다.
- 악성코드 진단 문자열을 이용하여 악성코드를 진단할 때는 **최소 10Bytes 이상**을 사용하는 것이 좋다.
- MD5 해시를 이용하여 진단할 수 있는 악성코드는 바이러스 유형을 제외한 파일 그 자체가 악성코드인 트로이목마, 백도어, 웜 등이다.

➤ 주 제 2 : 다양한 악성코드 진단 및 치료하기

- 일반적으로 사용자는 자신의 시스템 내에 어떤 악성코드가 존재하는지 확인하기 어렵다. 즉, 시스템이 안전한지 확인하려면 모든 종류의 전용백신을 모두 설치하고 실행해야만 한다. **전용백신**은 사회적으로 이슈가 되는 특정 악성코드가 자신의 시스템에 존재하는지 확인하기에는 편리하지만 시스템 전반의 **일반적인 보안 점검에는 사용하기 어렵다**는 단점이 있다.
- 이제 전용백신을 수정해서 다양한 악성코드를 진단 및 치료할 수 있는 백신을 만들어 보자. 앞의 주제1을 통해 현재 우리는 EICAR Test라는 하나의 악성코드만 진단할 수 있다. 테스트를 위해 새로운 악성코드(이를 **Dummy Test**라고 하자)를 하나 더 정의하자. [리스트5-2]를 통해 이 Dummy Test를 진단할 수 있는 파이썬 코드를 확인할 수 있다. 이 코드는 if문을 통해 매번 새로운 악성코드를 체크하는 부분을 바꾸어 주어야 한다. 다양한 악성코드의 MD5해시를 관리하기 위해서 파이썬의 자료구조를 이용한다. [리스트5-3]은 **파이썬의 리스트를 이용하여 악성코드 DB**를 별도로 정의할 수 있음을 보여준다.

Dummy Engine test file - KICOM Anti-Virus Project, 2012, Kei Choi

- **VirusDB**에 저장된 악성코드 패턴은 문자열 두 개(MD5해시와 악성코드 이름)가 세미콜론(:)으로 구분되어 있다. 악성코드를 검사할 때마다 VirusDB에서 악성코드 패턴을 가져온 다음 세미콜론으로 구분하여 MD5해시를 비교한다. MD5해시가 일치한다면 악성코드 이름을 출력해야 하는데 진단속도를 빠르게 하기 위해서는 미리 MD5해시와 **악성코드 이름을 구분해 두는 것**이 좋다. 이런 목적으로 [리스트5-4]에 VirusDB를 가공하는 함수를 선언하고 있다.

- 악성코드 DB를 최종적으로 가공한 vdb를 이용해서 악성코드를 검사할 수 있는 파이썬 코드가 [리스트5-6]에 주어져 있다. 백신은 악성코드를 잘 진단하는 것도 중요하지만 검사 속도가 느린 백신은 사용자에게 외면 받을 수 있다. 백신의 검사 속도를 높이기 위하여 VirusDB의 악성코드 패턴에 악성코드의 파일크기를 추가할 수 있다([리스트5-8]). 만약 검사 대상 파일의 크기가 악성코드 DB에 등록된 악성코드 파일의 크기와 일치한다면 불필요하게 악성코드를 또 검사할 필요가 없어진다. [리스트5-9]는 검사속도를 높이기 위해서 수정된 백신의 파이썬 코드이다.

➤ [과제 내용]

- 아래의 각 문제에 대한 파이썬 코드의 수행 결과를 캡처하여 분석하고 설명하는 내용을 개인적인 견해와 함께 레포트에 모두 반영하여야 한다.
1. [리스트 5-6]을 실행하여 악성코드를 진단해보자.
 2. 백신의 검사속도가 느린 이유가 무엇인지 파악하여 정리하여 보자.
 3. [리스트 5-9]을 실행하여 악성코드를 진단해보자.
 4. [리스트 5-6]와 [리스트 5-9]의 코드를 각각 실행 하였을때 속도 차이를 확인할 수 있는가? Yes or No? 그 이유는 무엇인가?

