



공인인증서 파밍사건에 대한 분석 및
파싱 코드 구현 보고서

Digital Forensic Track 2019-01-18

김태우

INDEX

1. 개요.....	3
1.1 사건 경위.....	3
1.2 파싱 코드 구현을 위한 환경.....	3
2. 코드 구현.....	5
2.1 MySQL 구성.....	5
2.2 전체 코드의 구성.....	6
2.2.1 Main.java.....	7
2.2.2 CrawlerFromSuspectURL.java.....	7
2.2.3 GetFileFromURLWithProxy.java.....	9
2.2.4 ParsingAndInesertMysql.java.....	14
3. 코드 실행.....	20
3.1 용의자의 사이트에서 HTML 소스 파싱 및 TXT 파일로 출력.....	20
3.2 인증서 다운로드.....	21
3.2 인증서 파싱 및 DB 삽입.....	22
4. DB 분석.....	23
5. DB 데이터 CSV 로 추출.....	25
6. 결론 및 후기.....	26

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.

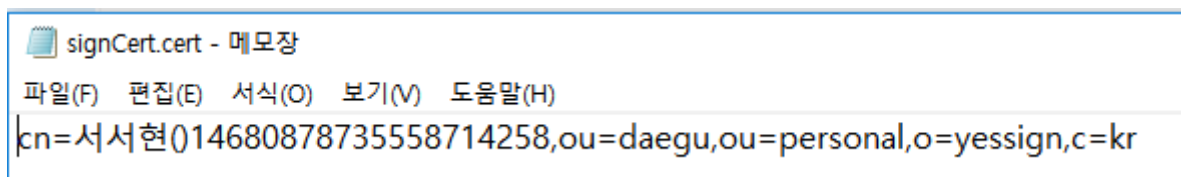


그림 1. 공인인증서의 내용

- 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	MySQL 8.0.13

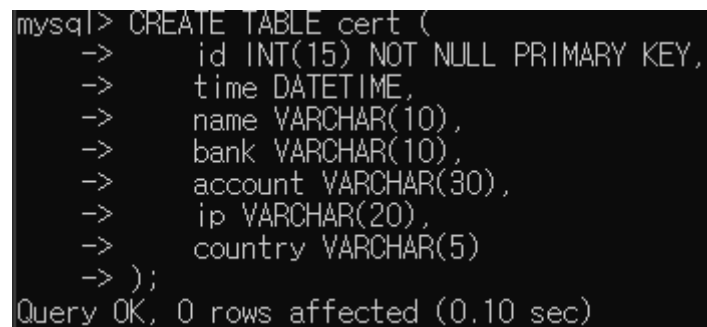
2. 코드 구현

2.1 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)  
);
```



```
mysql> 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)  
-> );  
Query OK, 0 rows affected (0.10 sec)
```

그림 2. MySQL cert table의 생성

```
mysql> desc cert;
```

Field	Type	Null	Key	Default	Extra
id	int(15)	NO	PRI	NULL	
time	datetime	YES		NULL	
name	varchar(10)	YES		NULL	
bank	varchar(10)	YES		NULL	
account	varchar(30)	YES		NULL	
ip	varchar(20)	YES		NULL	
country	varchar(5)	YES		NULL	

7 rows in set (0.00 sec)

그림 3. MySQL cert 테이블의 구성

2.2 전체 코드의 구성

다음 [그림 4]은 전체 코드의 구성을 나타내는 UML이다.

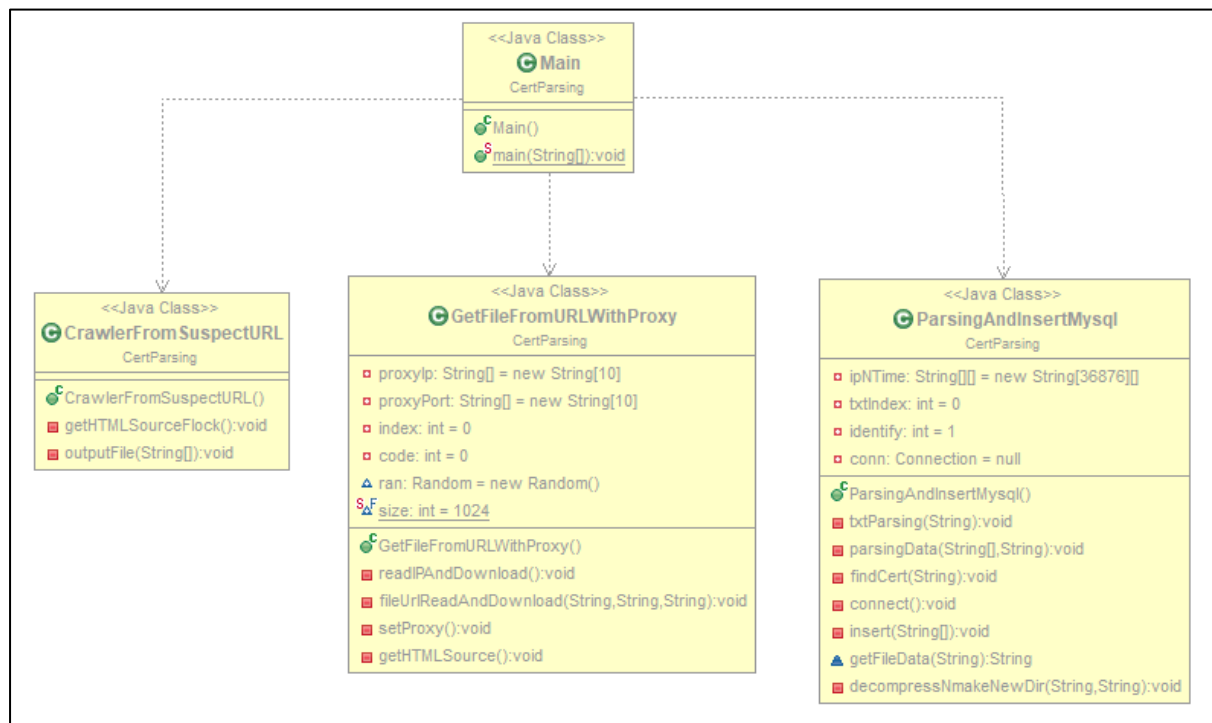


그림 4. 전체 코드의 UML

2.2.1 Main.java

메인 함수

