

졸업자격실험보고서

C라이브러리 확장을 이용한 미니백신  
프로그램

A Mini Vaccine Program Using C-Libraries  
Extending

지도교수 오 승 현

동국대학교 과학기술대학 컴퓨터공학과

최 수 호

2 0 2 0

졸업자격실험보고서

C라이브러리 확장을 이용한 미니백신  
프로그램

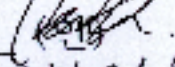
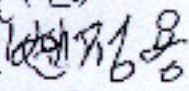
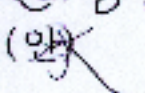
A Mini Vaccine Program Using C-Libraries  
Extending

최 수 호

지도교수 오 승 현

본 보고서를 졸업자격 실험보고서로 제출함.  
2020년 10월 13일

최수호의 졸업자격 실험보고 통과를 인준함.  
2020년 10월 14일

주	심	박	기	석	
부	심	변	정	용	
부	심	오	승	현	(인) 

동국대학교 과학기술대학 컴퓨터공학과

# 목 차

1. 서론 .....	1
1.1 실험 목적 .....	1
1.2 실험 범위 .....	1
 2. 이론적 배경 .....	2
2.1 Python 언어 .....	2
2.2 C 언어 .....	2
2.3 Python Extending을 통한 C언어 모듈화 .....	2
2.4 MD5 해시 값을 활용한 파일 무결성 .....	3
2.5 PyQt5를 사용한 GUI 구현 .....	4
 3. 시스템 설계 .....	6
3.1 기본 설계 .....	6
3.2 상세 설계 .....	8
3.2.[1] C라이브러리 - 검사 모듈 설계 .....	8
3.2.[2] C라이브러리 - MD5 해시 모듈 설계 .....	9
3.2.[3] GUI 설계 .....	10
 4. 프로그램 구현 .....	12
4.1 개발 환경 및 필요한 도구 .....	12
4.2 메인화면 .....	12

4.3 검사시작 .....	13
4.4 검사결과 .....	16
 5. 실험 .....	 18
5.1 테스트 파일을 이용한 검사 .....	18
5.2 실제 바이러스 파일을 이용한 검사 .....	19
 6. 결론 .....	 21
6.1 결론 .....	21
6.2 향후 과제 .....	21
 참고문헌 .....	 22
 부록	
소스코드	
 요약문	

## 그 립 목 차

[그림 2.3 - (1)] 오브젝트 파일 생성 .....	3
[그림 2.3 - (2)] 동적 라이브러리 파일 생성 .....	3
[그림 2.4 - (1)] MD5 알고리즘 .....	3
[그림 2.5 - (1)] PyQt .....	4
[그림 2.5 - (2)] PyQt5 기본 예제 코드 .....	4
[그림 2.5 - (3)] Qt Designer .....	5
[그림 3.1 - (1)] GUI 설계 .....	6
[그림 3.1 - (2)] 프로그램 Activity Diagram .....	7
[그림 3.2 - (1)] 검사모듈 Flow Chart .....	8
[그림 3.2 - (2)] MD5 해시 Flow Chart .....	9
[그림 3.2 - (3)] GUI Class Diagram .....	10
[그림 4.2 - (1)] 프로그램 메인화면 .....	12
[그림 4.3 - (1)] 프로그램 경로 선택 .....	13
[그림 4.3 - (2)] 프로그램 검사완료 1 .....	14
[그림 4.3 - (3)] 프로그램 검사완료 2 .....	14
[그림 4.3 - (4)] 프로그램 검사완료 3 .....	15
[그림 4.4 - (1)] 프로그램 검사결과 .....	16
[그림 5.1 - (1)] EICAR 테스트 파일 다운로드 .....	18
[그림 5.1 - (2)] EICAR 테스트 파일 분석 .....	18
[그림 5.1 - (3)] EICAR 테스트 파일 검사 .....	19
[그림 5.2 - (1)] 바이러스 파일 내용과 MD5 해시 값 .....	19

## 표 목 차

[표 4.2] 메인화면 주요 코드 .....	13
[표 4.3] 검사시작 주요 코드 .....	16
[표 4.4] 검사결과 주요 코드 .....	17

# 1. 서 론

## 1.1 실험 목적

최근 <sup>1)</sup>Python의 인기가 상당히 올라가고 있다. 문법이 쉬워 배우기도 쉽고 지원되는 API도 많아서 효율적으로 사용할 수 있다. 하지만 처리 속도가 느리다는 단점이 있다. 물론 컴퓨터의 성능이 좋아짐에 따라 크게 차이가 나진 않지만 기계학습, 딥러닝과 같은 고정적인 연산을 많이 하는 프로그램들은 Python과 다른 언어를 함께 사용한다. 이렇게 다른 언어를 Python에서 사용할 수 있게 확장하는 것을 <sup>2)</sup>Python Extending이라고 한다. Python Extending을 사용해서 간단한 프로그램을 만들어 Python의 확장성을 기대하며 단점을 보완할 수 있다고 생각한다. 또 본 연구에서는 GUI부분을 기존 Python의 GUI 프레임워크인 <sup>3)</sup>Tkinter와는 다른 프레임워크를 사용하여 시각적인 부분을 높였다.

## 1.2 실험 범위

본 연구의 주요 언어는 Python과 C언어이며, GUI부분은 Python과 C++의 Qt가 결합된 PyQt5를 사용했다. 사용된 툴은 IDLE, PyQt5 Designer, Visual Studio 2019이다. 구현한 프로그램은 백신의 아키텍처를 이용한 미니백신이다. C언어로 프로그램의 기능적인 부분들을 구현하고, 컴파일러를 이용해 C언어 라이브러리 파일로 변환하여 Python에서 사용할 수 있게 만들었다. 프로그램은 파일의 MD5 해시 값을 읽어 악성코드의 MD5 해시 값과 비교하게 된다. GUI에는 트리뷰 파일 탐색기를 구현하여 쉽게 검사할 디렉토리를 선택할 수 있게 하였고, 검사를 시작하면 C언어의 라이브러리를 읽어와 검사가 진행된다. 그렇게 검사된 결과를 Python에서 읽어와 출력해주고 검사결과 창에서 파일의 전체삭제와 선택삭제를 할 수 있게 설계되었다. 악성코드 파일은 상용화된 백신에서도 악성코드로 탐지되는 테스트 파일을 이용했다.

## 2. 이론적 배경

### 2.1 Python 언어

Python은 범용 프로그래밍 언어로 코드 가독성이 뛰어나고 문법이 간결하다. 또, 인터프리터 언어로 Windows, Mac OS, Linux등 여러 시스템에서 널리 사용된다. 여러 가지 라이브러리가 풍부하며, 다른 모듈로 쓰인 모듈을 연결하는 폴 언어로써 자주 이용된다. 하지만 C/C++ 언어에 비해 수행 속도가 느리다는 단점이 있다. 이 부분을 보완하기 위해서 Embedding과 Extending을 통해 Python을 다른 언어에서 사용할 수 있도록 이식하거나 필요한 부분만 골라서 C언어 등으로 모듈화가 가능하다.

### 2.2 C 언어

4)C언어는 사용자 중심의 고급 언어이지만 동시에 하드웨어를 직접 제어할 수 있는 저급언어의 특징을 가진 프로그래밍 언어이다. 모든 C 시스템에는 정규화된 표준 C 라이브러리가 존재하며 매우 효율적이며 수행 속도가 빠르다. 그리고 이식성이 좋아 여러 운영 체제에서 같은 코드로 사용이 가능하다. 이처럼 C는 시스템 프로그램 개발에 매우 적합하고 응용 프로그램 개발에도 많이 쓰인다.

### 2.3 Python Extending을 통한 C언어 모듈화

Python Extending이란 다른 언어를 Python에서 사용할 수 있도록 확장하는 것이다. 이 연구에서는 C언어를 동적 라이브러리 파일로 만들어 Python에 확장시킨다. C언어를 Python에서 확장시키려면 Cython이나 5)Ctypes을 이용한다. GCC컴파일러에 fPIC(Position-Independent Code) 옵션을 주어 C 코드 파일을 동적 라이브러리로 사용할 수 있도록 컴파일한다. 그리고 오브젝트파일(확장자 : .o)을 동적 라이브러리파일(확장자 : .so)로 묶어서 Ctypes를 이용해 Python에서 사용할 수 있게 만든다.



```

관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Windows\system32>gcc -fPIC -c code.c

```

<그림 2.3 - (1) : 오브젝트 파일 생성>

```

관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

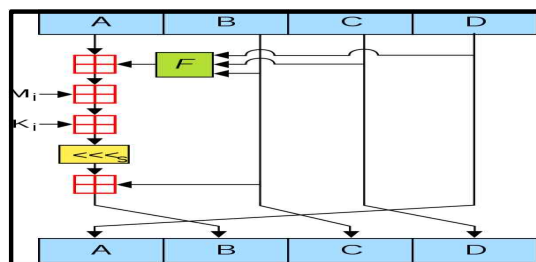
D:\Windows\system32>gcc -shared -o name.so code.o_

```

<그림 2.3 - (2) : 동적 라이브러리파일 생성>

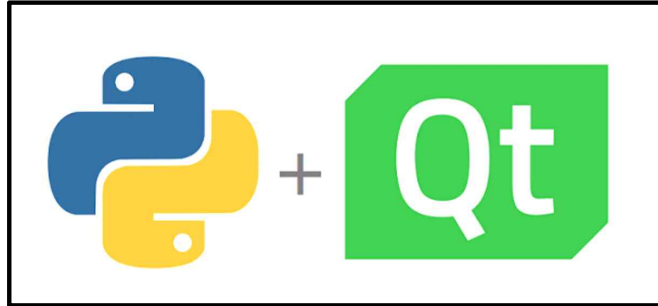
## 2.4 MD5 해시 값을 활용한 파일의 무결성

단방향 해시 알고리즘의 일종인 MD5(Message-Digest algorithm 5)는 128비트 암호화 해시 알고리즘이다. MD5는 임의의 길이의 메시지를 입력받아 128비트 고정 길이의 출력 값을 낸다. 입력 메시지는 패딩하여 한 블록마다 4개의 단계를 거쳐 변환된다. 그래서 주로 프로그램이나 파일이 원본 그대로인지 확인하는 무결성 검사 등에 쓰인다. MD5 해시를 이용하여 진단할 수 있는 악성코드는 바이러스 유형을 제외한 파일 그 자체가 악성코드인 트로이목마, 백도어, 웜 등이 있다.



