

공인인증서 파밍사건에 대한 분석 및 파싱 코드 구현 보고서

|  |
| --- |
| Digital Forensic Track 2019-01-13 |
| 김태우 |

INDEX

[1. 개요 3](#_Toc535249244)

[1.1 사건 경위 3](#_Toc535249245)

[1.2 파싱 코드 구현을 위한 환경 3](#_Toc535249246)

[2. 코드 구현 4](#_Toc535249247)

[2.1 인증서 내역 파싱을 위한 TXT파일 제작 4](#_Toc535249248)

[2.2 DB 구성 5](#_Toc535249249)

[2.2.1 SQLite 구성 5](#_Toc535249250)

[2.2.2 MySQL 구성 6](#_Toc535249251)

[2.3 전체 코드의 구성 7](#_Toc535249252)

[2.3.1 Main.java 8](#_Toc535249253)

[2.3.2 GetFileFromURL.java 8](#_Toc535249254)

[2.3.3 ParsingAndSaveSQL.java 12](#_Toc535249255)

[2.3.4 ParsingAndInsertMysql.java 17](#_Toc535249256)

[3.1 인증서 다운로드 19](#_Toc535249257)

[3.2 인증서 파싱 및 DB삽입 21](#_Toc535249258)

[3.2.1 SQLite DB Insert 21](#_Toc535249259)

[3.2.2 MySQL DB Insert 24](#_Toc535249260)

[4. DB 분석 25](#_Toc535249261)

[5. DB데이터 CSV로 추출 27](#_Toc535249262)

[5.1 SQLite DB의 CSV추출 27](#_Toc535249263)

[5.2 MySQL DB의 CSV추출 31](#_Toc535249264)

[6. 결론 및 후기 33](#_Toc535249265)

# 1. 개요

## 1.1 사건 경위

피의자는 한국인(해외거주민 포함)을 상대로 공인인증서를 탈취하는 악성코드를 유포하였습니다.

악성코드에 감염된 피해자의 PC에 저장된 공인인증서는 [IP주소].zip 파일의 형태로 해외 서버(http://fl0ckfl0ck.work/cert)에 업로드되어 있습니다.

다행히 디렉터리 리스팅 취약점으로 인하여 공인인증서 들은 노출되어 있습니다.

빠른 시일내에 폐기처분을 해야합니다.

- 팀장 지시사항

1. 피해자의 일련번호 - 피해시각(서버에 피해자의 공인인증서가 업로드된 시간) - 피해자의 이름 - 은행명 - 계좌번호 - IP주소 - 피해자의 현재 소재지(국가만)를 데이터베이스(mysql)에 저장하시오.

2. 은행별 유출된 공인인증서의 갯수를 계산하시오.

## 1.2 파싱 코드 구현을 위한 환경

표 1. 코드 구현 PC 환경표

|  |  |
| --- | --- |
| OS | Windows 10 Pro x64 17134 |
| Ram | 8GB |
| SSD | 256GB |
| HDD | 2TB |

표 2. 개발 언어 및 사용 도구

|  |  |
| --- | --- |
| 사용언어 | Java |
| IDE | Oxygen.3a Release (4.7.3a) |
| DB | SQLite 5.3 |
| DB to CSV | SQliteStudio 3.2.1 |

# 2. 코드 구현

## 2.1 인증서 내역 파싱을 위한 TXT파일 제작

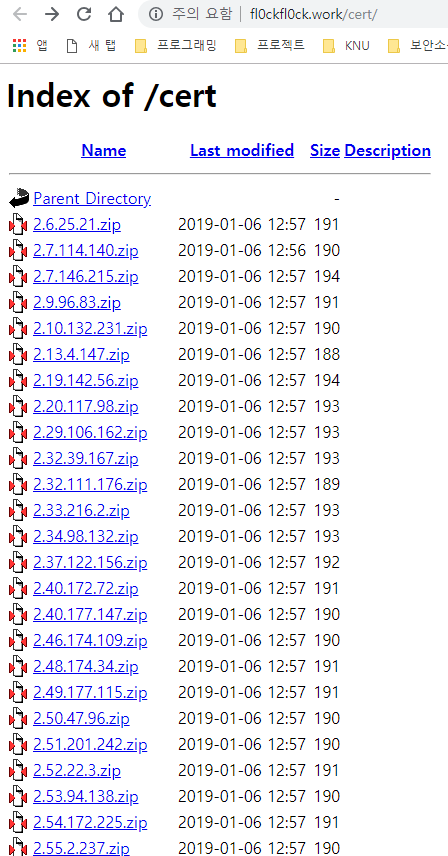


그림 1. 인증서 파싱을 위한 TXT파일 제작

해커의 사이트에 가서 TXT값들을 모두 추출하여 IPList.txt라는 파일을 제작하였다.

해당 파일은 [그림 2]와 같이 IP주소 시각 사이즈로 구성되어 있다.