➤ 주 제 3 : 악성코드 패턴 분리하기

- 주제2에서 다양한 악성코드를 진단 및 치료할 수 있는 전용백신을 완성하였다. 하지만 매번 악성코드의 패턴이 추가될 때마다 계속 악성코드 DB파일(antivirus.py)을 수정하는 것은 무리가 있다. 이어지는 향후 주제에서 **antivirus.py**은 실행파일로 바뀌게 된다. 실행파일을 수정하고 배포하는 것은 엄격한 테스트를 필요로 한다. 악성코드를 빨리 퇴치하기 위해서는 가급적 테스트를 줄여야 하는데 실행파일을 수정하는 것은 좋은 접근법이 아니다. 일반적으로 백신업체들은 **악성코드 패턴을 실행파일과 분리하여 별도의 파일로 배포**하고 있다. 여기서도 그렇게 해보자.
- **[리스트6-1]**은 악성코드 패턴이 담긴 VirusDB를 가진 antivirus.py 파이썬 코드이다. **[리스트6-2]**는 별도의 DB파일(virus.db)로 저장한 악성코드 패턴이다. 그러면 이제 antivirus.py의 VirusDB에 저장되어 있던 악성코드 패턴은 더 이상 필요가 없게 된다. 대신에 virus.db파일에 저장되어 있는 악성코드 패턴을 로딩 하는 **LoadVirusDB 함수**를 설계할 필요가 있다 (**리스트6-3 참조**).
- 백신과 악성코드 패턴이 분리되어 있기 때문에 이후에 발견되는 악성코드의 패턴을 **악성코드 패턴 파일(virus.db)**에 추가하고 사용자에게 배포하면 된다. 하지만 virus.db파일은 누구나 메모장으로 열어서 편집이 가능하기 때문에 해커도 이를 마음대로 조작할 수 있게 된다. 따라서 악성코드 패턴 파일은 오로지 백신업체에서만 수정하고 배포할 수 있어야만 한다. **즉, 악성코드 패턴 파일을 암호화하면 아무나 파일을 조작할 수 없게 된다.**

➤ [과제 내용]

- 아래의 각 문제에 대한 파이썬 코드의 수행 결과를 캡처 하여 분석하고 설명하는 내용을 개인적인 견해와 함께 레포트에 모두 반영하여야 한다.
1. [리스트6-4]에 제시된 별도의 파일로 분리된 악성코드 패턴을 로딩 하여 악성코드를 진단 및 치료하는 antivirus.py 코드를 실행시켜보자.
 2. 악성코드 패턴 파일(virus.db)을 암호화하는 도구는 백신업체만 가지고 있어야 한다. 따라서 별도의 도구로 만들어야 한다. [리스트6-5]는 악성코드 패턴 파일을 암호화하는 파이썬 코드(kmake.py)이다. 이를 실행하여 virus.db를 암호화한 virus.kmd를 생성하여 보자.
 3. antivirus.py에서는 암호화된 virus.kmd를 바로 로딩 할 수 없다. 따라서 이를 복호화한 다음 로딩 할 수 있도록 복호화 모듈이 필요하다. [리스트6-6]의 악성코드 패턴 파일을 복호화하는 파이썬 코드(DecodeKMD함수)를 분석해보자.
 4. 이제 antivirus.py에 DecodeKMD 함수를 적용하여 virus.kmd로부터 악성코드 패턴을 로딩 하여 보자 (이때 StringIO 모듈의 fp.readline 함수가 사용된다.). 즉, [리스트6-7]을 실행시켜보자.