<그림 2.4 - (1) : MD5 알고리즘>

## 2.5 PyQt5를 사용한 GUI 구현

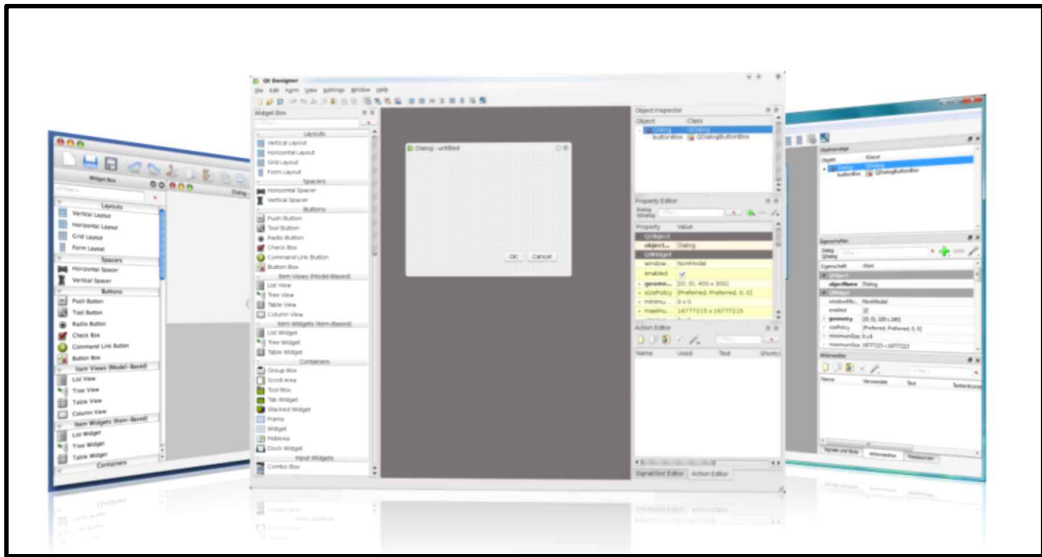


<그림 2.5 - (1) : PyQt>

7)PyQt는 Python과 C++의 Qt를 결합한 프레임워크이다. PyQt는 Qt가 제공하는 GUI, 네트워킹, 스레드, 정규 표현식, SQL 데이터베이스, SVG, OpenGL 등 약 440개의 클래스와 6000개 이상의 함수 및 메소드를 제공한다. 또한 PyQt는 Windows, Linux, UNIX, Android, Mac OS 및 iOS를 지원한다.그리고 8)Qt Designer를 통해 손쉽게 GUI를 구성하고 변경할 수 있다.

```
1 #! /usr/bin/env python3
2 # Character Encoding: UTF-8
3 #
4 # Here we provide the necessary imports.
5 # The basic GUI widgets are located in QtWidgets module.
6 import sys
7 from PyQt5.QtWidgets import QApplication, QWidget
8
9 # Every PyQt5 application must create an application object.
10 # The application object is located in the QtWidgets module.
11 app = QApplication(sys.argv)
12
13 # The QWidget widget is the base class of all user interface objects in PyQt5.
14 # We provide the default constructor for QWidget. The default constructor has no parent.
15 # A widget with no parent is called a window.
16 root = QWidget()
17
18 root.resize(320, 240) # The resize() method resizes the widget.
19 root.setWindowTitle("Hello, World!") # Here we set the title for our window.
20 root.show() # The show() method displays the widget on the screen.
21
22 sys.exit(app.exec_()) # Finally, we enter the mainloop of the application.
```

<그림 2.5 - (2) : PyQt5 기본 예제 코드>



<그림 2.5 - (3) : Qt Designer>

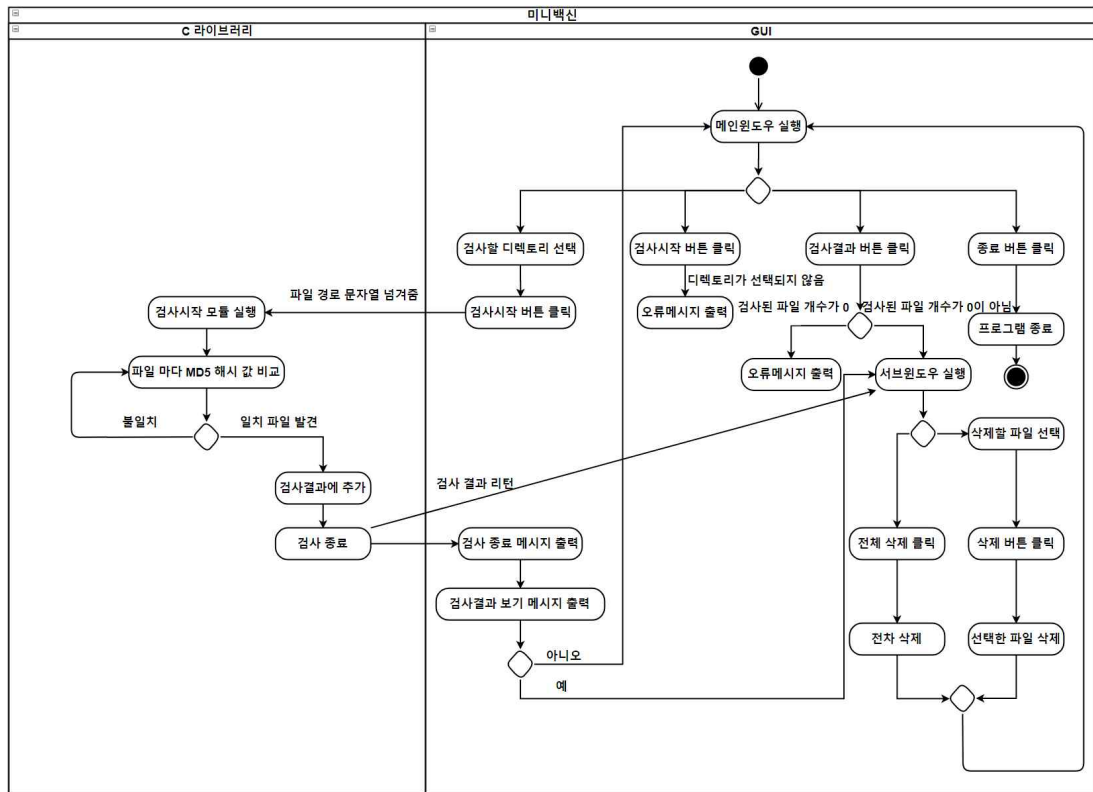
### 3. 시스템 설계

#### 3.1 기본 설계



<그림 3.1 - (1) : GUI 설계>

<그림 3.1 - (1)>은 전체적인 GUI 부분을 설계할 때 목업이다. 프로그램 제목이 그림과 함께 왼쪽 상단에 위치하고 그 밑에 검사하기 버튼, 검사결과 확인 버튼, 종료 버튼이 위치한다. 그 오른쪽은 검사할 파일을 쉽게 찾고 선택할 수 있도록 파일 탐색기가 들어간다. 그리고 그 위에는 검사가 어느 정도 진행되었는지 알 수 있는 Progress Bar와 어느 위치를 검사하고 있는지 나타내는 파일 경로가 위치한다.



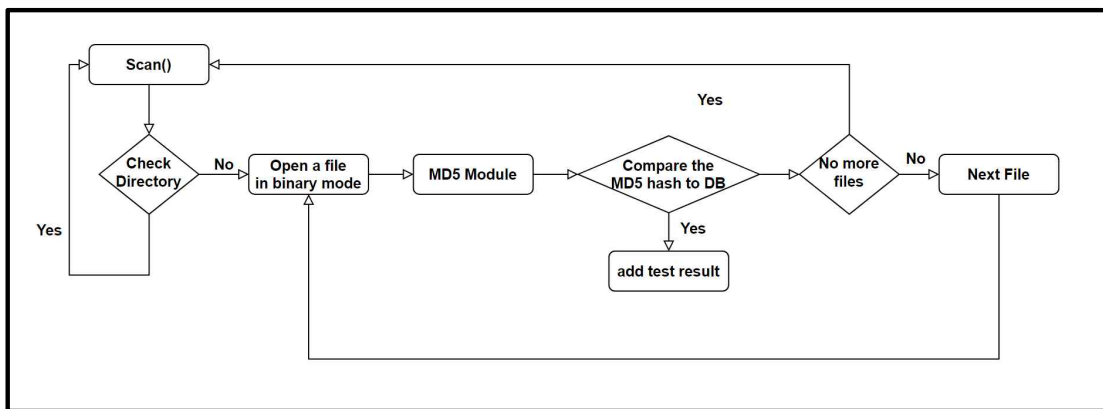
<그림 3.1 - (2) : 프로그램 Activity Diagram>

<그림 3.1 - (2)>는 프로그램의 전체적인 동작 과정을 나타내는 Activity Diagram이다. 먼저 프로그램이 실행되면 GUI 메인윈도우가 실행되며, 여기서 디렉토리 선택, 검사시작 버튼 클릭, 검사결과 버튼 클릭, 종료 버튼 클릭을 할 수 있다. 검사할 디렉토리가 선택이 되지 않았을 때, 검사시작 버튼을 클릭하면 오류메시지를 출력하고, 검사결과 버튼을 클릭했을 때, 검사된 파일의 개수가 0 개이면 오류메시지를 출력한다. 그래서 검사할 디렉토리를 선택한 뒤 검사시작 버튼을 누르면 그 디렉토리 경로를 C라이브러리 모듈에 넘겨준다. 그렇게 검사 시작 모듈이 시작되고 파일의 MD5 해시 값과 DB에 있는 해시 값을 비교한다. 일치하는 경우 검사결과 값에 추가가 되고 불일치하면 다음 파일로 넘어가 검사한다. 그렇게 디렉토리 안에 있는 모든 파일이 검사가 끝나면 검사결과 값을 리턴하며 검사종료 메시지와 함께 검사결과 확인을 위한 메시지가 나온다. ‘예’를 누를 경우 검사결과 확인을 위해 GUI 서브윈도우가 실행되며 리턴받은 검사결과가 출력된다. ‘아니오’를 누를 경우 기본 메인윈도우가 화면에 보이게 된다. 그

렇게 GUI 서브윈도우가 실행되면 파일을 선택해 삭제할 수 있는 선택삭제 버튼과 전체삭제 버튼이 위치하게 된다. 파일을 선택하지 않고 선택삭제 버튼을 클릭하면 오류메시지를 출력하고 파일을 선택한 뒤, 선택삭제 버튼을 클릭하면 선택한 파일이 삭제된다. 전체삭제 버튼을 클릭하면 검사결과에 나와있는 모든 파일이 삭제된다. 그리고 검사결과 창을 닫으면 다시 GUI 메인윈도우가 보이며 종료 버튼을 누를시 프로그램이 종료하게 된다.