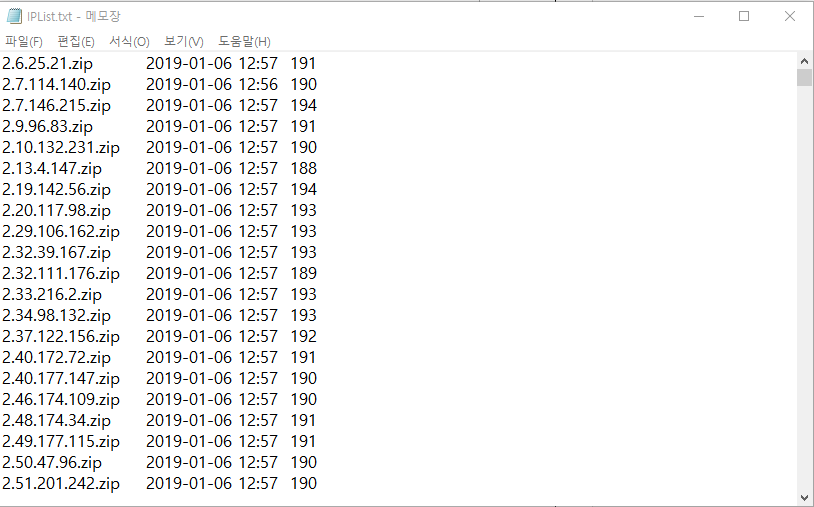


그림 2. IPList.txt 파일의 구성

## 2.2 DB 구성

### 2.2.1 SQLite 구성

DB명 certDB

다음 DDL을 통해 테이블을 구성하였다.

CREATE TABLE `cert` (

`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

`time` DATE,

`name` TEXT,

'bank' TEXT,

'account' TEXT,

'ip' BIGINT,

'country' TEXT

);

생성된 테이블은 [그림 3]과 같다.

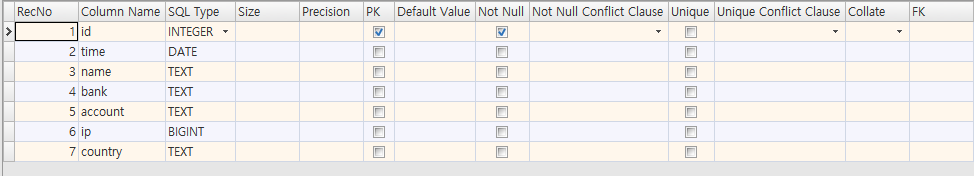


그림 3. certDB의 cert테이블 구조

## 2.2.2 MySQL 구성

DB명 certdb

다음 DDL을 통해 테이블을 구성하였다.

CREATE TABLE cert (

id INT(15) NOT NULL PRIMARY KEY,

time DATETIME,

name VARCHAR(10),

bank VARCHAR(10),

account VARCHAR(30),

ip VARCHAR(20),

country VARCHAR(5)

);