➤ 주 제 4 : 악성코드 진단 및 치료 모듈 분리하기

- 백신에서 업데이트가 가장 빈번한 것은 악성코드 패턴이다. 주제3에서 우리는 이를 별도의 파일로 분리했다. 이 파일만 업데이트하면 웬만한 악성 코드는 진단 및 치료를 할 수 있다. 하지만 다양한 악성코드를 분석하다 보면 악성코드 중에는 기존의 진단 방식으로는 진단되지 않는 악성코드도 있고, 기존 치료 방식으로 치료되지 않는 악성코드도 존재한다. 이런 경우가 발생하면 우리는 새로운 진단 방식과 치료방식을 설계해서 백신코드에 반영해야 한다. 즉, antivirus.py 소스코드를 수정한 다음 다시 배포해야 한다. 그렇다면 악성코드 패턴파일과 마찬가지로 악성코드 진단 및 치료 모듈도 분리할 수 있지 않을까? 이제부터 이런 방법에 대해서 알아본다.
- [리스트7-2]는 기존의 antivirus.py에서 MD5해시를 이용한 악성코드 검사부분을 별도의 함수(ScanMD5)로 추출한 것을 보여준다. 즉, 악성코드 진단 부분만 두 개의 함수(SearchVDB, ScanMD5)로 추출하여 별도의 파일인 scanmod.py 파일로 저장한다(리스트7-3). 따라서 이전의 antivirus.py에서 SearchVDB와 ScanMD5를 떼어낸 antivirus.py도 scanmod모듈을 포함하도록 수정되어야 한다 (리스트7-4).
- 여태까지 우리는 악성코드를 진단하는 2가지 방법을 살펴보았다. MD5해시를 이용하는 방법과 주제1에서 살펴본 악성코드 진단 문자열을 이용하는 방법이다. 진단 문자열을 이용하는 방법은 다시 2가지로 구분되는데, 파일을 열고 처음부터 끝까지 체크하여 진단 문자열이 있는지 확인하는 방법(전체위치 검색법)과 특정 위치에 진단 문자열이 존재하는지 확인하는 방법(특정위치 검색법)이다. 전체위치 검색법은 속도는 느리지만 악성코드변형을 진단하는데 탁월하다. 특정위치 검색법은 변형을 진단 하는데는 힘들지만 검사속도면에서 빠르다. [리스트7-5]와 [리스트7-6]은 특정위치 검색법을 이용한 악성코드 검사방법이다 (scan_str.py). [리스트7-7]은 Scanstr함수를 기존 scanmod.py에 추가한 파이썬 코드이다.

- 악성코드 진단 모듈과 마찬가지로 치료 모듈도 별도의 파일로 분리할 수 있다. 치료 모듈은 curemod.py라는 파일로 저장한다. 여기서 치료방법은 악성코드에 감염된 파일을 삭제하는 것이므로 삭제함수인 CureDelete함수를 작성하면 된다. [리스트7-8]과 [리스트7-9]는 악성코드 치료모듈을 파이썬 모듈로 분리하여 사용하는 방법을 보여준다.
- 이제 진단 모듈과 치료모듈이 분리되었지만 악성코드 패턴파일인 virus.db에는 아직 MD5해시를 이용하는 방법만 있으므로 [리스트7-10]은 MD5해쉬 이외에 특정위치 검색법이 추가된 virus.db를 보여준다. antivirus.py는 예전의 virus.db구조만 인식하기 때문에 마지막으로 antivirus.py를 수정해야 한다. [리스트7-12]와 [리스트7-13]에 MD5해쉬와 특정위치 검색법을 사용한 악성코드 검사를 위한 부분이 추가 및 수정되었다. Antivirus.py의 메인 부분이 [리스트7-13]과 같이 수정되고 scanmod.py의 ScanMD5를 ScanVirus함수로 바꾸고 MD해시 뿐만 아니라 특정위치 검색법을 이용하여 검사를 진행하도록 한다.

➤ [과제 내용]

- 아래의 각 문제에 대한 파이썬 코드의 수행 결과를 캡처 하여 분석하고 설명하는 내용을 개인적인 견해와 함께 레포트에 모두 반영하여야 한다.
- 1. SearchVDB와 ScanMD5를 추출하고 scanmod로 모듈화한 백신 프로그램 (antivirus.py) 인 [리스트 7-4]를 실행하여 보아라. 결과는 [그림7-1]과 같아야 한다.
- 2. [리스트7-5]의 특정위치 검색법(scan_str.py)을 사용하여 악성코드를 검사하여 보아라. 결과는 [그림7-2]와 같아야 한다.
- 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 악성코드는md5 해쉬를 이용해서 검사했고 EICAR Test 악성코드는 특정위치 검색법을 이용하여 검사했는지 확인하여라.

