[BOB6]DF12_Tech_06_문의성

디지털 포렌식 트랙 6 기 문의성

작성일자: 2018. 01. 29.

목차

I 과제 내용

1. 개발환경

해당 프로그램은 Ubuntu 16.04 OS 에서 작업을 진행했습니다. Python 환경은 Python3.5 Version 을 이용했으며 PyQt5 를 통해서 GUI 를 작업했습니다.

Python 3.5 Version

PyQt5

2. 사용법

1) main.py 파일과 같은 경로에 main.ui 파일을 support.py 와 같은경로에 support.ui 를 위치시킵니다.

2) Python3 구동을 위해 필요한 다음 모듈을 설치해야 합니다.

우선 pip 버전이 낮은 경우 먼저 pip 버전의 업데이트가 필요합니다!!

(pip3 로 설치가 안되는 경우 pip 로 설치 후 경로가 2.x 버전에 설치되었다면 3.x lib 폴더로 이동이 필요합니다!!!)

Pip install daemon
Pip3 install PyQt5
Pip3 install dnspython
Pip install csv

3) Python3 를 통해서 main.py 를 실행합니다.(관리자 권한으로)

Main.py 가 실행되면 즉시 'fl0ckfl0ck.info' 도메인에 질의를 시작하며 해당 python 프로그램은 백그라운드에서 돌고 있습니다.(시나리오 상황상 바로 해당 서버에 질의를 할 수 있게 프로그램을 제작 했으며 혹 도메인 변경을 원하시면 main.py 118 번, 122 번 라인에 존재하는

```
fulldata = r.query('fl0ckfl0ck.info').response
```

해당 함수 안의 값을 변경해주면 됩니다!!(도메인 입력 받는 부분을 구현할까 고민하다 시나리오 상 바로 fl0ckfl0ck.info 를 질의하는 것이 맞다고 판단하여 구현하지는 않았습니다!!) main.py 가 실행된 경로에 'collect.csv' 파일이름으로 csv 파일 출력결과물이 생성 됩니다!

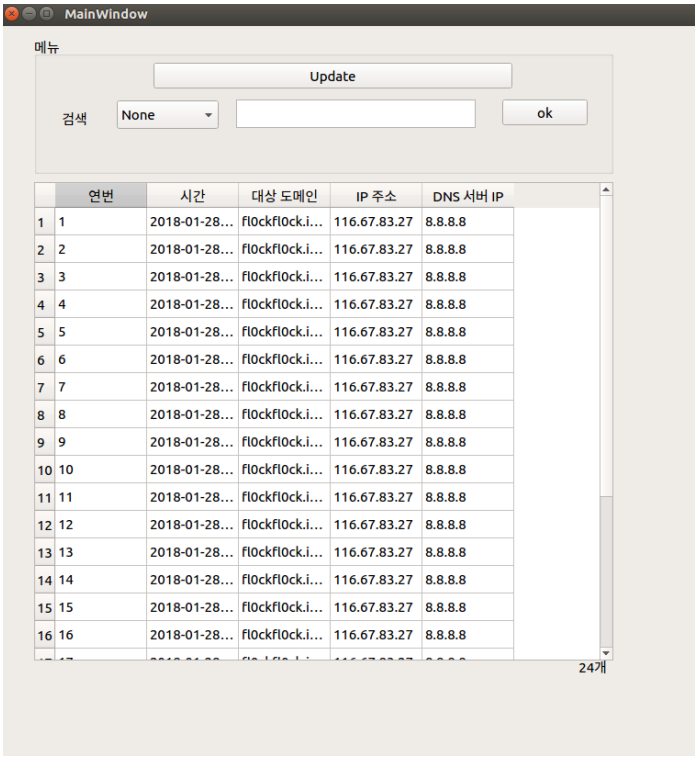
그리고 추가적으로 다음 GUI 창을 확인 할 수 있습니다!

다음의 화면에서 Show All Data 를 누르면 지금까지 질의를 통해서 CSV 파일로 출력한 DNS 정보를 확인할 수 있고 시간, 연변, 대상 도메인, IP 주소 , DNS 서버 IP 에 따른 검색을 진행할 수 있습니다. 추가적으로 아래에는 몇 개의 결과물이 있는지 개수 확인이 가능 합니다.