## 3.2 상세 설계

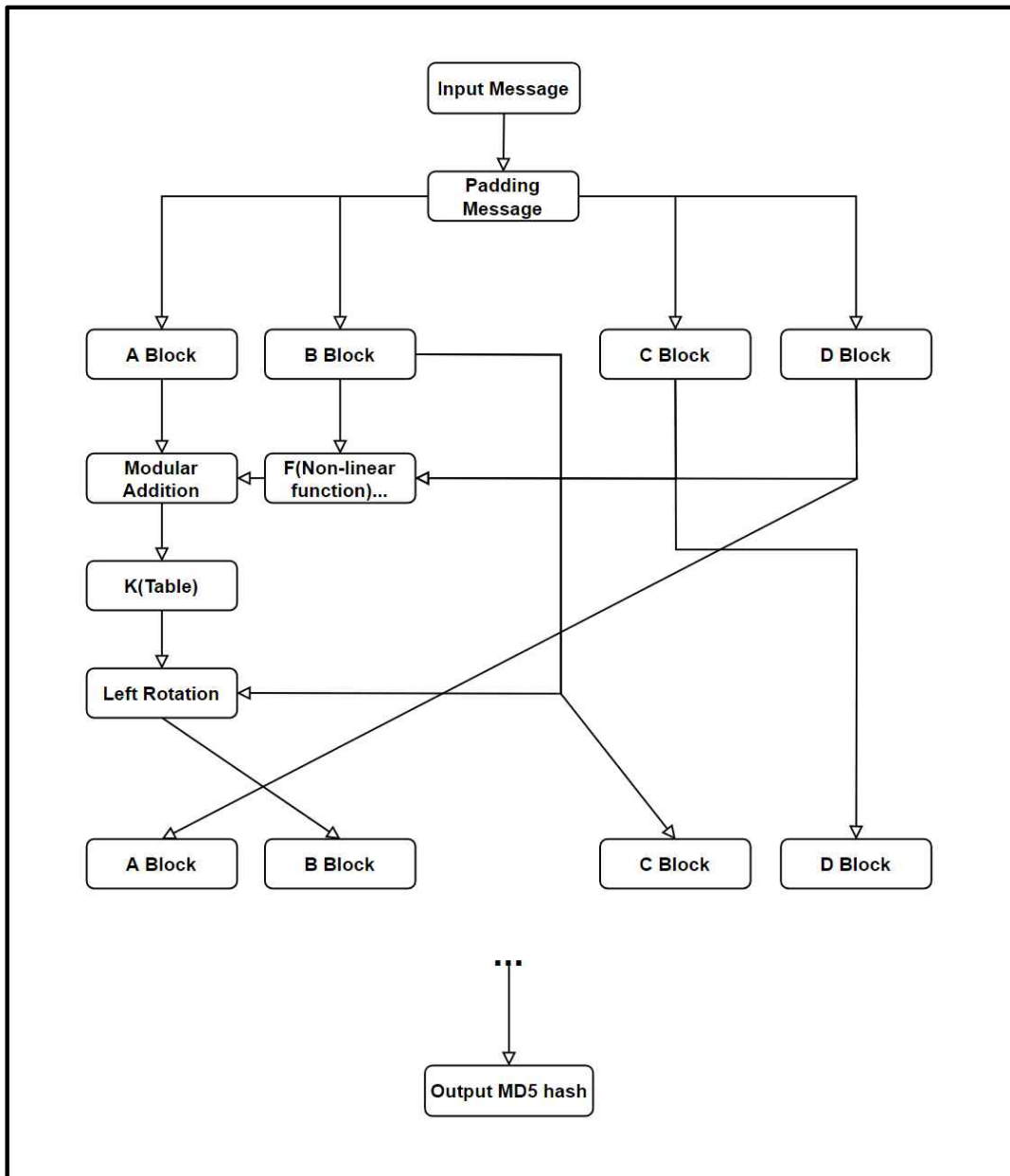
### 3.2.[1] C라이브러리 - 검사 모듈 설계



<그림 3.2 - (1) : 검사모듈 Flow Chart>

<그림 3.2 - (1)>은 C언어로 작성될 검사모듈인 C 라이브러리에 대한 Flow Chart이다. 먼저 검사가 시작되면 입력받은 디렉토리에서 디렉토리인지 파일인지 검사를 하게 된다. 파일일 경우, 그 파일의 내용을 ‘바이너리 읽기’모드로 열어 MD5 해시 모듈을 통해 MD5 해시 값으로 바꿔 DB에 있는 MD5 해시 값과 비교하게 된다. DB에 있는 해시 값과 동일하면 검사결과에 추가 되고 다를 경우 다음 파일로 넘어가 다시 파일을 ‘바이너리 읽기’모드로 열어 비교하게 된다. 다음 파일이 더 없을 경우 모듈이 끝나게 된다. 그리고 디렉토리일 경우, 경로에 그 디렉토리가 추가된 경로로 재귀함수를 실행시켜 다시 경로를 입력받아 실행하게 된다.

### 3.2.[2] C라이브러리 - MD5 해시 모듈 설계

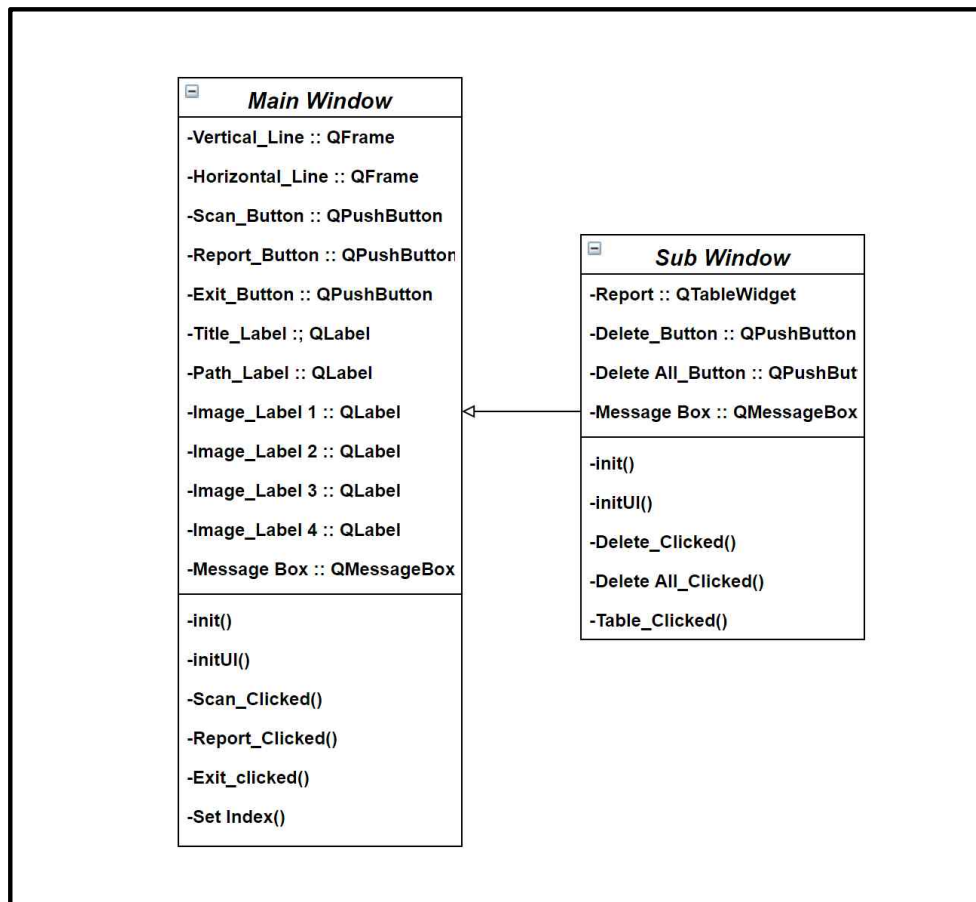


<그림 3.2 - (2) : MD5 해시 Flow Chart>

<그림 3.2 - (2)>는 입력된 파일의 내용을 MD5 해시 값으로 바꿔주는 모듈의 한 라운드의 Flow Chart이다. 처음에 입력받은 내용을 512bit의 블록들로 패딩을 이용하여 쪼개지게 만든다. 그리고 128bit 블록을 32bit 블록(A, B, C, D)로 초기화한다. 32bit 블록 하나당 총 4개의 라운드를 거치며 라운드는 비선형함수, 모듈러 덧셈, 레프트 로테이션으로 이루어지며 비선형함수는 F, G, H, I로 이루어져

있다. 그렇게 해서 128bit블록 하나 당 16번의 라운드를 거치게 되며 총 64번의 암호화를 하면 해시 값을 얻을 수 있다.

### 3.2.[3] GUI 설계



<그림 3.2 - (3) : GUI Class Diagram>

<그림 3.2 - (3)>은 GUI의 Class Diagram이다. 기본설계 부분의 GUI를 이용하여 설계를 하였다. 처음 Main Window에는 제목을 입력할 Title\_Label, 경로를 보여줄 Path\_Label, 프로그램의 이미지를 띄워줄 Image\_Label 1, 버튼의 이미지를 띄워줄 Image\_Label 2, 3, 4가 있다. Label은 PyQt5의 QLabel 위젯을 사용한다. 또, 공간을 나뉘줄 수직, 수평라인 Vertical\_Line과 Horizontal\_Line, 검사시작 버튼, 검사결과 버튼, 종료 버튼인 Scan\_Button, Report\_Button, Exit\_Button이 있다. 마찬가지로 Line과 Button은 PyQt5의 QFrame, QPushButton 위젯을 사용한다. 파일 탐색기는 TreeView 탐색기를 이용. 마지막으로 사용자에게 메시지를



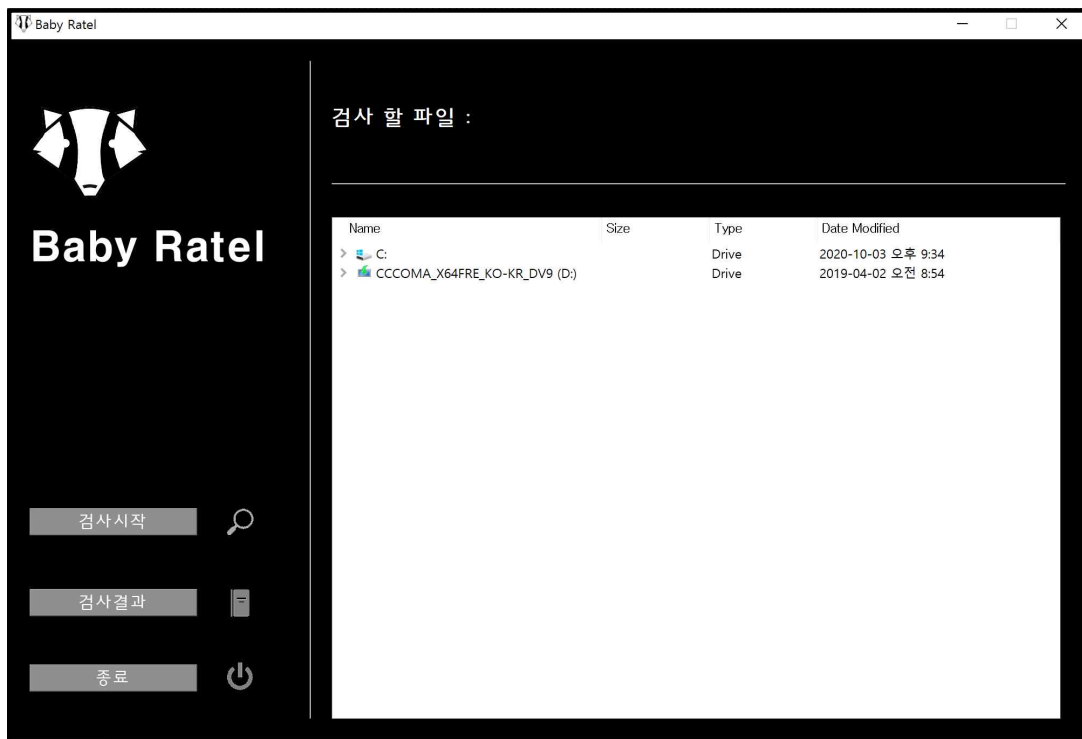
띄워줄 Message Box가 포함되어 있다. 모듈로는 Label, Button 등을 정의하는 init()모듈, 그리고 설정하는 initUI()모듈이 있고, 버튼을 클릭했을 때의 이벤트를 정의하는 모듈 3개와 파일 탐색기를 클릭했을 때 Label부분에 경로를 보이게 해주는 Set\_Index()모듈이 있다. Sub Window는 검사결과 버튼을 클릭했을 때 보여지고 검사결과를 띄워주는 Report와 선택삭제 버튼, 전체삭제 버튼이 있다. Report는 PyQt5의 QTableWidgetItem를 이용하였다. 마찬가지로 사용자에게 메시지를 띄워주는 Message Box가 있다. 모듈은 정의부분의 init(), 설정부분의 initUI(), 선택삭제를 할 때 Table을 클릭했을 때 값을 받아오는 Table\_Clicked() 모듈과 선택삭제 버튼, 전체삭제 버튼을 클릭했을 때 이벤트인 Delete\_Clicked(), Delete All\_Clicked()가 있다.