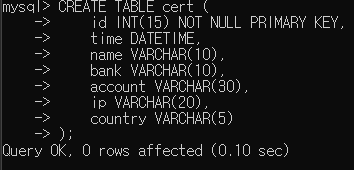


그림 . MySQL cert table의 생성

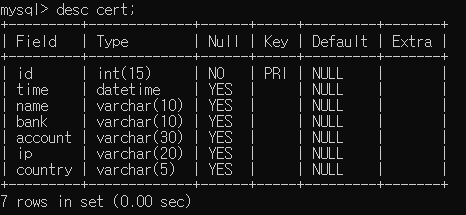


그림 . MySQL cert 테이블의 구성

## 2.3 전체 코드의 구성

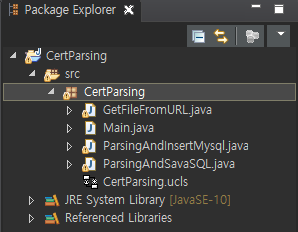


그림 6. 전체 코드의 구성

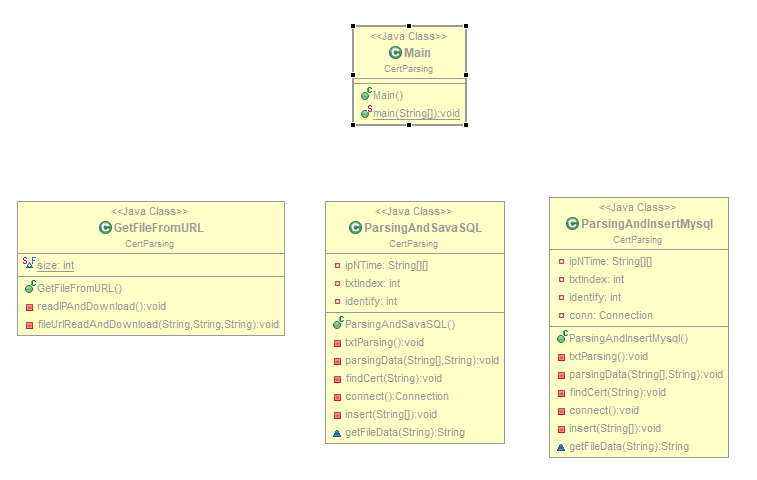


그림 7. 전체 코드의 UML

### 2.3.1 Main.java

메인 함수

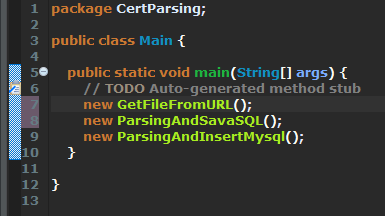


그림 8. 메인 함수

### 2.3.2 GetFileFromURL.java

파싱한 TXT파일을 기반으로 압축된 인증서 파일 다운로드를 진행하는 코드

#### - 멤버 변수 및 생성자



그림 9. GetFileFromURL클래스의 멤버변수와 생성자

#### - readIPANDDownload 메서드

저장된 TXT파일을 파싱하여 파일을 다운받는 메서드

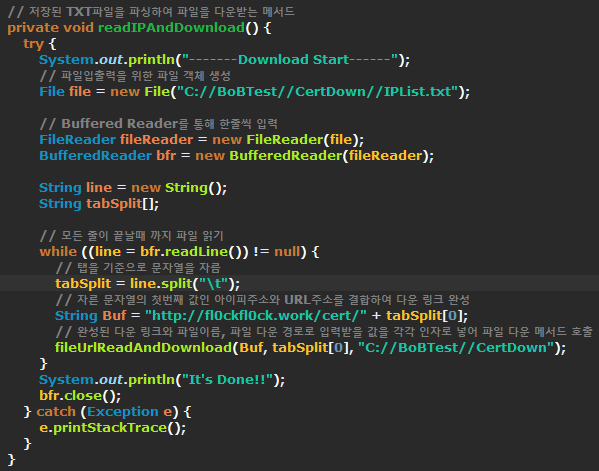


그림 10. readIPAndDownload 메서드

#### - fileUrlReadAndDownload 메서드

파일의 URL주소와 파일이름, 다운로드 경로를 받아 파일을 다운받는 메서드

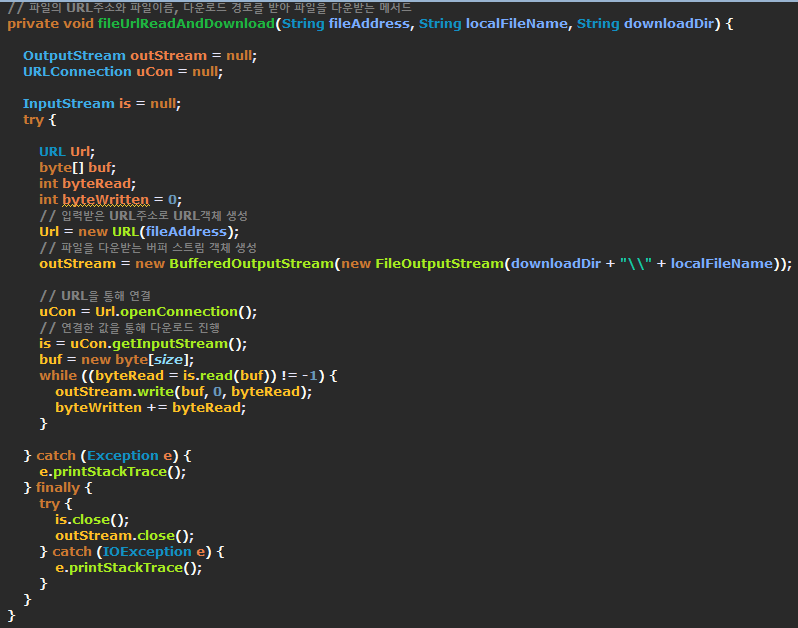


그림 11. fileUrlReadAndDownload 메서드

### 2.3.3 ParsingAndSaveSQL.java

다운받은 데이터를 파싱하고 DB에 저장하는 클래스

#### - 멤버 변수 및 생성자

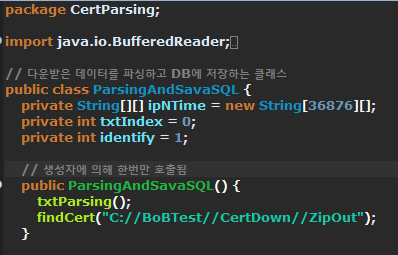


그림 12. ParsingAndSaveSQL 클래스의 멤버 변수와 생성자

#### - txtParsing 메서드

주어진 txt파일을 읽고 정규식을 이용하여 IP를 뽑아내고, 시간정보를 뽑아내는 메서드

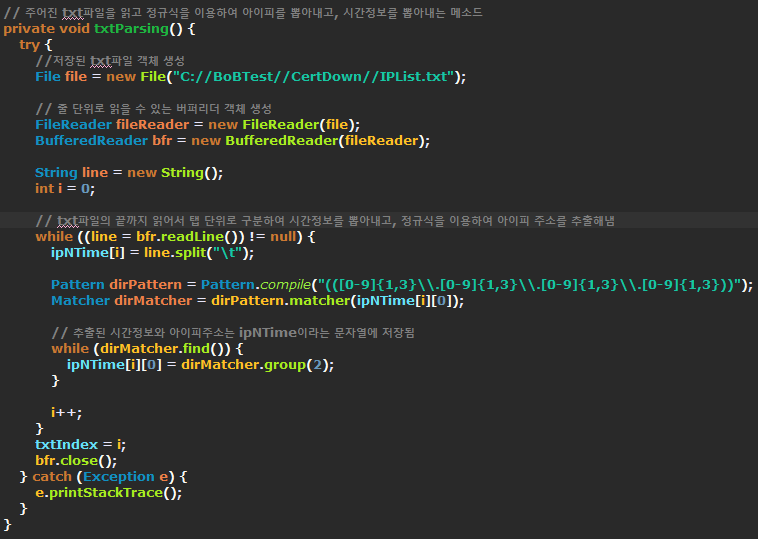


그림 13. txtParsing 메서드

#### - parsingData 메서드

파싱한 txt파일정보(IP, 시각)와 파싱한 cert데이터를 가지고 DB에 삽입