➤ 주 제 5 : 전용백신 배포본 만들기

- 지금까지 살펴본 전용백신을 **배포본**으로 만들어보자. 물론 상업적인 백신처럼 보기 좋은 GUI는 없다. 파이썬으로 백신을 만들었으니 배포본을 사용하는 사용자의 PC에도 파이썬이 설치되어 있어야 할 것 같다. 하지만 파이썬으로 작성된 소스코드(*.py)를 윈도우 **실행 파일(*.exe)로 변환하는 도구(py2exe, PyInstaller)를 이용**하면 백신을 하나의 실행파일로 만들 수 있다. 여기서는 PyInstaller를 사용하기로 한다. Py2exe를 사용하면 실행파일 이외에 부수적인 파일이 많아서 불편하다.
- 매번 파이썬으로 된 백신 소스코드를 수정하고 빌드(PyInstaller를 실행하여 배포본을 만드는 과정)하는 것을 간편하게 만들기 위해서 참고자료의 8장을 토대로 **build.bat**를 만들고 실행한다. 생성된 antivirus.exe를 실행하기 위해서는 패턴파일이 필요하기 때문에 virus.kmd파일을 antivirus.exe가 존재하는 폴더로 복사한다. antivirus.exe백신 실행시에 발생하는 exit오류를 없애기 위하여 antivirus.py의 sys.exit함수를 정확히 기술한다. eicar.txt를 antivirus.exe가 있는 폴더에 복사하고 검사하여 악성코드를 진단한다. 결과가 제대로 나오는지 확인한다.
- 실제로 주제4에서 antivirus.py로부터 악성코드 진단 및 치료 모듈을 분리하였지만 PyInstaller가 윈도우 실행파일로 변환하는 과정에서 import된 모듈을 **모두 윈도우 실행 파일(antivirus.exe) 안에 내장시켜 버린다.** 따라서 겉으로 canmod.py와 curemod.py가 보이지 않아도 악성코드를 진단하고 치료하는데 문제가 없다. 그럼 주제4에서 의도했던 대로 진단 및 치료 모듈을 외부에 놓아 둘수는 없을까? 이는 파이썬 모듈을 동적으로 import함으로써 가능하다.

- 즉, antivirus.exe에는 이미 PyInstaller에 의해 scanmod 모듈(정확하게는 파이썬 소스코드가 컴파일된 scanmod.py)과 curemod모듈이 포함되어 있다. 따라서 antivirus.exe는 scanmod와 curemod모듈 없이 단독으로 실행이 가능하지만, 같은 폴더내의 외부에 scanmod모듈이 존재한다면 scanmod모듈을 동적으로 임포트한다.

➤ [과제 내용]

1. 아래의 각 문제에 대한 파이썬 코드의 수행 결과를 캡처 하여 분석하고 설명하는 내용을 개인적인 견해와 함께 레포트에 모두 반영하여야 한다.
2. **PyInstaller**를 사용하여 주제4에서 작성한 파이썬으로 된 백신을 윈도우 실행파일로 변환한다. [그림8-2] 및 [그림8-3]과 같은 결과를 보이는지 확인한다.
3. 변환된 백신의 윈도우 실행파일을 테스트해본다. exit오류가 발생하면 8.3.1절의 내용에 따라 수정한다. [그림8-6]과 같은지 확인한다.
4. [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 배포하는 방법도 테스트해 본다.

➤ Windows 바이러스 및 위협 방지

- 윈도우키 + R (실행) → gpedit.msc → 컴퓨터 구성 → 관리 템플릿 → Windows 구성 요소 → Microsoft Defender 바이러스 백신 → Microsoft Defender 바이러스 백신 끄기 → 사용(E) → 확인
- gpedit.msc 명령어가 불가할 경우, windows home 에디션 때문이며 다음 링크에서 받은 파일을 관리자 권한으로 실행 -> 재부팅 하여 재시도
- https://dguackr-my.sharepoint.com/:u:/g/personal/code_dgu_ac_kr/EV3pFXgKa_NGtPI5ik6z-IsBniqAUPHUp8ppfceleZMVdw?e=JR9C07
- 해당 컴퓨터 보안 실습을 모두 진행한 후에는 다시 “구성되지 않음(C)”으로 변경하고 윈도우 보안 유지 권장

➤ python 2.7.x 버전 설치 방법 (in Linux)