## 4. 프로그램 구현

### 4.1 개발 환경 및 필요한 도구

개발은 64비트 Windows 10에서 진행하였다. 사용된 툴은 C언어 코딩을 위한 Visual Studio 2019를 설치하고 Python, 그리고 pip을 이용하여 PyQt5와 PyQt5 Designer를 설치하였다.

### 4.2 메인 화면



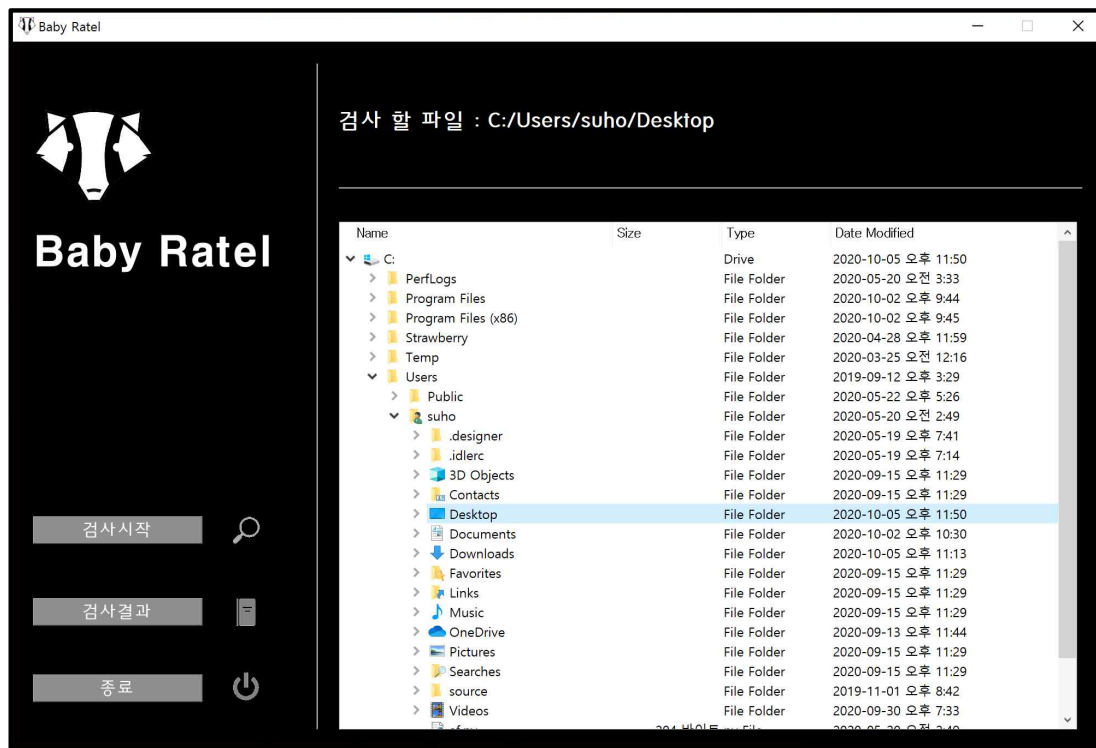
<그림 4.2 - (1) : 프로그램 메인 화면>

<그림 4.2 - (1)>은 프로그램의 메인 화면이다. 좌측상단에 프로그램의 이름과 이미지가 있으며 밑에는 검사시작 버튼, 검사결과 버튼, 종료 버튼이 있다. 버튼 옆으로는 시각적으로 알아보기 쉽게 이미지를 넣었다. 그리고 우측에는 트리뷰 파일탐색기가 있고 상단에는 검사 할 파일 경로가 찍히는 라벨이 들어가 있다. <표 4.2 - (1)>은 에서 사용된 주요 코드 구성이다.

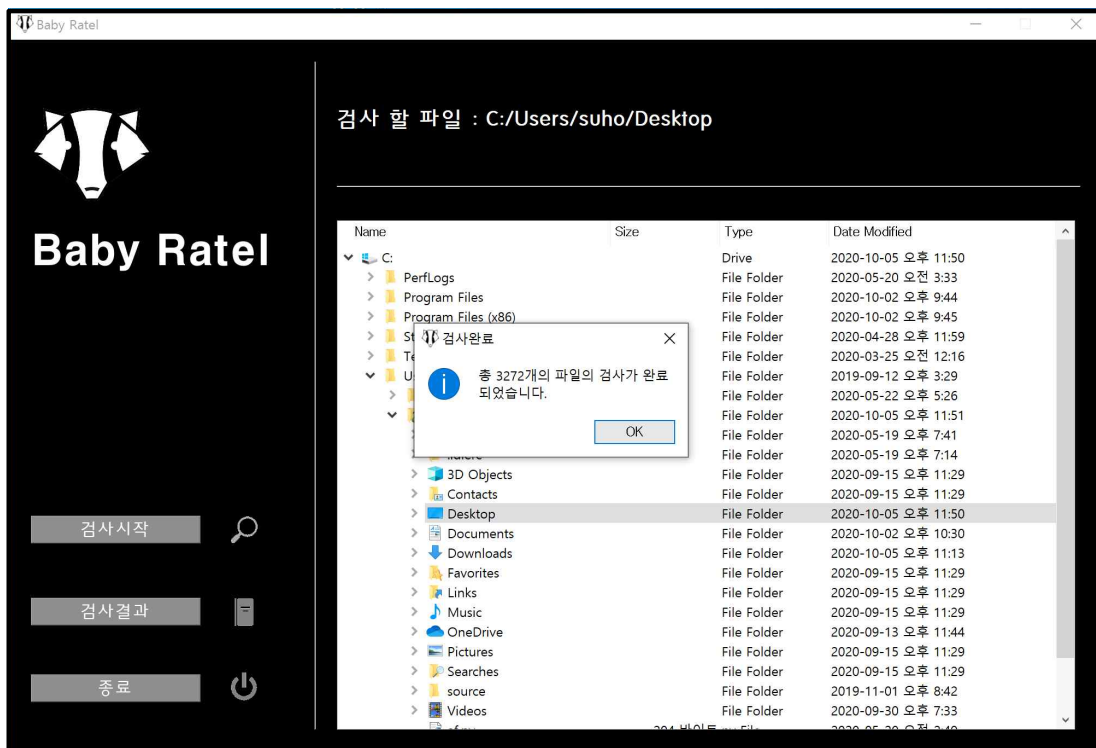
<표 4.2 - (1) : 메인 화면 주요 코드>

MainWindow :: ctypes.CDLL(“라이브러리 파일명”)	C 라이브러리를 Python에서 사용할 수 있게 한다.
MainWindow :: QPushButton()	클릭 가능한 버튼을 생성한다.
MainWindow :: QTreeView()	트리뷰 아이টে를 생성한다.
MainWindow :: QFileSystemModel()	트리뷰의 모델을 설정한다.
MainWindow :: QLabel()	라벨을 생성한다.
MainWindow :: QMessageBox()	메시지 박스를 생성한다.