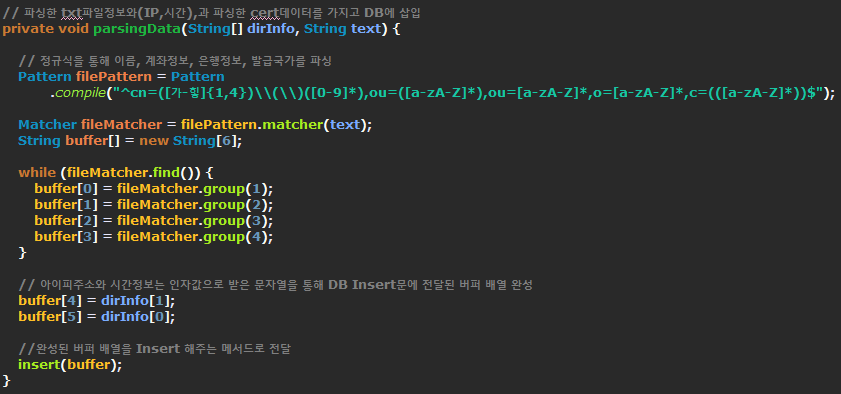


그림 14. parsingData 메서드

#### - findCert 메서드

파싱된 아이피와 시간값을 통합하여 DB에 삽입하는 메서드

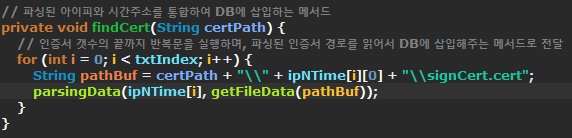


그림 15. findCert 메서드

#### - connect 메서드

DB의 연결을 성립을 위한 메서드

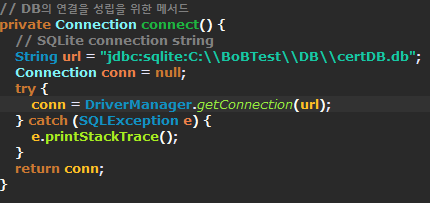


그림 16. connect 메서드

#### - insert 메서드

DB에 값을 삽입하기 위한 메서드

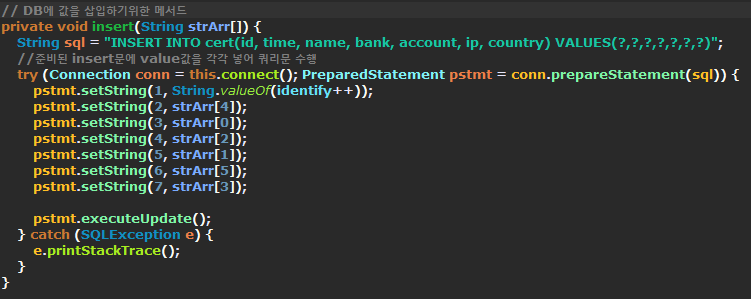


그림 17. insert 메서드

#### - getFileData 메서드

인증서 파일을 읽어 들이기 위한 메서드

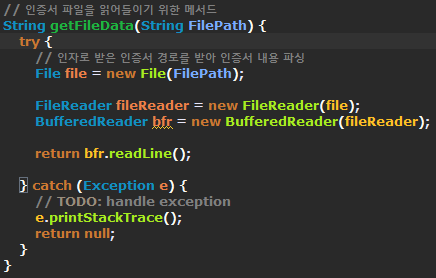


그림 18. getFileData 메서드

### 2.3.4 ParsingAndInsertMysql.java

다운받은 데이터를 파싱하고 MySQL DB에 저장하는 클래스

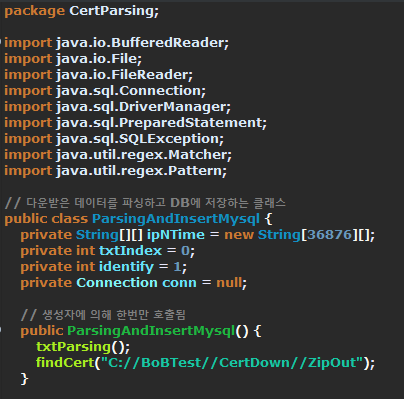


그림 . ParsingAndInsertMysql 클래스의 멤버 변수와 생성자

- connect 메서드

MySQL과 연결을 성립하기 위한 메서드

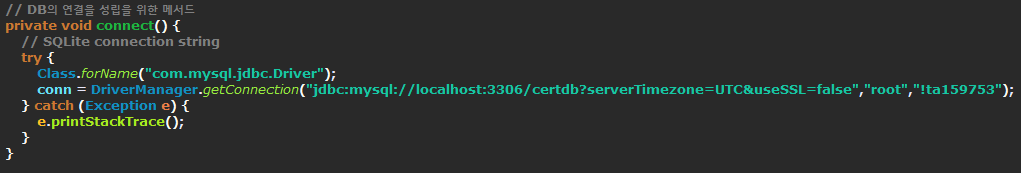


그림 . connect 메서드

connect메서드 이외에는 ParsingAndSaveSQL 클래스의 메서드들과 같으므로 생략한다.

구현을 하며, 정규식에서 이름, 계좌정보, 은행정보, 발급국가를 파싱하는 부분에서, 이름이 5글자 인 데이터를 확인하여 [그림 22]와 같이 읽어오는 데이터 값을 한글자에서 다섯글자까지로 수정하였다.

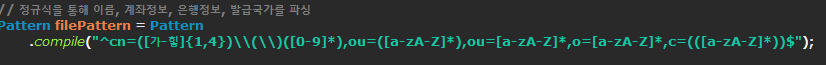


그림 . 이름에 대한 이슈 확인

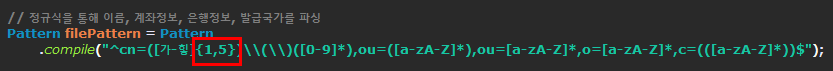


그림 . 이름에 대한 이슈 해결

3. 코드 실행

## 3.1 인증서 다운로드

GetFileFromURL클래스의 생성자를 수행함으로써 자동으로 인증서가 다운로드 된다.