터미널 및 위의 GUI 프로그램을 종료 했어도 main.py 프로그램은 백그라운드로 계속 동작하며 ps –ef 명령을 통해서 다음과 같이 정상적으로 돌고 있음을 확인할 수 있습니다.

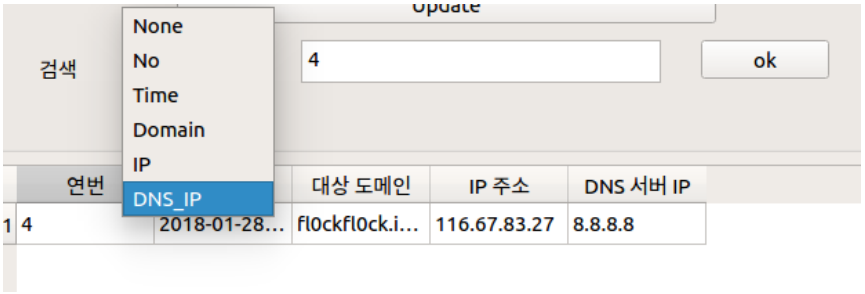
```
root@ubuntu:/home/moon# ps -ef | grep python
root        4362    2213  0 18:12 ?        00:00:00 python3 main.py
root        4440    4429  0 18:17 pts/17   00:00:00 grep --color=auto python
root@ubuntu:/home/moon#
```

#show 눌렀을 때 화면 설명



다음과 같이 정보를 한 눈에 확인이 가능하며 아래에 몇 개의 데이터가 존재하는지 개수 파악이 가능합니다. Update 를 누르게 되면 csv 파일로부터 데이터를 다시 불러들여오기 때문에 만약에 데이터가 새로 갱신된 경우 새로 추가 됩니다.

추가적으로 검색을 통해서 필터링 작업이 가능합니다. 검색기능은 아래와 같이 컬럼 별로 검색이 가능하며 ComboBox 를 통해서 검색할 컬럼 대상을 선택 할 수 있습니다.



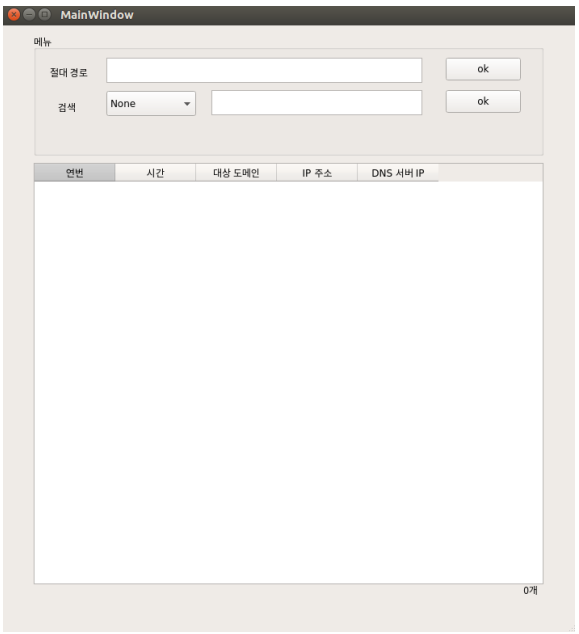
이후 값을 입력 받아 해당 컬럼에 해당 값에 일치하는 값들만 필터링 하는 방식입니다.

간단한 예시로 연번(No) 4 번인 것을 필터링하면

다음의 실행 결과를 확인할 수 있습니다.