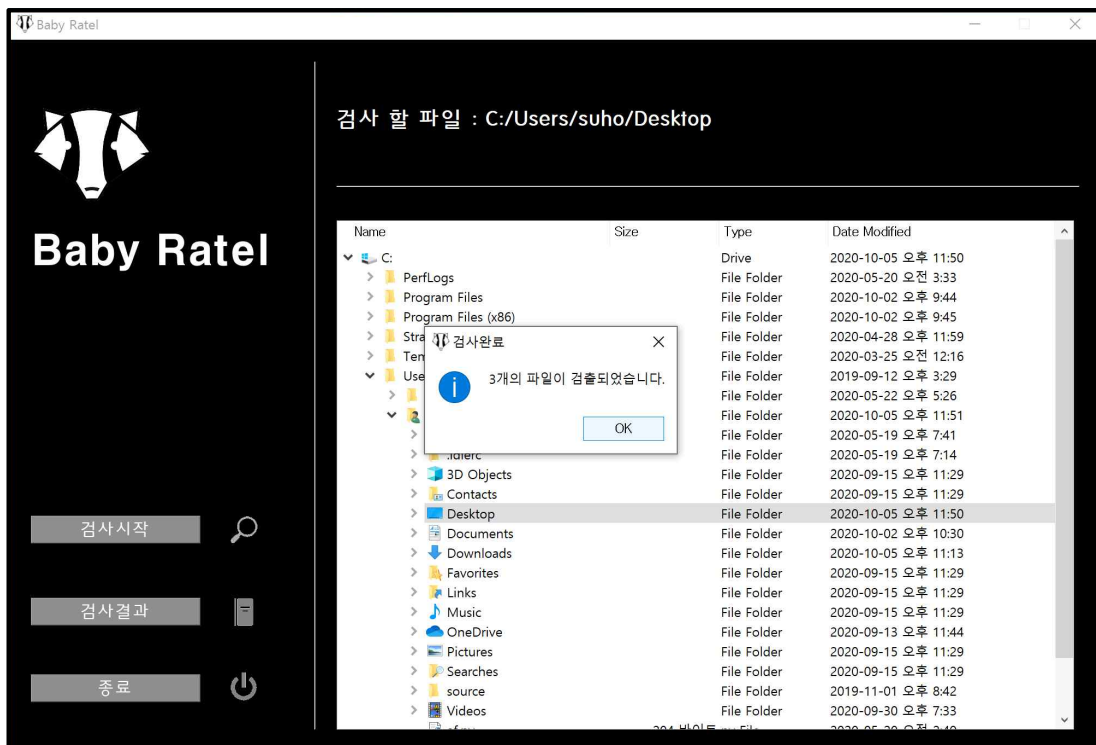
### 4.3 검사시작



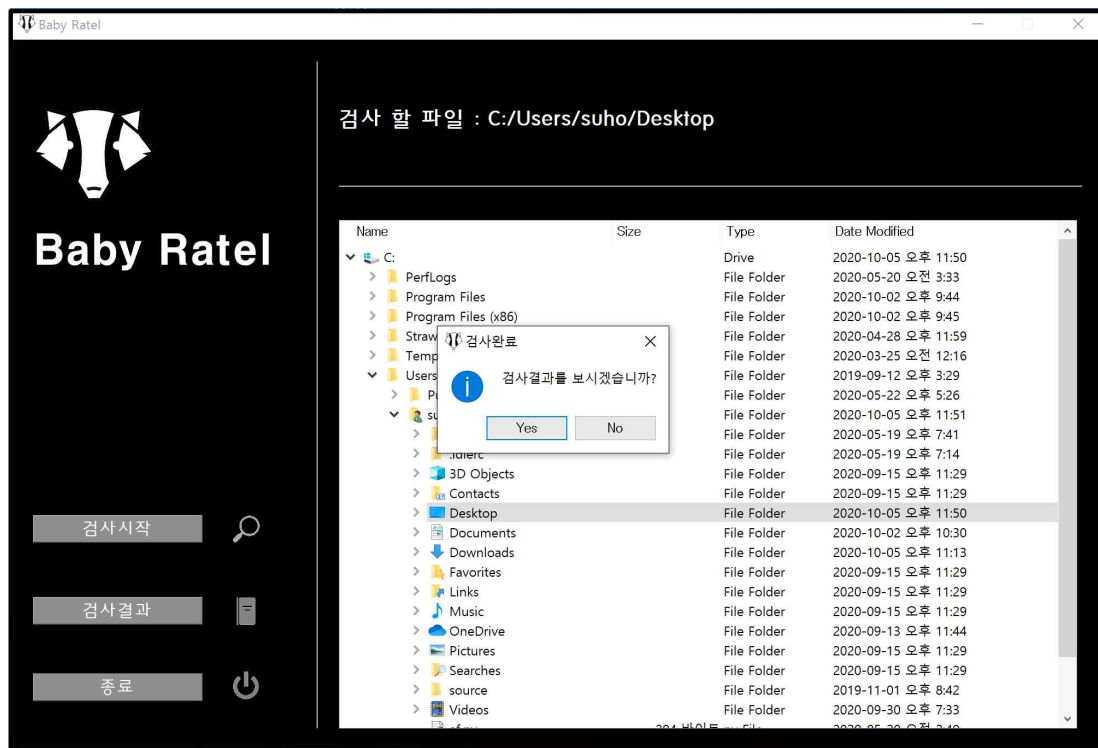
<그림 4.3 - (1) : 프로그램 경로 선택>



<그림 4.3 - (2) : 프로그램 검사완료 1>



<그림 4.3 - (3) : 프로그램 검사완료 2>



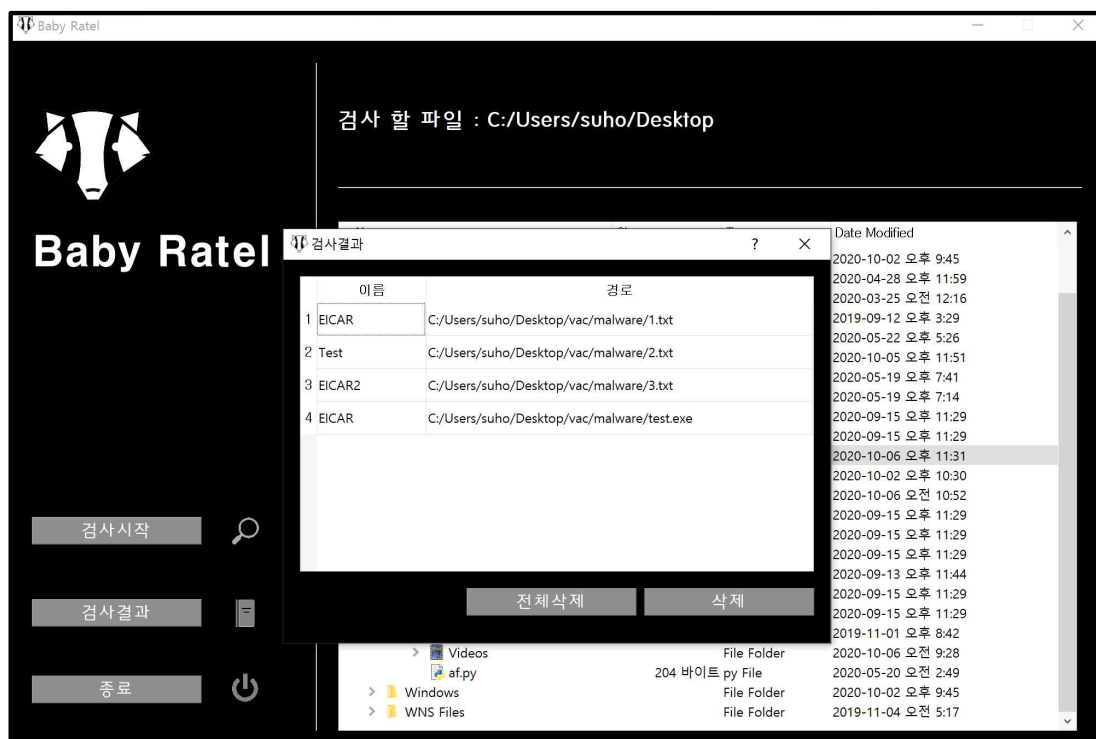
<그림 4.3 - (4) : 프로그램 검사완료 3>

<그림 4.3 - (1, 2, 3, 4)>는 경로를 선택한 후 검사시작 버튼을 클릭하여 검사를 진행했을 때 나타나는 화면들이다. 검사시작 버튼을 누르면 C 라이브러리에 지정한 경로 값을 인자로 넘겨주며 검사가 진행된다. 검사가 진행되면 디렉토리인지 파일인지에 따라 파일이면 MD5 해시 값을 비교하여 같으면 검사결과로 반환한다. 또, 검사한 파일의 개수와 검출된 파일의 개수도 반환한다. 그렇게 검사가 완료되면 검사완료 메시지가 뜨게 설정하였다. 완료메시지에는 몇 개의 파일이 검사되었고 몇 개의 파일이 검출되었는지 나오면서 검사결과 창을 볼 것인지 안 볼 것인지 선택할 수 있게 설정하였다. 만약 검사결과 창을 보지 않겠다고 하더라도 따로 검사결과 버튼을 클릭해 볼 수 있다. <표 4.3 - (1)>은 검사와 관련된 주요 코드이다.

<표 4.3 - (1) : 검사시작 주요 코드>

Main Window :: scan_btn.connect()	버튼을 눌렀을 때 동작하는 이벤트와 연결한다.
Main Window :: scan_clicked() :: msgBox.critical()	오류 메시지를 출력한다.
Main Window :: scan_clicked() :: msgBox.information()	정보를 전달하는 메시지를 출력한다.
Main Window :: scan_clicked() :: lib.scan(path)	C 라이브러리에 경로를 전달해 검사가 시작된다.
Main Window :: scan_clicked() :: ctypes.c_char_p().value.decode('utf-8')	반환된 포인트형 문자열인 검사결과를 utf-8형식으로 디코딩한다.
Main Window :: scan_clicked() :: path.encode('utf-8')	전달한 경로 값을 utf-8형식으로 인코딩한다.

#### 4.4 검사결과



<그림 4.4 - (1) : 프로그램 검사결과>

<그림 4.4 - (1)>은 프로그램 결과버튼을 클릭했을 때 나오는 창이다. 검사결과를 검사하고 나서 반환받은 값을 출력해준다. 검사결과 버튼을 누르면 SubWindow가 나오며 선택해서 삭제할 수 있는 삭제버튼과 전체를 다 삭제할 수 있는 전체삭제 버튼이 있다. 삭제할 파일을 선택한 후 삭제를 누르면 선택한 파일이 삭제되며 삭제가 완료되었다는 메시지가 출력된다. 전체삭제를 누르면 파일이 전부 삭제되며 전체삭제가 완료되었다는 메시지가 출력된다. <표 4.4 - (1)>은 검사결과 창에 사용된 주요코드이다.

<표 4.4 - (2) : 검사결과 주요 코드>

SubWindow :: QTableWidgetItem()	검사결과를 출력해줄 테이블 위젯을 생성한다.
SubWindow :: QPushButton()	클릭 가능한 버튼을 생성한다.
SubWindow :: QMessageBox()	메시지박스를 생성한다.
SubWindow :: report.cellClicked.connect(cell_clicked())	테이블 셀을 클릭했을 때의 이벤트와 연결한다.
SubWindow :: delete_Btn.clicked.connect(delete_clicked())	삭제 버튼을 클릭했을 때의 이벤트와 연결한다.
SubWindow :: deleteAll_Btn.clicked.connect (deleteAll_clicked())	전체삭제 버튼을 클릭했을 때의 이벤트와 연결한다.
SubWindow :: delete_clicked() :: os.remove(경로값)	시스템콜 명령으로 파일을 삭제하는데 사용된다.
SubWindow :: deleteAll_clicked() :: os.remove(경로값)	시스템콜 명령으로 파일을 삭제하는데 사용된다.

## 5. 실험

### 5.1 테스트 파일을 이용한 검사

### ANTI MALWARE TESTFILE

#### Intended use

**Additional notes:**

- This file used to be named ducklin.htm or ducklin-html.htm or similar based on its original author Paul Ducklin and was made in cooperation with CARO.
- The definition of the file has been refined 1 May 2003 by Eddy Willems in cooperation with all vendors.
- The content of this documentation (title-only) was adapted 1 September 2006 to add verification of the activity of anti-malware or anti-spyware products. It was decided not to change the file itself for backward-compatibility reasons.

**Who needs the Anti-Malware Testfile**  
(read the complete text, it contains important information)  
Version of 7 September 2006

If you are active in the anti-virus research field, then you will regularly receive requests for virus samples. Some requests are easy to deal with: they come from fellow-researchers whom you know well, and whom you trust. Using strong encryption, you can send them what they have asked for by almost any medium (including across the Internet) without any real risk.

Other requests come from people you have never heard from before. There are relatively few laws (though some countries do have them) preventing the secure exchange of viruses between consenting individuals, though it is clearly irresponsible for you simply to make viruses available to anyone who asks. Your best response to a request from an unknown person is simply to decline politely.

A third set of requests come from exactly the people you might think would be least likely to want viruses „users of anti-virus software“. They want some way of checking that they have deployed their software correctly, or of deliberately generating a „virus incident in order to test their corporate procedures, or of showing others in the organisation what they would see if they were hit by a virus“.

**Reasons for testing anti-virus software**

Obviously, there is considerable intellectual justification for testing anti-virus software against real viruses. If you are an anti-virus vendor, then you do this (or should do it) before every release of

#### Download Anti Malware Testfile

In order to facilitate various scenarios, we provide 4 files for download. The first, eicar.com, contains the ASCII string as described above. The second file, eicar.com.txt, is a copy of this file with a different filename. Some readers reported problems when downloading the first file, which can be circumvented when using the second version. Just download and rename the file to „eicar.com“. That will do the trick. The third version contains the test file inside a zip archive. A good anti-virus scanner will spot a „virus“ inside an archive. The last version is a zip archive containing the third file. This file can be used to see whether the virus scanner checks archives more than only one level deep.

Once downloaded run your AV scanner. It should detect at least the file „eicar.com“. Good scanners will detect the „virus“ in the single zip archive and may be even in the double zip archive. Once detected the scanner might not allow you any access to the file(s) anymore. You might not even be allowed by the scanner to delete these files. This is caused by the scanner which puts the file into quarantine. The test file will be treated just like any other real virus infected file. Read the user's manual of your AV scanner what to do or contact the vendor/manufacturer of your AV scanner.