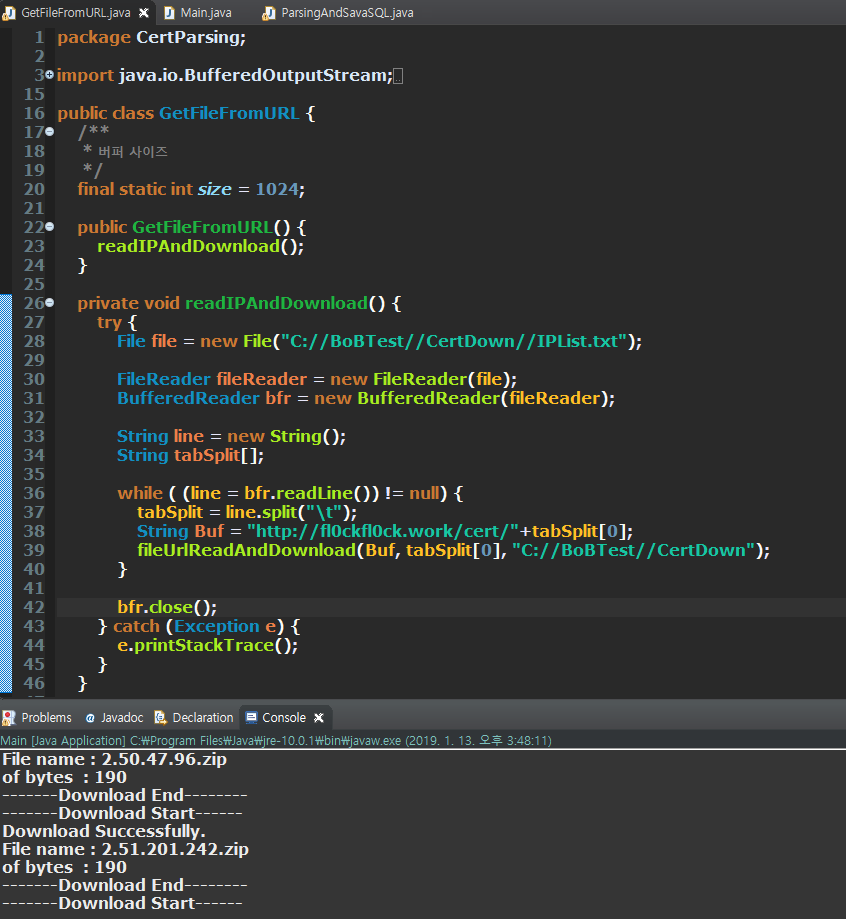


그림 23. 해커의 사이트에서 자동으로 인증서 파일을 다운받는 장면

다운로드가 완료된 인증서는 [그림 17]과 같다.

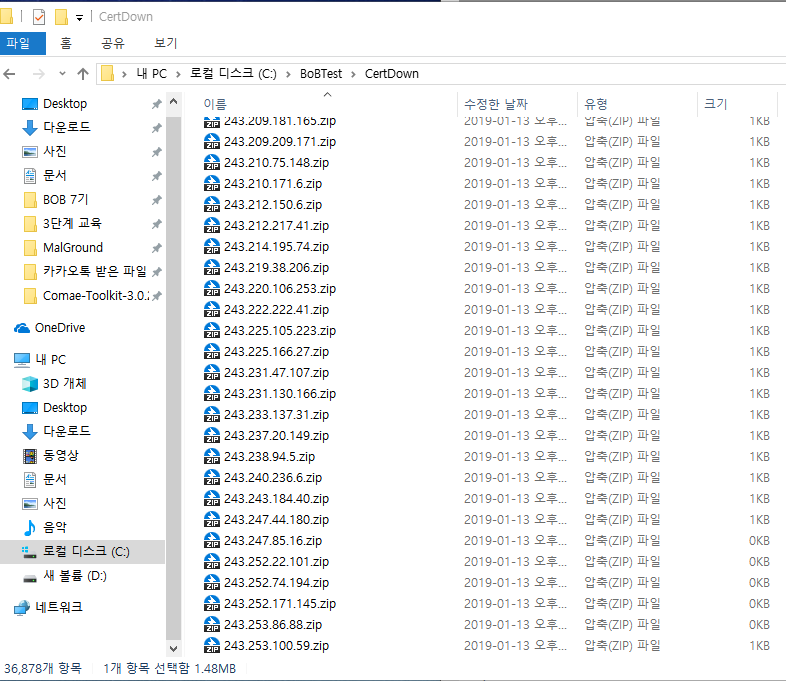


그림 24. 다운로드가 완료된 인증서들

총 36,876개의 인증서가 다운로드 되었다.

## 3.2 인증서 파싱 및 DB삽입

[그림 18, 19]과 같이 자동으로 인증서를 파싱하며 DB에 Insert하는 것을 확인할 수 있다.