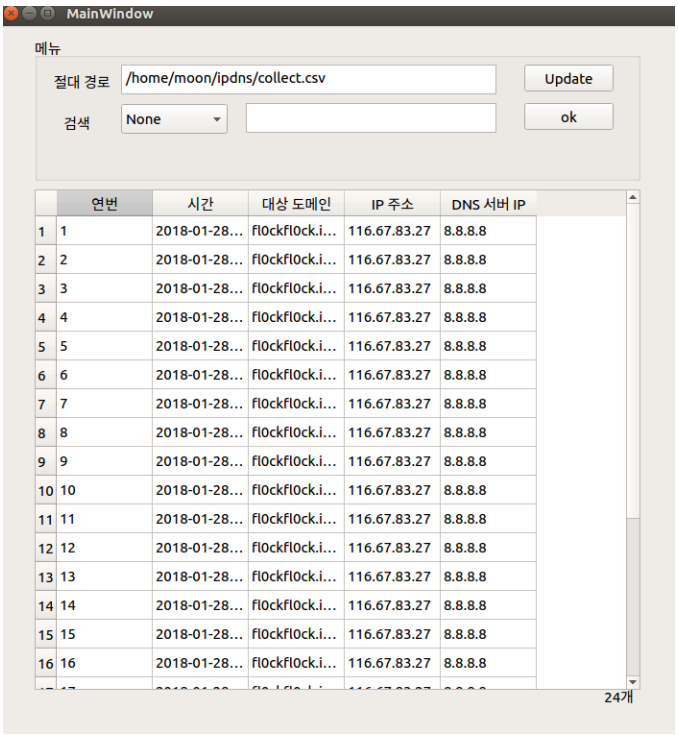
4) Support.py 를 실행합니다(관리자권한으로)

Support.py 는 DNS 를 조회하기 위한 GUI 프로그램으로 해당 프로그램을 python3 로 실행하면 다음과 같은 화면을 볼 수 있습니다.



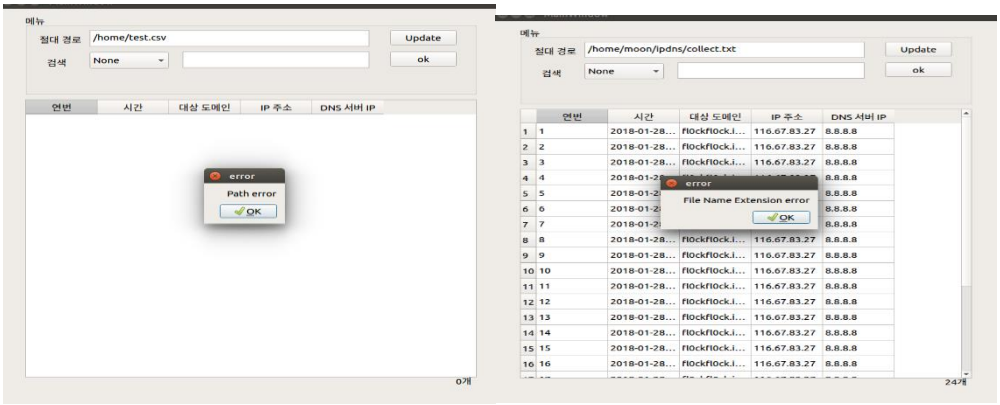
위 GUI 프로그램에서는 우선 절대 경로 부분에 csv 파일의 경로를 입력해 줍니다. Csv 파일은 main.py 를 실행했을 때 출력결과물로 생성되며 main.py 와 같은 경로에 collect.csv 라는 파일이름으로 생성 됩니다.

파일 경로를 입력 후 상위 ok 버튼을 누르면 DNS 정보를 조회할 수 있고 아래의 검색 부분을 통해서 연번, 시간, 대상 도메인, IP 주소, DNS 서버 IP 에 따른 검색이 가능합니다. 추가적으로 아래에 총 검색된 데이터의 개수를 알 수 있습니다.



다음과 같이 올바른 경로를 입력했을 경우 해당 csv 파일로부터 데이터를 불러와 GUI 에 표시해주고 main.py 의 GUI 마찬가지로 검색기능을 제공하고 있습니다.

추가적으로 경로 부분에 csv 파일이 아니거나 해당 경로가 잘못된 경우



다음과 같이 에외 처리를 통해서 에러가 발생되면서 프로그램이 종료합니다.