**IMPORTANT NOTE**

EICAR cannot be held responsible when these files or your AV scanner in combination with these files cause any damage to your computer. **YOU DOWNLOAD THESE FILES AT YOUR OWN RISK.** Download these files only if you are sufficiently secure in the usage of your AV scanner. EICAR cannot and will not provide any help to remove these files from your computer. Please contact the manufacturer/vendor of your AV scanner to seek such help.

Download area using the standard protocol HTTP			
– Sorry, HTTP download ist temporarily not provided. –			
Download area using the secure, SSL enabled protocol HTTPS			
eicar.com 68 Bytes	eicar.com.txt 68 Bytes	eicar_com.zip 184 Bytes	eicarcom2.zip 308 Bytes

**How to delete the test file from your PC**

<그림 5.1 - (1) : EICAR 테스트 파일 다운로드>

<그림 5.1 - (1)>은 EICAR에서 제공하는 테스트 파일을 다운로드 할 수 있는 곳이다. 상용화된 백신들에서도 바이러스로 탐지되지만 컴퓨터에 아무 영향도 없어서 안전하게 테스트 할 수 있다.

63  
/ 65

① 63 engines detected this file

275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f

eicar.com-6518

68.00 B  
Size

Community Score

attachment text via-tor

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 10+

Basic Properties ①

MD5 44d88612fea8a8f36de82e1278abb02f

SHA-1 3395856ce81f2b7382dee72602f798b642f1410

SHA-256 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f

SSDEEP 3:a+JraNvsgzVqSwHq9:tJuOgzsko

File type Text

Magic ASCII text, with no line terminators

File size 68.00 B (68 bytes)

<eicar test file>

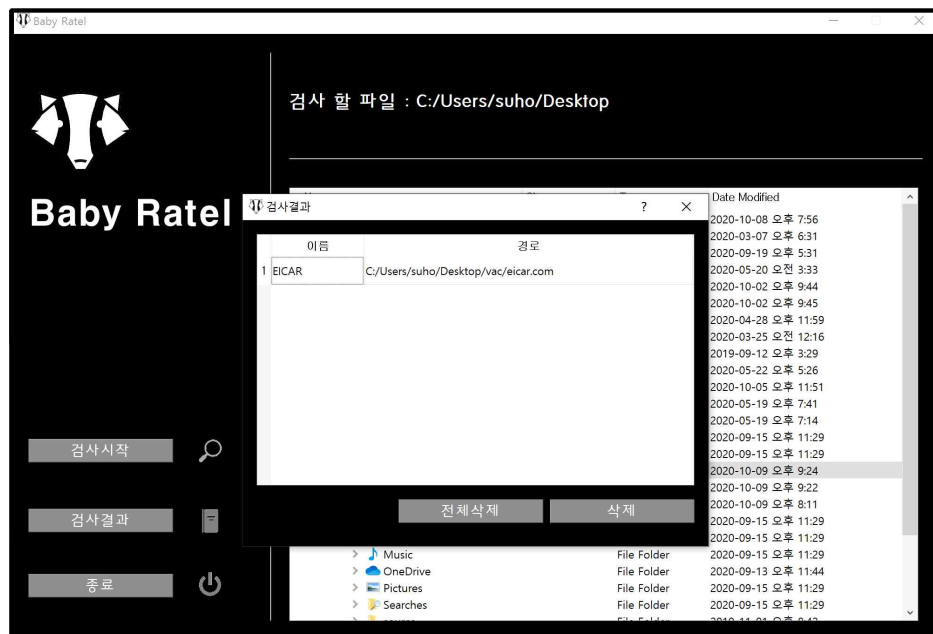
eicar

<그림 5.1 - (2) : EICAR 테스트 파일 분석>

- 18 -



<그림 5.1 - (2)>는 EICAR에서 제공하는 파일을 분석한 내용이다. MD5 해시 값이 분석이 되었으니 DB에 값을 추가한 뒤, 검사를 해보았다.

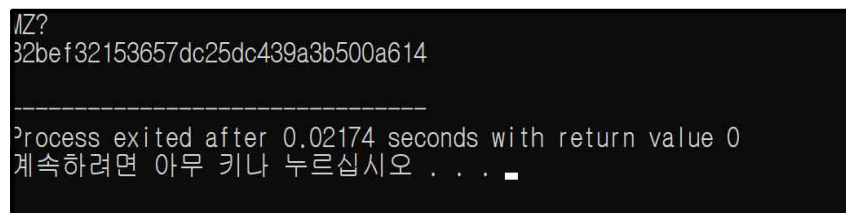


<그림 5.1 - (3) : EICAR 테스트 파일 검사>

<그림 5.1 - (3)>은 테스트 파일로 검사를 한 결과이다. C 라이브러리를 읽고 문자열의 인코딩과 디코딩이 잘 되어서 검사결과가 잘 출력된 것을 볼 수 있다.

## 5.2 실제 바이러스 파일을 이용한 검사

이 테스트는 실제 바이러스 파일을 이용한 검사이기 때문에 가상머신을 이용한 가상환경에서 진행하였다. 바이러스 샘플을 다운받을 수 있는 곳에서 샘플을 다운받아 검사를 해보았다. 하지만 검출이 되지 않았다. 그래서 확인해보니 MD5 해시 값이 달라서 검출이 되지 않았다.



<그림 5.2 - (1) : 바이러스 파일 내용과 MD5 값>

<그림 5.2 - (1)>은 실제 바이러스 샘플 파일의 내용과 MD5 해시 값을 출력해본 결과이다. 내용이 읽히다말고 중간에서 끊겨 MD5 해시 값도 변형이 되어서 출력이 되었다. 파일을 읽는 부분에서 수정이 필요한 것 같다. 아쉽게도 이 부분은 해결하지 못했다.

## 6. 결론

### 6.1 결론

본 연구는 Python Extending을 이용해 Python과 다른 언어를 함께 사용하여 간단한 프로그램을 제작하는 것이다. 다른 언어와 함께 사용할 경우 언어마다 데이터 타입이 다른 경우도 있고 문자열의 경우 인코딩과 디코딩을 해줘야 제대로 전달이 되고 반환도 되었다. 프로그램은 테스트 파일을 이용하여 실험을 해본 결과 잘 작동이 되는 것을 확인했고, 실제 바이너스 파일은 읽지 못해서 조금 아쉬운 부분이 있었다. Python Extending은 컴퓨터가 좋아짐에 따라 처리속도가 작은 프로그램에선 크게 차이가 없지만 연산이 많은 프로그램들에서는 아직도 많이 사용되고 있다. 처리속도가 부족한 Python과 다른 언어를 같이 쓴다면 효율이 좋은 프로그램을 만들 수 있을 것이다.

### 6.2 향후 과제

본 연구의 향후과제는 Python Extending은 잘 구현이 되었다. 하지만 프로그램의 부족한 부분이 많다.

(1) DB파일을 따로 분리하여 암호화와 복호화 할 것.

DB파일을 따로 분리시켜 암호화와 복호화를 진행시켜 DB의 은밀성과 안전성을 향상시켜야 한다. 지금은 DB가 C 라이브러리 파일안에 그냥 들어가 있기 때문에 은밀성과 안전성이 많이 떨어진다.

(2) 실제 바이너스 파일을 이용한 검사.

실제 바이너스 파일을 읽지 못한 부분을 좀 더 연구해서 프로그램의 부족한 부분들을 채울 생각이다. 바이너리 파일을 읽는 곳에서 문제가 발생한 듯 보여 그 부분에서의 코드를 좀 더 생각해볼 예정이다.

## 참고문헌

- [1] 최원혁, “파이썬으로 배우는 Anti-Virus 구조와 원리”, 비제이퍼블릭, 2017
- [2] PyQt5 Tutorial <https://wikidocs.net/book/2165>
- [3] PyQt5 Designer Manual <https://doc.qt.io/qt-5/qtdesigner-manual.html>
- [4] MD5 Message-Digest Alogrithm <https://www.ietf.org/rfc/rfc1321>
- [5] MD5 위키백과 <https://ko.wikipedia.org/wiki/MD5>

## 부록

### 소스 코드 [PyQt5 - MainWindow]

```
import sys
import ctypes
import os
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QApplication
from PyQt5.QtGui import QIcon

# 메인윈도우
class M_Window(QMainWindow):
    # 검사결과 관련 클래스 변수
    split_rway = None
    split_rname = None
    report_handler = 0

    def __init__(self):
        super().__init__()

        # C 라이브러리 참조
        self.lib = ctypes.CDLL("./lib_engine.so")

        # 라인
        self.v_Line = QFrame(self)
        self.h_Line = QFrame(self)

        # 버튼
        self.scan_Btn = QPushButton("검사시작", self)
        self.report_Btn = QPushButton("검사결과", self)
        self.exit_Btn = QPushButton("종료", self)

        # 라벨
        self.path_Label = QLabel(self)
        self.title_Label = QLabel(self)
        self.scan_Img = QLabel(self)
        self.report_Img = QLabel(self)
        self.exit_Img = QLabel(self)
        self.title_Img = QLabel(self)
```

## 소스 코드 [PyQt5 - MainWindow]

```
# 파일트리뷰
self.path = "C:"
self.index = None
self.tv = QTreeView(self)
self.model = QFileSystemModel(self)
self.pname = None
self.virus_num = 0

# 메시지박스
self.msgBox1 = QMessageBox(self)
self.msgBox2 = QMessageBox(self)
self.msgBox3 = QMessageBox(self)
self.msgBox4 = QMessageBox(self)
self.msgBox5 = QMessageBox(self)
self.msgBox6 = QMessageBox(self)

self.initUI()

def initUI(self):
    # 메인윈도우 창 설정
    self.setWindowTitle("Baby Ratel")
    self.setGeometry(500, 400, 2000, 1300)
    self.setFixedSize(2000, 1300)
    self.setStyleSheet("QMainWindow{background-color: rgb(0, 0, 0);}")
    self.setWindowIcon(QIcon("ratel_img.png"))

    # 메시지박스 스타일시트
    self.msgBox1.setStyleSheet("QMessageBox{background-color: rgb(255, 255, 255);}")
    self.msgBox2.setStyleSheet("QMessageBox{background-color: rgb(255, 255, 255);}")
    self.msgBox3.setStyleSheet("QMessageBox{background-color: rgb(255, 255, 255);}")
    self.msgBox4.setStyleSheet("QMessageBox{background-color: rgb(255, 255, 255);}")
    self.msgBox5.setStyleSheet("QMessageBox{background-color: rgb(255, 255, 255);}")
```