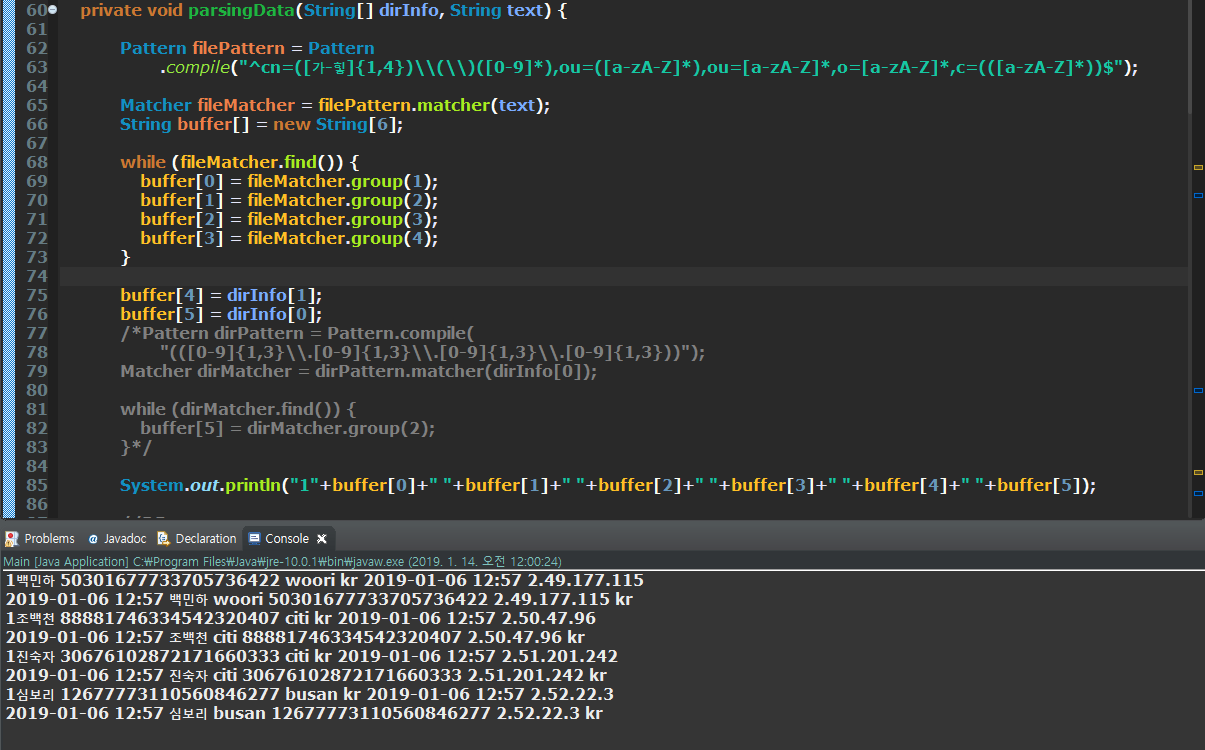


그림 25. 인증서 파싱 및 DB 삽입

### 3.2.1 SQLite DB Insert

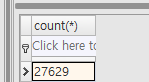


그림 26. DB에 삽입되며 데이터 개수의 증가

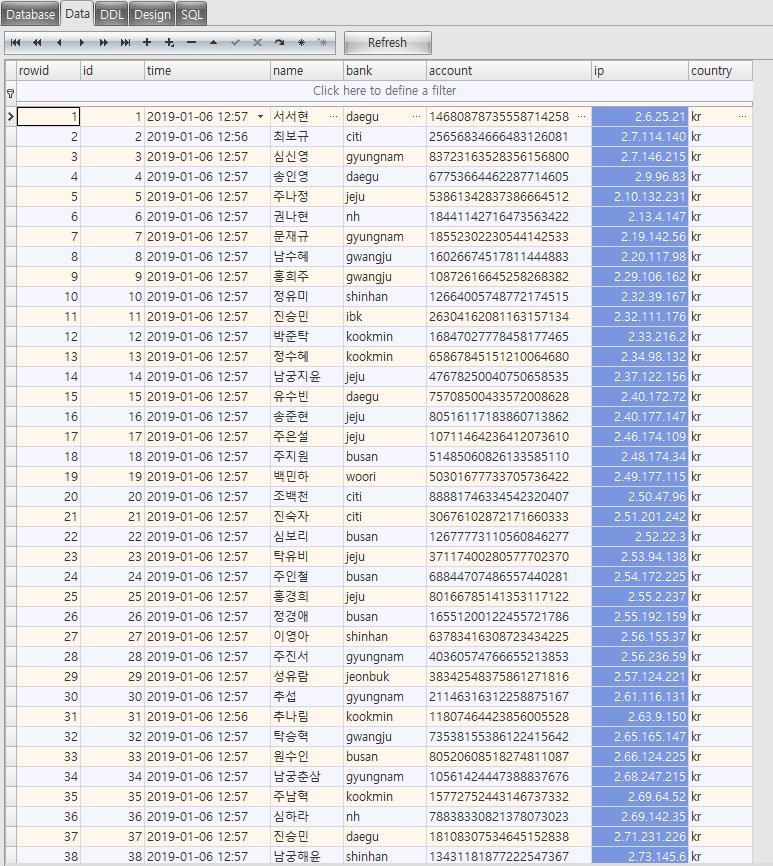


그림 27. 삽입이 완료된 DB

select count(\*) from cert 명령을 통해 DB의 총 데이터 개수가 36,876개로 모든 인증서가 삽입된 것을 확인 할 수 있다.

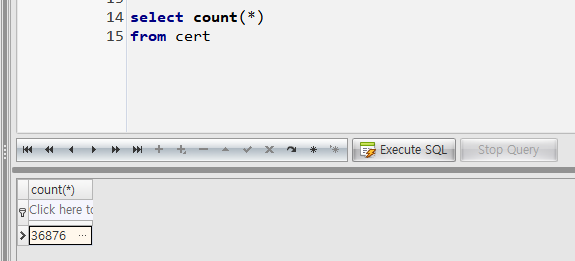


그림 28. DB의 총 데이터의 개수

### 3.2.2 MySQL DB Insert

[그림 29]와 같이 MySQL에서 DB에 Insert되며 데이터가 증가하고, 36,876개의 데이터가 삽입된 것을 확인할 수 있다.

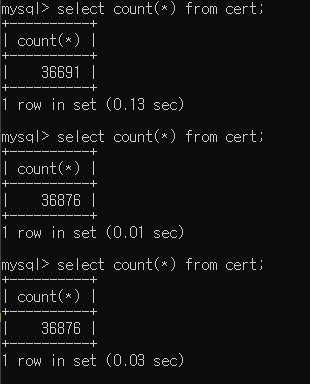


그림 . MySQL DB 데이터의 증가

또한 [그림 30]과 같이 데이터가 정상적으로 Insert된 것을 확인할 수 있다.