```
package CertParsing;

public class Main {

    public static void main(String[] args) {

        //1. CrawlerFromSuspectURL을 이용하여 용의자 사이트에서 HTML소스 부른뒤 파싱하여 TXT파일로 저장
        new CrawlerFromSuspectURL();
        //2. 저장한 TXT파일을 읽어들이며 프록시를 통해 IP를 숨기고 인증서 다운로드
        new GetFileFromURLWithProxy();
        // 3. 다운로드 한 파일들의 압축을 풀고, 인증서 값을 파싱하여 Mysql DB에 저장
        new ParsingAndInsertMysql();

    }

}
```

그림 5. 메인 함수

2.2.2 CrawlerFromSuspectURL.java

용의자의 사이트에서 HTML소스를 부른 뒤 파싱하여 TXT파일로 저장하는 클래스

- 생성자

```
package CertParsing;

import java.io.BufferedWriter;

public class CrawlerFromSuspectURL {

    public CrawlerFromSuspectURL() {
        getHTMLSourceFlock();
    }

}
```

그림 6. CrawlerFromSuspectURL클래스의 생성자

- getHTMLSourceFlock 메서드

용의자의 사이트로부터 HTML소스를 불러서 파일 정보와 시간 데이터만 파싱하는 메서드

```
private void getHTMLSourceFlock() {
    try {
        // 용의자의 사이트주소
        String urlStr = "http://f10ckf10ck.work/cert";

        // get메소드를 통해 용의자의 HTML 소스 불러옴
        Document doc = Jsoup.connect(urlStr).header("Accept-Encoding", "gzip, deflate")
            .userAgent("Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0")
            .maxBodySize(0).timeout(600000).get();

        // 1. 웹 사이트에서 <tr>값을 선택한다
        Elements articles = doc.select("tr");
        // 2. <tr>값의 갯수를 구한다.
        int size = articles.size();

        // 파일 출력을 위해 문자 배열을 임시로 선언한다.
        String valueBuffer[] = new String[(size - 5)];

        int countBuff = 0;
        int index = 0;

        // 앞 3 뒤 1
        System.out.println(size);

        for(Element article : articles) {
            // System.out.println(article);

            // 웹 사이트에서 f12로 확인된 값을 통해 셀렉터 값을 넣어주고, 순수 값만 뽑아낸다.
            if(countBuff < 3) {
                countBuff++;
                continue;
            }
            if(countBuff >= (size-2)) {
                break;
            }
        }
    }
}
```

그림 7. getHTMLSourceFlock 메서드 1

```
valueBuffer[index++] = article.select("td:nth-child(2)").text()+"\t"+article.select("td:nth-child(3)").text()+"\r\n";
countBuff++;

}

outputFile(valueBuffer);

} catch (Exception e) {
    // TODO: handle exception
    e.printStackTrace();
}
}
```

그림 8. getHTMLSourceFlock 메서드 2

- outputFile 메서드

파싱한 HTML소스를 문자열로 받아 파일로 저장하는 메서드