## 소스 코드 [PyQt5 - MainWindow]

```
self.msgBox6.setStyleSheet("QMessageBox{background-color:  rgb(255,  255, 255);}")

# 수직 라인
self.v_Line.setGeometry(550, 40, 21, 1221)
self.v_Line.setStyleSheet("background-color: line-rgb(255, 255, 255);")
self.v_Line.setFrameShape(QFrame.VLine)
self.v_Line.setFrameShadow(QFrame.Sunken)

# 수평 라인
self.h_Line.setGeometry(600, 260, 1361, 16)
self.h_Line.setStyleSheet("background-color: line-rgb(255, 255, 255);")
self.h_Line.setFrameShape(QFrame.HLine)
self.h_Line.setFrameShadow(QFrame.Sunken)

# 검사시작 버튼
self.scan_Btn.setGeometry(40, 870, 311, 51)
self.scan_Btn.setStyleSheet("background-color:  rgb(139,  139,  139); color: rgb(255, 255, 255); font: 75 12pt \"HY강M\";")
self.scan_Btn.clicked.connect(self.scan_clicked)

# 검사시작 이미지
self.scan_Img.setGeometry(400, 846, 100, 100)
self.scan_Img.setPixmap(QtGui.QPixmap("scan_img.png"))

# 검사결과 버튼
self.report_Btn.setGeometry(40, 1020, 311, 51)
self.report_Btn.setStyleSheet("background-color:  rgb(139,  139,  139); color: rgb(255, 255, 255); font: 75 12pt \"HY강M\";")
self.report_Btn.clicked.connect(self.report_clicked)

# 검사결과 이미지
self.report_Img.setGeometry(400, 996, 100, 100)
self.report_Img.setPixmap(QtGui.QPixmap("report_img.png"))

# 종료 버튼
self.exit_Btn.setGeometry(40, 1160, 311, 51)
self.exit_Btn.setStyleSheet("background-color:  rgb(139,  139,  139); color: rgb(255, 255, 255); font: 75 12pt \"HY강M\";")
```

## 소스 코드 [PyQt5 - MainWindow]

```
self.exit_Btn.clicked.connect(self.exit_clicked)

# 종료 이미지
self.exit_Img.setGeometry(400, 1132, 100, 100)
self.exit_Img.setPixmap(QtGui.QPixmap("exit_img.png"))

# 경로 라벨
self.path_Label.setGeometry(600, 120, 1001, 51)
self.path_Label.setStyleSheet("color: rgb(255, 255, 255); font: 75 15pt \"HY강
M\"; font-weight: bold;")
self.path_Label.setText("검사 할 파일 : ")

# 제목 라벨
self.title_Label.setGeometry(40, 250, 521, 271)
self.title_Label.setStyleSheet("color: rgb(255, 255, 255); font: 100 28pt \"HY
견고딕\";")
self.title_Label.setText("Baby Ratel")

self.title_Img.setGeometry(40, 60, 300, 300)
self.title_Img.setPixmap(QtGui.QPixmap("ratel_Img.png"))

# 트리뷰 파일 탐색기
self.model.setRootPath(self.path)
self.tv.setModel(self.model)
self.tv.setGeometry(600, 330, 1351, 931)
self.tv.setStyleSheet("background-color: rgb(255, 255, 255);")
self.tv.setColumnWidth(0, 500)
self.tv.clicked.connect(self.setIndex)

self.show()

# 검사시작 버튼 클릭 이벤트
def scan_clicked(self):
    if self.pname == None:
        self.msgBox1.critical(self.msgBox1, '오류', '검사할 파일을 선택해주세요.',
QMessageBox.Ok)
```



## 소스 코드 [PyQt5 - MainWindow]

```
else:
    self.lib.init()
    M_Window.report_handler = 0
    path = self.pname
    e_Path = path.encode('utf-8')
    self.r_scan = self.lib.scan(e_Path)

    if self.r_scan == 1:
        file_num = self.lib.r_filenum()
        self.virus_num = self.lib.r_virusnum()
        self.msgBox2.information(self.msgBox2, '검사완료', '총 ' +
str(file_num) + '개의 파일의 검사가 완료되었습니다.', QMessageBox.Ok)
        self.msgBox4.information(self.msgBox4, '검사완료',
str(self.virus_num) + '개의 파일이 검출되었습니다.', QMessageBox.Ok)
        rway = self.lib.r_way()
        d_rway = ctypes.c_char_p(rway).value.decode('utf-8')
        M_Window.split_rway = d_rway.split(',')
        rname = self.lib.r_name()
        d_rname = ctypes.c_char_p(rname).value.decode('utf-8')
        M_Window.split_rname = d_rname.split(',')

        if self.virus_num != 0:
            ans = self.msgBox5.information(self.msgBox5, '검사완료', '검사결
과를 보시겠습니까?', QMessageBox.Yes | QMessageBox.No)

            if ans == QMessageBox.Yes:
                self.report_clicked()

# 검사결과 버튼 클릭 이벤트
def report_clicked(self):
    if self.virus_num != 0 and M_Window.report_handler != 1:
        Sub_Window(self)
    else:
        self.msgBox6.warning(self.msgBox6, '오류', '검사결과가 없습니다.',
QMessageBox.Ok)
```

## 소스 코드 [PyQt5 - MainWindow]

```
# 종료 버튼 클릭 이벤트
def exit_clicked(self):
    ans = self.msgBox3.question(self.msgBox3, "종료", "종료하시겠습니까?",
QMessageBox.Yes | QMessageBox.No)
    if ans == QMessageBox.Yes:
        sys.exit()

# 경로 설정
def setIndex(self, index):
    self.index = index
    self.pname = self.model.filePath(self.index)
    self.path_Label.setText("검사 할 파일 : " + self.pname)

def r_way(self):
    return M_Window.split_rway

def r_name(self):
    return M_Window.split_rname

# 삭제 이벤트
def remove_report(self, v):
    del M_Window.split_rname[v]
    del M_Window.split_rway[v]

# 전체삭제 이벤트
def removeAll_report(self):
    del M_Window.split_rway
    del M_Window.split_rname
    M_Window.report_handler = 1
```

## 소스 코드 [PyQt5 - SubWindow]

```
# 검사결과 창
class Sub_Window(QDialog):
    def __init__(self, parent):
        super(Sub_Window, self).__init__(parent)

        # 결과(테이블)
        self.report = QTableWidget(self)
        self.column = 2
        self.row = parent.virus_num
        self.column_Headers = ['이름', '경로']

        # 버튼
        self.delete_Btn = QPushButton("삭제", self)
        self.deleteAll_Btn = QPushButton("전체삭제", self)

        # 메시지박스
        self.msgBox1 = QMessageBox(self)
        self.msgBox2 = QMessageBox(self)
        self.msgBox3 = QMessageBox(self)
        self.msgBox4 = QMessageBox(self)
        self.msgBox5 = QMessageBox(self)

        # 이름, 경로
        self.way = parent.r_way()
        self.name = parent.r_name()
        self.delete_idx = None
        self.x = None

        self.initUI()

    def initUI(self):
        # 결과 창 설정
        self.setWindowTitle("검사결과")
        self.setGeometry(1000, 800, 1000, 700)
        self.setFixedSize(1000, 700)
        self.setStyleSheet("QDialog {background-color: rgb(0, 0, 0);}")
```

## 소스코드 [PyQt5 - SubWindow]

```
#메세지 박스 스타일시트
self.msgBox1.setStyleSheet("QMessageBox{background-color:  rgb(255,  255,
255);}")
self.msgBox2.setStyleSheet("QMessageBox{background-color:  rgb(255,  255,
255);}")
self.msgBox3.setStyleSheet("QMessageBox{background-color:  rgb(255,  255,
255);}")
self.msgBox4.setStyleSheet("QMessageBox{background-color:  rgb(255,  255,
255);}")
self.msgBox5.setStyleSheet("QMessageBox{background-color:  rgb(255,  255,
255);}")

# 테이블 설정
self.report.setGeometry(30, 30, 940, 540)
self.report.setColumnCount(self.column)
self.report.setRowCount(self.row)
self.report.setEditTriggers(QAbstractItemView.NoEditTriggers)
self.report.setHorizontalHeaderLabels(self.column_Headers)
for k, v in enumerate(self.name):
    col = 0
    item = QTableWidgetItem(v)
    self.report.setItem(k, col, item)
for k, v in enumerate(self.way):
    col = 1
    item = QTableWidgetItem(v)
    self.report.setItem(k, col, item)
self.report.setColumnWidth(1, 706)

# 삭제 버튼
self.delete_Btn.setGeometry(660, 600, 311, 51)
self.delete_Btn.setStyleSheet("background-color:  rgb(139, 139, 139); color:
rgb(255, 255, 255); font: 75 12pt \"HY강함\";")
self.report.cellClicked.connect(self.cell_clicked)
self.delete_Btn.clicked.connect(self.delete_clicked)
```

## 소스 코드 [PyQt5 - SubWindow]

```
# 전체삭제 버튼
self.deleteAll_Btn.setGeometry(335, 600, 311, 51)
self.deleteAll_Btn.setStyleSheet("background-color: rgb(139, 139, 139); color:
rgb(255, 255, 255); font: 75 12pt \"HY강M\";")
self.deleteAll_Btn.clicked.connect(self.deleteAll_clicked)

self.show()

# 테이블 클릭 이벤트
def cell_clicked(self):
    self.x = self.report.selectedIndexes()
    if self.x[0].column() == 0:
        self.msgBox1.critical(self.msgBox1, '오류', '파일 경로를 선택해주시오.',
QMessageBox.Ok)
        self.delete_idx = self.report.item(self.x[0].row(), self.x[0].column()).text()
        # 디버그 : print(self.delete_idx)

# 삭제 버튼 클릭 이벤트
def delete_clicked(self):
    if self.delete_idx == None:
        self.msgBox1.warning(self.msgBox1, '오류', '선택된 파일이 없습니다',
QMessageBox.Ok)
    else:
        ans = self.msgBox2.question(self.msgBox2, '삭제', '선택된 파일을 삭제하시
겠습니까?', QMessageBox.Yes | QMessageBox.No)

        if ans == QMessageBox.Yes:
            os.remove(self.delete_idx)
            self.report.removeRow(self.x[0].row())
            M_Window().remove_report(self.x[0].row())
            self.msgBox3.information(self.msgBox3, '삭제완료', '선택된 파일이 삭
제 완료되었습니다.', QMessageBox.Ok)
```

## 소스 코드 [PyQt5 - SubWindow]

```
# 전체 삭제 버튼 클릭 이벤트
def deleteAll_clicked(self):
    ans = self.msgBox4.question(self.msgBox4, '전체삭제', '전체 파일을 삭제하시겠습니까?', QMessageBox.Yes | QMessageBox.No)

    if ans == QMessageBox.Yes:
        for i in range(self.row):
            col = 1
            remove_way = self.report.item(i, col).text()
            #os.remove(remove_way)
            self.report.clearContents()
            M_Window().removeAll_report()
            self.msgBox5.information(self.msgBox5, '삭제완료', '파일 삭제가 완료되었습니다.', QMessageBox.Ok)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = M_Window()
    sys.exit(app.exec_())
```

## 소스코드 [C 코드 - lib-engine.c]

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h> // 디렉토리 관련 헤더파일
#include <string.h>
#include <sys/stat.h> // 파일의 정보 관련 헤더파일
#include <unistd.h> // 여러가지 자료형
#include "md5.h"

#define BUF_SIZE 2048
#define db_Num 3

// 경로, 이름, 파일갯수, 검사된 파일 갯수
char way[BUF_SIZE];
char name[BUF_SIZE];
int file_num, virus_num;

// db 이름 / 바이러스 내용
struct dbInfo{
    char name[30];
    char *str;
};

struct dbInfo db[db_Num]={
    {"EICAR", "44d88612fea8a8f36de82e1278abb02f"},
    {"Test", "77bff0b143e4840ae73d4582a8914a43"},
    {"EICAR2", "6a4cd5563a37ee5b97a319a27066d8c6"},
};