그림 . MySQL에 삽입된 데이터

# 4. DB 분석

은행별 유출된 공인인증서의 개수 파악은 다음 쿼리문을 통해 파악할 수 있다

SELECT bank,count(\*) as 'count'

FROM cert

GORUP BY bank

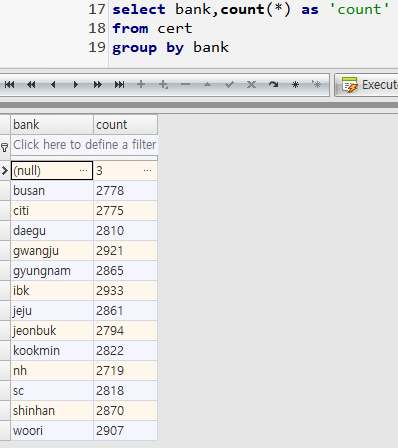


그림 31. SQLite 각 은행별 유출된 공인인증서 개수

SQLite에서는 이름이 5글자인 데이터를 처리하지 않아 3건의 null이 발생하였다.



그림 . MySQL 각 은행별 유출된 공인인증서 개수

표 3. 각 은행별 유출된 공인인증서 개수

|  |  |
| --- | --- |
| bank | count |
| ibk | 2933 |
| gwangju | 2921 |
| woori | 2909 |
| shinhan | 2870 |
| gyungnam | 2865 |
| jeju | 2861 |
| kookmin | 2822 |
| sc | 2818 |
| daegu | 2810 |
| jeonbuk | 2794 |
| busan | 2778 |
| citi | 2775 |
| nh | 2720 |

가장 많은 공인인증서가 유출된 은행은 ibk은행인 것을 확인 할 수 있다.

# 5. DB데이터 CSV로 추출

## 5.1 SQLite DB의 CSV추출

해당 과정은 SQliteStudio 도구로 쉽게 진행 할 수 있다.