3. 소스코드

1) Button Click

```
def btn_click(self):
    global path
    path = 'collect.csv'
    self.pushButton.setText("Update")
    try:
        f=open(path,'r')
    except:
        print("path error")
        QMessageBox.about(self, "error", "Path error")
        exit(1)

    r=csv.reader(f)
    while (self.tableWidget.rowCount() > 0):
        self.tableWidget.removeRow(0);
    self.tableWidget.setRowCount=0
    for i in r:
        if(str(i)=='[]'):
            continue
        row=self.tableWidget.rowCount()
        self.tableWidget.insertRow(row)
        self.tableWidget.setItem(row,0,QTableWidgetItem(i[0]))
        self.tableWidget.setItem(row,1,QTableWidgetItem(i[1]))
        self.tableWidget.setItem(row,2,QTableWidgetItem(i[2]))
        self.tableWidget.setItem(row,3,QTableWidgetItem(i[3]))
        self.tableWidget.setItem(row,4,QTableWidgetItem(i[4]))
    self.label.setText(str(self.tableWidget.rowCount()) + "개")
    f.close()
```

위 소스코드는 처음 main.py 실행 시 나오는 GUI 화면 위 버튼 클릭 이벤트에 관한 함수입니다. 간단히 함수의 내용은 버튼 클릭 시 collect.csv 파일을 불러와서 csv 파일 내에 존재하는 데이터를 테이블을 통해 GUI 로 출력하는 역할을 진행합니다. 여기서 해당 버튼의 텍스트가 'Update'로 변경되고 해당 버튼을 다시 누르면 새로 고침 기능이 적용 됩니다!.

아래의 btn_click2 의 경우에는 main.py 실행 시 생성되는 GUI 아래 버튼 클릭 시 일어나는 이벤트와 관련된 함수 입니다. btn_click 함수와 비슷한 동작을 진행하며 차이점은 btn_click2 의 경우 검색을 위한 버튼이기 때문에 combobox 의 내용을 바탕으로 검색할 범위(대상)을 파악하고 if 문을 통해서 검색 대상이 정해지지 않은 경우를 제외하고 정해진 검색 범위(대상)과 해당 범위에서 일치하는 입력 받은 값과 일치하는 데이터를 csv 파일에서 추출하여 테이블에 GUI 형태 출력을 진행해주고 있습니다.

```
def btn_click2(self):
    find_data = self.textEdit_3.toPlainText()
    if(self.comboBox.currentText()!='None'):
        combo_data=str(self.comboBox.currentText())
        f = open(path, 'r')
        r = csv.reader(f)
        while(self.tableWidget.rowCount()>0):
            self.tableWidget.removeRow(0);
        self.tableWidget.setRowCount = 0
        if(combo_data!='None'):
            for i in r:
                if (str(i) == '[]'):
                    continue
                if(str(i[0])==find_data):
                    row = self.tableWidget.rowCount()
                    self.tableWidget.insertRow(row)
                    self.tableWidget.setItem(row, 0, QTableWidgetItem(i[0]))
                    self.tableWidget.setItem(row, 1, QTableWidgetItem(i[1]))
                    self.tableWidget.setItem(row, 2, QTableWidgetItem(i[2]))
                    self.tableWidget.setItem(row, 3, QTableWidgetItem(i[3]))
                    self.tableWidget.setItem(row, 4, QTableWidgetItem(i[4]))

                elif(str(i[1])==find_data):
                    row = self.tableWidget.rowCount()
                    self.tableWidget.insertRow(row)
                    self.tableWidget.setItem(row, 0, QTableWidgetItem(i[0]))
                    self.tableWidget.setItem(row, 1, QTableWidgetItem(i[1]))
                    self.tableWidget.setItem(row, 2, QTableWidgetItem(i[2]))
                    self.tableWidget.setItem(row, 3, QTableWidgetItem(i[3]))
                    self.tableWidget.setItem(row, 4, QTableWidgetItem(i[4]))

                elif (str(i[2]) == find_data):
                    row = self.tableWidget.rowCount()
                    self.tableWidget.insertRow(row)
                    self.tableWidget.setItem(row, 0, QTableWidgetItem(i[0]))
                    self.tableWidget.setItem(row, 1, QTableWidgetItem(i[1]))
                    self.tableWidget.setItem(row, 2, QTableWidgetItem(i[2]))
                    self.tableWidget.setItem(row, 3, QTableWidgetItem(i[3]))
                    self.tableWidget.setItem(row, 4, QTableWidgetItem(i[4]))

                elif (str(i[3]) == find_data):
                    row = self.tableWidget.rowCount()
                    self.tableWidget.insertRow(row)
                    self.tableWidget.setItem(row, 0, QTableWidgetItem(i[0]))
                    self.tableWidget.setItem(row, 1, QTableWidgetItem(i[1]))
                    self.tableWidget.setItem(row, 2, QTableWidgetItem(i[2]))
                    self.tableWidget.setItem(row, 3, QTableWidgetItem(i[3]))
                    self.tableWidget.setItem(row, 4, QTableWidgetItem(i[4]))

                elif (str(i[4]) == find_data):
                    row = self.tableWidget.rowCount()
                    self.tableWidget.insertRow(row)
                    self.tableWidget.setItem(row, 0, QTableWidgetItem(i[0]))
                    self.tableWidget.setItem(row, 1, QTableWidgetItem(i[1]))
                    self.tableWidget.setItem(row, 2, QTableWidgetItem(i[2]))
                    self.tableWidget.setItem(row, 3, QTableWidgetItem(i[3]))
                    self.tableWidget.setItem(row, 4, QTableWidgetItem(i[4]))
        self.label.setText(str(self.tableWidget.rowCount()) + "개")
        f.close()
    else:
        QMessageBox.about(self, "error", "No Choice Column")
```