- mkdir ~/temp
- cd ~/temp
- wget https://www.python.org/ftp/python/2.7.14/Python-2.7.14.tgz
- tar -xzf Python-2.7.14.tgz
- cd Python-2.7.14
- ./configure --enable-optimizations
- sudo make altinstall
- alias python=python2.7
- python

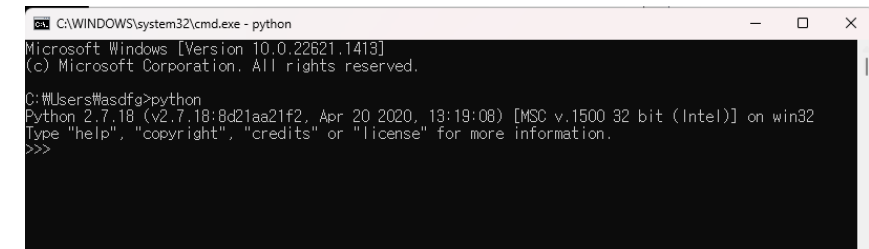
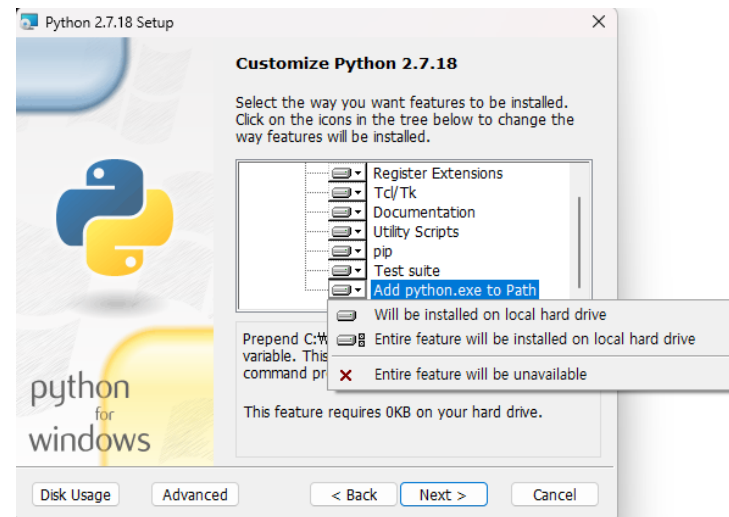
➤ python 2.7.x 버전 설치 방법 (in windows)

- <https://www.python.org/downloads/release/python-2718/>

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		38c84292658ed4456157195f1c9bcbe1	17539408	SIG
XZ compressed source tarball	Source release		fd6cc8ec0a78c44036f825e739f36e5a	12854736	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	ce98eeb7bdf806685adc265ec1444463	24889285	SIG
Windows debug information files	Windows		20b111ccfe8d06d2fe8c77679a86113d	25178278	SIG
Windows debug information files for 64-bit binaries	Windows		bb0897ea20fda343e5179d413d4a4a7c	26005670	SIG
Windows help file	Windows		b3b753dffe1c7930243c1c40ec3a72b1	6322188	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64	a425c758d38f8e28b56f4724b499239a	20598784	SIG
Windows x86 MSI installer	Windows		db6ad9195b3086c6b4cefb9493d738d2	19632128	SIG

- 다음 그림에서 “Entire feature will be installed on local hard drive” 로 설정 (나머지는 모두 단순 Next)









설치 완료

➤ pyinstaller 3.2 버전 설치 방법 (in windows)

- <https://github.com/pyinstaller/pyinstaller/releases/tag/v3.2>

▼ Assets 6

 PyInstaller-3.2.tar.gz	2.65 MB	May 4, 2016
 PyInstaller-3.2.tar.gz.asc	801 Bytes	May 4, 2016
 PyInstaller-3.2.zip	2.94 MB	May 4, 2016
 PyInstaller-3.2.zip.asc	801 Bytes	May 4, 2016
 Source code (zip)		May 4, 2016
 Source code (tar.gz)		May 4, 2016

- 다운 받은 파일을 Python27이 설치된 경로로 압축해제

📁 > 내 PC > Windows (C:) > Python27 >

이름	수정한 날짜	유형	크기
DLLs	2023-04-12 오전 3:38	파일 폴더	
Doc	2023-04-12 오전 3:38	파일 폴더	
include	2023-04-12 오전 3:38	파일 폴더	
Lib	2023-04-12 오전 4:32	파일 폴더	
libs	2023-04-12 오전 3:38	파일 폴더	
PyInstaller-3.2	2023-04-12 오전 4:32	파일 폴더	
Scripts	2023-04-12 오전 4:32	파일 폴더	