// MD5 해시값 변환
char *md5(char *msg){
    int i = 0;
    unsigned char digest[16] = {0,};
    static char out_Hash[50] = {0,};
    md5_st md5_ctx;

    md5_init(&md5_ctx);
    md5_convert(&md5_ctx, msg, strlen(msg));
    md5_gethash(&md5_ctx, digest);
```

## 소스코드 [C 코드 - lib-engine.c]

```
        for (i = 0; i<16; i++)
            sprintf(out_Hash + (i * 2), "%02x", digest[i]);

        return out_Hash;
    }

// 검사시작
int scan(const char* path){
    DIR *dr = NULL;
    struct dirent *de = NULL;
    struct stat fi;
    char fn[BUF_SIZE];
    char *buf_MD5;
    FILE *fp;
    int i;

    if((dr = opendir(path)) == NULL){
        printf("%s\n", path);
        printf("Could not open directory");
    }

    while((de = readdir(dr)) != NULL){
        char buf[BUF_SIZE]={0,};
        if(strcmp(de->d_name, ".") == 0 || strcmp(de->d_name, "..") == 0)
            continue;

        sprintf(fn, "%s/%s", path, de->d_name);

        if(stat(fn, &fi) == -1){
            continue;
        }
    }
}
```



## 소스코드 [C 코드 - lib-engine.c]

```
        if(S_ISDIR(fi.st_mode)){
            scan(fn);
        }else if(S_ISREG(fi.st_mode)){
            fp = fopen(fn, "rb");

            if(fp==NULL){
                perror("File open failed\n");
                exit(0);
            }

            fread(buf, sizeof(buf), 1, fp);
            fclose(fp);
            buf_MD5 = md5(buf);

            for(i = 0; i<db_Num; i++){
                if(!strcmp(buf_MD5, db[i].str)){
                    chmod(fn, 0777);
                    strcat(way, fn);
                    strcat(way, ",");
                    strcat(name, db[i].name);
                    strcat(name, ",");
                    virus_num++;
                }
            }
            file_num++;
        }
        closedir(dr);
        return 1;
    }

// 결과 반환
char *r_way(){
    return way;
}

char *r_name(){
    return name;
}
```

## 소스코드 [C 코드 - lib-engine.c]

```
int r_filenum(){
    return file_num;
}

int r_virusnum(){
    return virus_num;
}

// 초기화
void init(){
    strcpy(way, "\\0");
    strcpy(name, "\\0");
    file_num = 0;
    virus_num = 0;
}
```

## 소스코드 [C 코드 - md5.h]

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>

#define leftrotation(x, c) (((x) << (c)) | ((x) >> (32 - (c))))
#define md5_bytes 64
#define md5_buf_size (md5_bytes*2)

// 블록 4개, 문자열 버퍼
typedef struct _md5_st{
    unsigned int b0;
    unsigned int b1;
    unsigned int b2;
    unsigned int b3;

    char buf[md5_buf_size];
    unsigned int buf_stored_len;

    unsigned __int64 updated_len;
}md5_st;

// 라운드당 시프트
static const unsigned int md5_r[md5_bytes] = {
    7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,
    5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20,
    4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,
    6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21
};
```

## 소스코드 [C 코드 - md5.h]

```
// K 값 테이블
static const unsigned int md5_k[md5_bytes] = {
    0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdcee5, 0xf57c0faf, 0x4787c62a,
    0xa8304613, 0xfd469501,
    0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be, 0x6b901122, 0xfd987193,
    0xa679438e, 0x49b40821,

    0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa, 0xd62f105d, 0x02441453,
    0xd8a1e681, 0xe7d3fbc8,
    0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed, 0xa9e3e905, 0xfcefa3f8,
    0x676f02d9, 0x8d2a4c8a,

    0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c, 0xa4bbee44, 0x4bdecfa9,
    0xf6bb4b60, 0xbebfbc70,
    0x289b7ec6, 0xea127fa, 0xd4ef3085, 0x04881d05, 0xd9d4d039, 0xe6db99e5,
    0x1fa27cf8, 0xc4ac5665,

    0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039, 0x655b59c3, 0x8f0ccc92,
    0xffeff47d, 0x85845dd1,
    0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1, 0xf7537e82, 0xbd3af235,
    0x2ad7d2bb, 0xeb86d391
};

// 완성된 해쉬코드 넣는 함수
void md5_puthash(unsigned int hash, unsigned char *dst){
    if (!dst) return;

    dst[0] = (unsigned char)(hash);
    dst[1] = (unsigned char)(hash >> 8);
    dst[2] = (unsigned char)(hash >> 16);
    dst[3] = (unsigned char)(hash >> 24);
}
```

## 소스코드 [C 코드 - md5.h]

```
// 메인 루프
int md5_block_update(md5_st *ctx, char *block){
    int i = 0;
    unsigned int a,b,c,d;
    unsigned int *w;

    if (!ctx || !block)
        return -1;

    a = ctx->b0;
    b = ctx->b1;
    c = ctx->b2;
    d = ctx->b3;
    w = (unsigned int*)block;

    // 메인 라운드
    while (i < md5_bytes){
        unsigned int f = 0;
        unsigned int g = 0;
        unsigned int temp = 0;

        if (i < 16){
            f = (b & c) | ((~b) & d);
            g = i;
        }else if (i < 32){
            f = (d & b) | ((~d) & c);
            g = (5 * i + 1) % 16;
        }else if (i < 48){
            f = b ^ c ^ d;
            g = (3 * i + 5) % 16;
        }else{
            f = c ^ (b | (~d));
            g = (7 * i) % 16;
        }
    }
```

## 소스코드 [C 코드 - md5.h]

```
        temp = d;
        d = c;
        c = b;
        b = b + leftrotation((a + f + md5_k[i] + w[g]), md5_r[i]);
        a = temp;

        i++;
    }

    // 해쉬 결과 추가
    ctx->b0 += a;
    ctx->b1 += b;
    ctx->b2 += c;
    ctx->b3 += d;
    ctx->updated_len += md5_bytes;

    return 0;
}

// 초기화
int md5_init(md5_st *ctx){
    if(!ctx) return -1;

    memset(ctx, 0, sizeof(md5_st));

    ctx->b0 = 0x67452301;
    ctx->b1 = 0xefcdab89;
    ctx->b2 = 0x98badcfe;
    ctx->b3 = 0x10325476;

    ctx->buf_stored_len = 0;
    ctx->updated_len = 0;

    return 0;
}
```

## 소스코드 [C 코드 - md5.h]

```
// 메시지를 512bit로 자르는 함수
int md5_convert(md5_st *ctx, char *msg, int msg_len){
    int remain_len = msg_len;
    int offset = 0;

    if (!ctx || !msg || msg_len <= 0) return -1;
    if (ctx->buf_stored_len < 0) return -1;

    if (ctx->buf_stored_len > 0){
        int to_copy_len = md5_bytes - ctx->buf_stored_len;
        to_copy_len = (to_copy_len > remain_len) ? remain_len :
to_copy_len;

        memcpy(&ctx->buf[ctx->buf_stored_len], msg, to_copy_len);

        ctx->buf_stored_len += to_copy_len;
        remain_len -= to_copy_len;
        offset += to_copy_len;

        if (ctx->buf_stored_len == md5_bytes){
            md5_block_update(ctx, ctx->buf);
            ctx->buf_stored_len = 0;
        }
    }

    while (remain_len >= md5_bytes){
        md5_block_update(ctx, &msg[offset]);

        remain_len -= md5_bytes;
        offset += md5_bytes;
    }
}
```

## 소스코드 [C 코드 - md5.h]

```
        if (remain_len){
            memcpy(ctx->buf, &msg[offset], remain_len);
            ctx->buf_stored_len = remain_len;
        }

        return 0;
    }

// 나머지 데이터 처리 후 반환
int md5_gethash(md5_st *ctx, unsigned char digest[16]){
    int offset = 0;
    unsigned __int64 total_msg_bits_size = 0;

    if (!ctx) return -1;

    offset = ctx->buf_stored_len;
    total_msg_bits_size = 8 * (ctx->updated_len + (unsigned
__int64)ctx->buf_stored_len);

    ctx->buf[offset++] = 128;
    memset(&ctx->buf[offset], 0, md5_buf_size - offset);

    if (offset <= md5_bytes - sizeof(__int64)){
        offset = md5_bytes - sizeof(__int64);
    }else{
        offset = md5_buf_size - sizeof(__int64);
    }

    // 리틀 엔디안
    memcpy(&ctx->buf[offset], &total_msg_bits_size, sizeof(__int64));
    offset += sizeof(__int64);

    md5_block_update(ctx, ctx->buf);
    if (offset > md5_bytes) md5_block_update(ctx, &ctx->buf[md5_bytes]);
}
```



소스코드 [C 코드 - md5.h]

```
    // 해쉬코드 넣어서 반환
    md5_puthash(ctx->b0, digest);
    md5_puthash(ctx->b1, digest + 4);
    md5_puthash(ctx->b2, digest + 8);
    md5_puthash(ctx->b3, digest + 12);

    return 0;
}
```

# 2020 졸업자격 실험보고 요약문

학번 : 2015211330

이름 : 최수호

제목 : C라이브러리 확장을 이용한 미니백신 프로그램

## 1. 목표 및 필요성

Python은 최근 인기가 많은 언어입니다. 또 여러 다른 언어와 함께 사용하기에도 효율적인 언어입니다. 그래서 Python Extendig을 통해 Python과 다른 언어를 통합함으로써 Python의 확장성을 기대할 수 있고 Python의 단점을 보완해 줄 수 있습니다. 또한 GUI 부분의 PyQt5의 경우 Python과 C++의 QT를 융합함으로써 크로스플랫폼 어플리케이션 프레임워크와 인터프리터 언어의 장점을 다 가지고 있으면서 기존의 Python의 Tkinter보다 시각적으로도 좋은 디자인을 구현할 수 있습니다.

## 2. 기존 연구 결과

기존 백신 관련해서는 Python로 구현되었고 오픈소스인 Kicom백신이 있습니다.

## 3. 자신이 한 기여도

C언어 코드를 라이브러리 파일로 만들어 Python Extending을 통해 Python에서 사용가능하게끔 했습니다. 그리고 PyQt5를 이용한 GUI 구현 및 기존 백신의 아키텍처를 간단히 이용한 미니백신 프로그램 구현했습니다.

## 4. 실험 결과

Pyqt5에서 C언어 함수를 호출 그리고 반환한 문자열 타입의 인자를 디코딩, 인코딩을 통해 정상적으로 받아오고 보냄, 정상적으로 간단한 백신의 역할을 함.

1) Python

<https://ko.wikipedia.org/wiki/파이썬>

2) Python Extending

<https://docs.python.org/3/extending/index.html>

3) Python Tkinter

<https://en.wikipedia.org/wiki/Tkinter>

4) C 언어

[https://ko.wikipedia.org/wiki/C\\_\(프로그래밍\\_언어\)](https://ko.wikipedia.org/wiki/C_(프로그래밍_언어))

5) Ctypes

<https://docs.python.org/ko/3/library/ctypes.html>

6) MD5

<https://ko.wikipedia.org/wiki/MD5>

7) PyQt

<https://en.wikipedia.org/wiki/PyQt>

8) Qt Designer Manual

<https://doc.qt.io/qt-5/qtdesigner-manual.html>