2) getidx

```
def getidx(f):
    line='0'
    check=csv.reader(f)
    idx='0'
    for i in check:
        if(str(i).split(',')[0]!='['):
            idx=str(i).split(',')[0]
            idx=str(i).split('[')[1]
            idx=str(i).split('\'')[1]
    return str(idx)
```

다음 함수는 csv 파일에 데이터를 이어서 저장할 때 No 값을 알기 위해서 csv 파일로부터 최종 No 값을 가져와 주는 역할을 진행하는 함수입니다.

3) BackGround

```
class Dummy:
    def write(self,s):
        pass
#fork
if os.fork():
    os._exit(0)

os.setpgrp()    #make solo group
os.umask(0)
#stdin, stdout, stderr garbage setting is don't influenced console
sys.stdin.close()
sys.stdout=Dummy()
sys.stderr=Dummy()
#daemon process execute run funtion
thread=threading.Thread(target=run,args=(i,))
thread.start()
```

다음 소스코드는 리눅스에서 백그라운드로 계속 돌기 위한 소스코드 입니다. Fork 는 자식을 만드는 함수로 os 안의 fork 함수를 통해서 만들었습니다. Fork 를 만드는 이유는 보통 프로세스를 부모프로세스가 제거하는데, fork 로 자식을 만든 후 부모 프로세스를 죽여서 fork 프로세스의 부모역할을 init 프로세스가 맡게 하는 방법으로 프로세스를 백그라운드에서 돌리기 위함입니다. Stdin,stdout,stderr 등은 백그라운드 프로세스에서 필요하지 않기 때문에 의미 없는 값을 집어 넣거나 close 를 진행 했습니다. 또한 이렇게 생성된 thread 가 run 함수를 실행하게 설정 후 thread 를 돌림으로써 백그라운드에서 run 함수를 실행할 수 있게 소스코드를 작성하였습니다.

Support.py 소스코는 csv 파일을 경로를 입력 받아 조회하는 방법으로 Main.py 소스코드 내용 일부와 거의 같습니다.

4) run

```
def run(self):
    while True:
        r=dns.resolver.Resolver()
        r.nameservers=['8.8.8.8'] #Dns server is google
        DnsServer='8.8.8.8'
        try:
            fulldata=r.query('flOckflOck.info').response  #check Domain
        except dns.resolver.Timeout or dns.resolver.NoAnswer:
            r.nameservers=['208.67.222.222']    #Dns server is opendns
            DnsServer='208.67.222.222'
            fulldata = r.query('flOckflOck.info').response  # check Domain
        f = open("collect.csv", 'a')  # if not exist file then we needs create file
        f.close()
        f=open("collect.csv",'r')
        strdata=str(fulldata)


        #split & extraction Domain and IP datas
        tok=strdata.split(';ANSWER')[1]
        tok=tok.split(';AUTHORITY')[0]
        Entire=[]   #Domain + IP
        DnsName=[]  #Domain
        Ipdata=[]   #IP
        j=0
        DnsName.append((tok.split(' ')[0]).split('₩n')[1])
        while(j<len(tok.split(' '))-1):
            Entire.append(tok.split(' ')[j+4])  #IP + Domain
            j+=4
        for i in Entire:
            if(i.split('₩n')[1]!=None):
                DnsName.append(i.split('₩n')[1])
                Ipdata.append(i.split('₩n')[0])
            else:
                Ipdata.append(i)

        #Write File
        idx=getidx(f)
        f.close()
        f = open("collect.csv", 'a')  # output file
        wr=csv.writer(f)
        t=datetime.datetime.utcnow()    #GMT Time
        ktime=t+datetime.timedelta(hours=9) #GMT+9
        for i in range(0,len(Ipdata)):
            idx=int(idx)+1
            wr.writerow([str(idx),ktime.strftime("%Y-%m-%d %H:%M:%S"),DnsName[i],Ipdata[i],DnsServer])
           # f.write(str(idx)+' '+ktime.strftime("%Y-%m-%d %H:%M:%S")+' '+DnsName[i]+' '+Ipdata[i]+' '+DnsServer+"₩n")


        #Execuete every 1 hours
        f.close()
        time.sleep(3600)
```