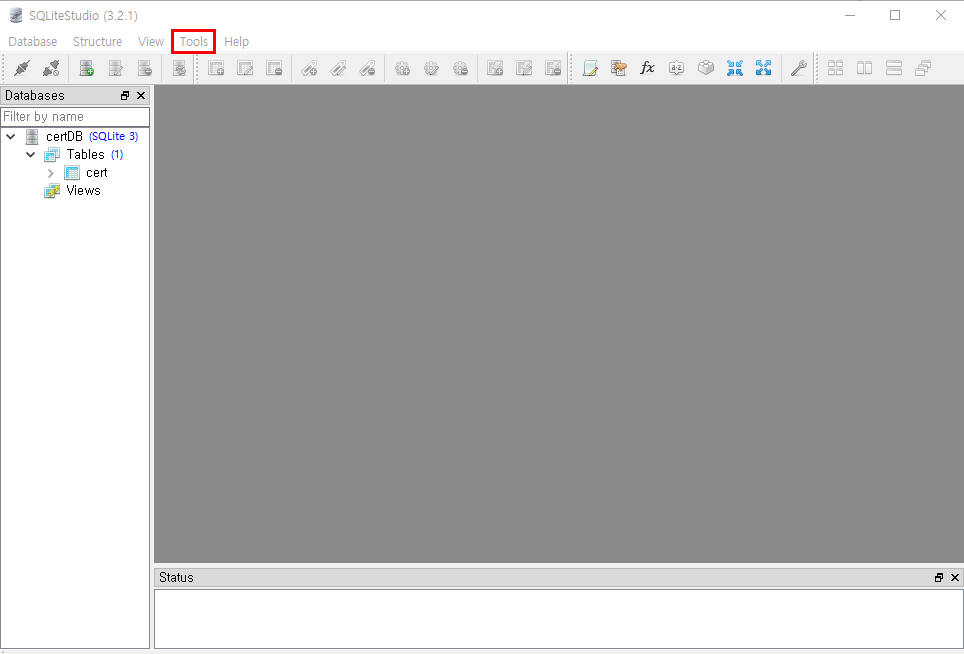


그림 33. SQliteStudio의 실행

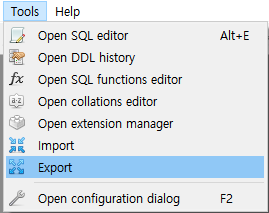


그림 34. SQliteStudio의 추출기능

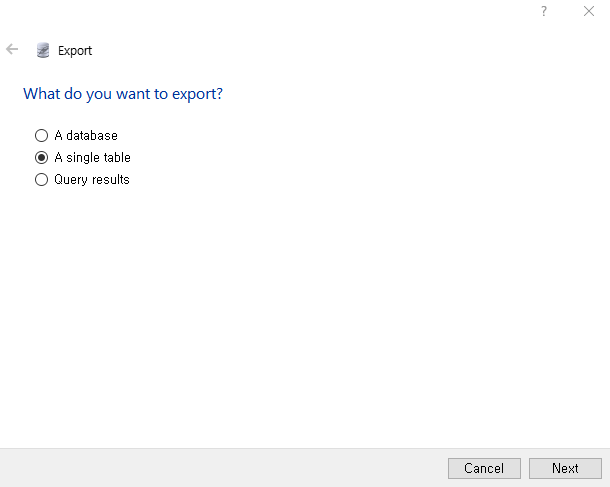


그림 35. 추출할 대상의 선택

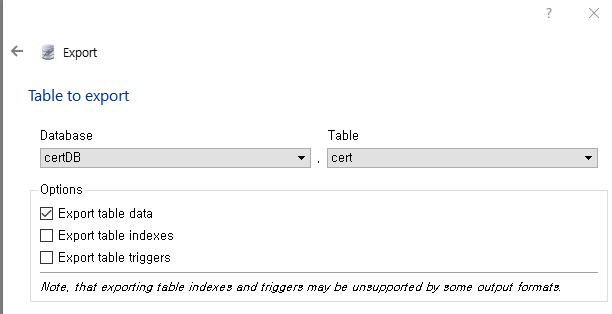


그림 36. 추출 테이블 선택

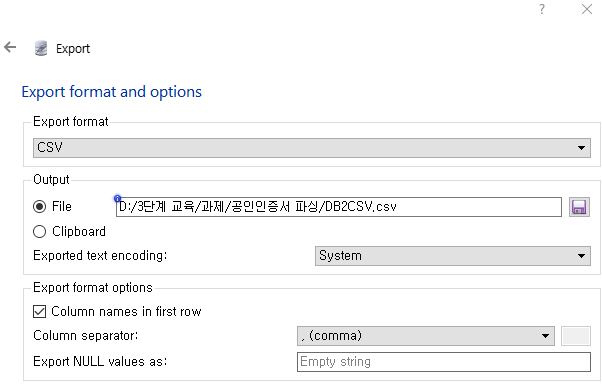


그림 37. 선택한 테이블의 추출

해당 도구의 실행 후 Tools의 Export기능으로 CSV파일로 추출하였다.

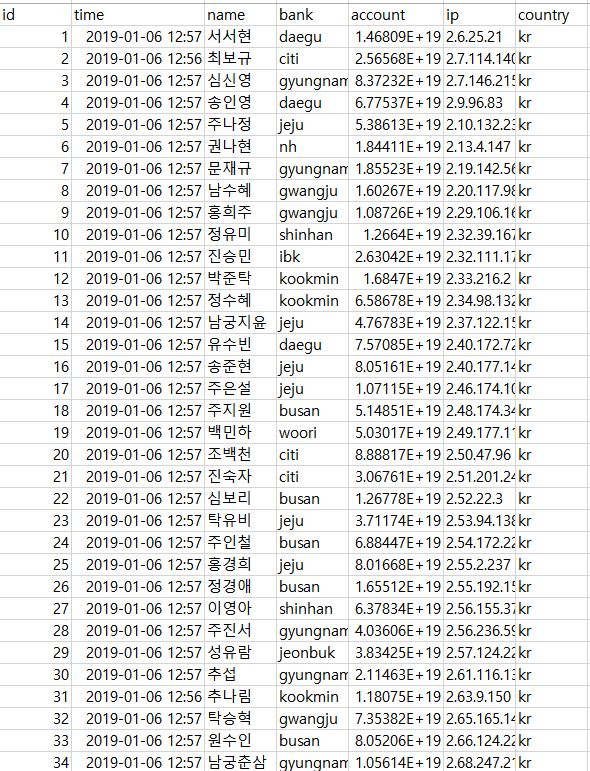


그림 38. SQLite DB에서 추출된 CSV파일의 일부

## 5.2 MySQL DB의 CSV추출

해당 과정은 다음과 같은 쿼리문을 통해 진행 하였다.

SELECT \* FROM cert

INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/db2csv.csv'

CHARACTER SET euckr

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

ESCAPED BY '\\'

LINES TERMINATED BY '\n'

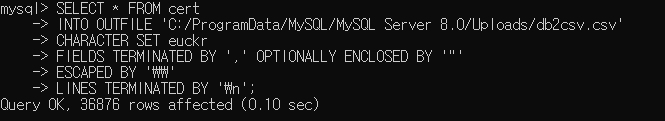


그림 . MySQL데이터의 CSV추출을 위한 쿼리문 수행

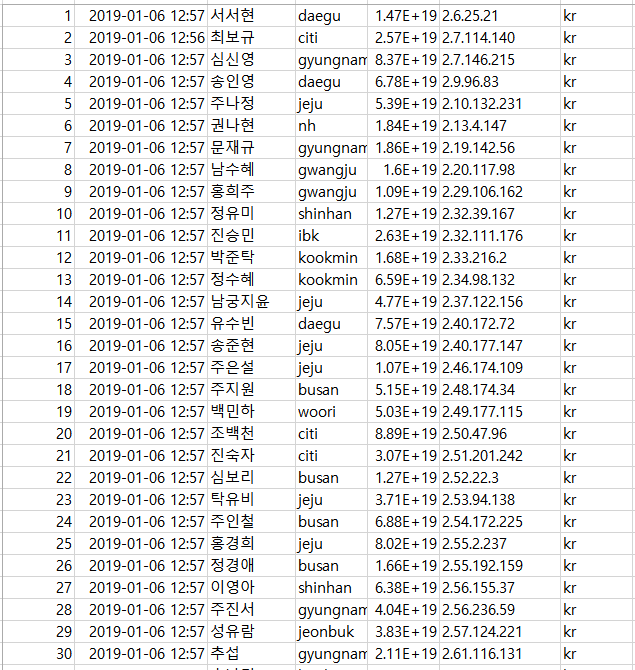


그림 . MySQL DB에서 추출된 CSV파일의 일부

# 6. 결론 및 후기

수사관의 관점에서 피해상황을 최소화 하고자 코드를 제작하여 자동으로 인증서를 폐기하기위해 DB에 삽입하는 코드를 구현하였다. 이전에 진행하였던 과제였으나, 중간에 데이터가 완벽히 들어가지 않는등 다양한 이슈가 발견하였으나 다시한번 코드를 구현하며 모두 해결할 수 있었다.

또한 SQLite에서 이름 데이터에 대하여 한글자에서 네글자 까지만 처리하게 하였으나, MySQL에서는 한글자에서 다섯글자까지 처리하게 하여 모든 데이터가 정상적으로 Insert될 수 있도록 조치하였다.