➤ pyinstaller 3.2에서 오류가 난다면? (No module pefile)

- pip install pefile 명령어 입력하여 모듈 설치

```
Requirement already satisfied: pywin32 in c:\python27\lib\site-packages (228)
PS C:\Users\asdfg\Desktop\test> python C:\Python27\PyInstaller-3.2\pyinstaller.py -F .\antivirus.py
Traceback (most recent call last):
  File "C:\Python27\PyInstaller-3.2\pyinstaller.py", line 14, in <module>
    from PyInstaller.__main__ import run
  File "C:\Python27\PyInstaller-3.2\PyInstaller\__main__.py", line 21, in <module>
    import PyInstaller.building.build_main
  File "C:\Python27\PyInstaller-3.2\PyInstaller\building\build_main.py", line 34, in <module>
    from .api import PYZ, EXE, COLLECT, MERGE
  File "C:\Python27\PyInstaller-3.2\PyInstaller\building\api.py", line 38, in <module>
    from PyInstaller.utils.win32 import winmanifest, icon, versioninfo, winresource
  File "C:\Python27\PyInstaller-3.2\PyInstaller\utils\win32\versioninfo.py", line 17, in <module>
    import pefile
ImportError: No module named pefile
```

```
PS C:\Users\asdfg\Desktop\test> pip install pefile
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade to Python 3.10 or above.
Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/#python-version-support.
Collecting pefile
  Downloading pefile-2019.4.18.tar.gz (62 kB)
    |#####| 62 kB 476 kB/s
```

➤ pyinstaller 3.2에서 오류가 난다면? (No module pefile)

- pip install pypiwin32 명령어 입력하여 모듈 설치

```
PS C:\Python27\Scripts> pip install pypiwin32
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade to Python 3.10 or above.
Python 2.7 in January 2021. More details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/#python-version-support.
Collecting pypiwin32
  Downloading pypiwin32-223.tar.gz (622 bytes)
Collecting pywin32>=223
  Downloading pywin32-228-cp27-cp27m-win32.whl (6.9 MB)
    |#####| 6.9 MB 7.3 MB/s
Using legacy 'setup.py install' for pypiwin32, since package 'wheel' is not installed.
Installing collected packages: pywin32, pypiwin32
  Running setup.py install for pypiwin32 ... done
Successfully installed pypiwin32-223 pywin32-228
PS C:\Python27\Scripts> cd C:\Users\asdfg\Desktop\test
```

➤ 실습 진행 방법

- 간단한 이론 복습 및 해당주차 실습문제 설명
- 실습 후 보고서를 작성하여 2주뒤 화요일 자정 전(23:59) 까지 E-Class에 제출(이메일 제출 불가, 반드시 E-Class를 통해 제출)
- 실습 과제 제출 기한 엄수 (제출기한 이후로는 0점 처리)

➤ 실습 보고서 [1/2]

- 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술
- 프로그램 설계 / 알고리즘
 - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
 - 문제 해결 과정 및 핵심 알고리즘 기술
- 결과 / 결과 분석
 - 결과 화면을 캡처 하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
- 소감
 - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

➤ 실습 보고서 [2/2]

- 제출 방법

- 보고서를 작성하여 E-Class “과제” 메뉴를 통해 제출
 - “이름_학번_실습주차.zip” 형태로 제출 (e.g. : 홍길동_2022123456_실습1주차.zip)
 - 파일명에 공백, 특수 문자 등 사용 금지
 - 보고서 파일은 한글/워드 등 모두 상관 없지만 PDF 변환 후 이클래스 제출 (압축은 필요한 경우에만 사용)

- 유의사항

- 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
- 정해진 기한 내 제출
 - 기한을 넘길 시 0점 처리
 - E-Class가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일 전에 제출
 - 당일 E-Class 오류로 인한 미제출은 불인정
- 보고서를 자신이 작성하지 않은 경우 실습 전체 점수 0점 처리

Q & A