다음 함수는 실질적으로 dns 질의를 진행하는 핵심 함수입니다. 먼저 DNS 서버로는 8.8.8.8 과 208.67.222.222 를 사용했습니다. 다음과 같이 try 문을 통해서 DNS 서버와의 통신이 타임 아웃이 발생하거나 서버측 응답이 없는 경우 8.8.8.8->208.67.222.222 로 DNS 서버 변경을 진행했습니다. 추가적으로 output csv 파일은 collect.csv 로 main.py 와 같은 경로에 생성했습니다. dnspython 을 통한 질의로 IP 정보를 가져올 수 있고 해당 데이터를 짤라서 필요한 부분만 수집했습니다. 다음으로 시간 정보의 경우 현재 UTC(GMT 와 차이가 실질적으로 나지 않기 때문에)를 받아와 +9 를 해줌으로써 GMT+9 시간을 구했습니다.

4. 동작원리

우선 Main.py 프로그램은 실행 시 fork 를 통한 방법으로 백그라운드에서 프로세스가 실행될 수 있게 하고 동시에 GUI 프로그램을 실행시켜 조회 기능이 가능하도록 구현했습니다.

DNS 질의의 경우 처음에는 8.8.8.8 서버에 DNS 질의를 진행하도록 하였고 서버의 응답이 없거나 타임아웃이 발생된 경우 두 번째 DNS 서버인 208.67.222.222 서버에 질의를 진행하도록 예외처리문을 이용해서 작성했습니다. DNS 서버에서 날라온 데이터를 바탕으로 IP 값을 추출해내고 DNS 서버, IP, 도메인 주소를 기입하고 No 값의 경우에는 csv 파일이 존재하는 경우 해당 파일의 최근 No 를 얻어와서 No 값이 이어 질 수 있게 작업하였습니다. 마지막으로 GMT+9 의 시간 값의 경우 현재 GMT 시간값을 얻고 +9 를 해서 직접 구했하고 데이터를 CSV 파일형식으로 만들어 CSV 파일로 출력하였습니다.

Support.py 의 경우 DNS 조회 프로그램으로 실행 시 csv 파일 경로를 입력받아 csv read 를 통해 데이터를 테이블에 출력하고 데이터의 컬럼에 따라 값을 조회할 수 있게끔 GUI 를 만들었습니다. 추가적으로 가장 아래에는 테이블에 row 수를 카운트 해서 몇 개의 데이터가 출력되었는지 한 눈 에 볼 수 있게 개수를 출력하였습니다.

II 정리 및 요약

DNS 질의를 통한 과제로 주어진 시나리오에 맞게 소스코드 작업을 진행했습니다.
리눅스 백그라운드 프로세스 설정 및 DNS 질의를 통해 IP 를 얻어오는 과정을 진행하면서 dnspython 모듈과 얻은 데이터의 가공, GUI 프로그래밍을 통해 프로그램을 개발 했습니다.

기술적인 부담을 크게 요구하지 않았지만 무엇보다 멘토님께서 실제 사건에 사용하셨던 프로그램을 재밌는 시나리오를 통해 작업함으로써 재밌는 경험이 되는 과제였었습니다.

감사합니다!!!