```
private void outputFile(String[] HTMLSource) {
    try {
        // 인자로 받은 인증서 경로를 받아 인증서 내용 파싱
        File file = new File("C:\\BoBTest\\Proxy\\IPList.txt");

        FileWriter fileWriter = new FileWriter(file);
        BufferedWriter bfr = new BufferedWriter(fileWriter);

        for (int i = 0; i < HTMLSource.length; i++) {
            bfr.write(HTMLSource[i]);
        }

        bfr.close();
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }
}
```

그림 9. outputFile 메서드

2.2.3 GetFileFromURLWithProxy.java

저장한 TXT파일을 읽어, 다운받을 파일을 파악하고, 프록시를 설정하여 다운받는 수사관 PC의 IP정보를 숨겨 인증서를 다운로드 받는 클래스

- 멤버 변수 및 생성자

```
package CertParsing;

import java.io.BufferedOutputStream;

public class GetFileFromURLWithProxy {
    private String proxyIp[] = new String[10];
    private String proxyPort[] = new String[10];
    private int index = 0;
    private int code = 0;

    Random ran = new Random();
    final static int size = 1024;

    // 생성자를 통해 한번만 실행됨
    public GetFileFromURLWithProxy() {
        readIPAndDownload();
    }
}
```

그림 10. GetFileFromURLWithProxy 클래스의 멤버변수와 생성자

- readIPAndDownload 메서드

저장된 TXT파일을 파싱하여 파일을 다운받는 메서드

```
// 저장된 TXT파일을 파싱하여 파일을 다운받는 메서드
private void readIPAndDownload() {
    try {
        System.out.println("-----Download Start-----");
        // 파일입출력을 위한 파일 객체 생성
        File file = new File("C://BoBTest//CertDown//IPList.txt");

        // Buffered Reader를 통해 한줄씩 입력
        FileReader fileReader = new FileReader(file);
        BufferedReader bfr = new BufferedReader(fileReader);

        String line = new String();
        String tabSplit[];

        // 모든 줄이 끝날때 까지 파일 읽기
        while ((line = bfr.readLine()) != null) {
            // 탭을 기준으로 문자열을 자름
            tabSplit = line.split("\t");
            // 자른 문자열의 첫번째 값인 아이피주소와 URL주소를 결합하여 다운 링크 완성
            String Buf = "http://flockflock.work/cert/" + tabSplit[0];
            // 완성된 다운 링크와 파일이름, 파일 다운 경로로 입력받을 값을 각각 인자로 넣어 파일 다운 메서드 호출
            fileUrlReadAndDownload(Buf, tabSplit[0], "C://BoBTest//CertDown");
        }
        System.out.println("It's Done!!");
        bfr.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

그림 11. readIPAndDownload 메서드

- fileUrlReadAndDownload 메서드

파일의 URL주소와 파일이름, 다운로드 경로를 받아 파일을 다운받는 메서드

```
// 파일의 URL주소와 파일이름, 다운로드 경로를 받아 파일을 다운받는 메서드
private void fileUrlReadAndDownload(String fileAddress, String localFileName, String downloadDir) {

    OutputStream outputStream = null;
    URLConnection uConn = null;

    InputStream is = null;
    HttpURLConnection huc = null;

    try {
        URL url;
        byte[] buf;
        int bytesRead;
        int byteWritten = 0;
        // 입력받은 URL주소로 URL객체 생성
        url = new URL(fileAddress);

        // 프록시를 setProxy 메서드에 의해 설정되는데, 응답코드가 200일때만 파일을 다운받음,
        // 응답코드 200이 아닐시, 프록시를 재설정함
        do {
            if (code != 200) {
                setProxy();
            }
            huc = (HttpURLConnection) url.openConnection();
            huc.setRequestMethod("GET"); // OR huc.setRequestMethod ("HEAD");
            huc.connect();
            code = huc.getResponseCode();
            System.out.println(code);
        }
        while (code != 200);

        //프록시 설정이 된 후 다운로드 진행
        outputStream = new BufferedOutputStream(new FileOutputStream(downloadDir + "\\\" + localFileName));

        is = huc.getInputStream();
        buf = new byte[size];
        while ((byteRead = is.read(buf)) != -1) {
```

그림 12. fileUrlReadAndDownload 메서드 1

```
        outputStream.write(buf, 0, byteRead);
        byteWritten += byteRead;
    }

    catch (Exception e) {
        e.printStackTrace();
    }
    finally {
        try {
            is.close();
            outputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

그림 13. fileUrlReadAndDownload 메서드 2

- setProxy 메서드

무료 프록시 사이트를 파싱하여 프록시를 설정하는 메서드

```
// 프록시를 설정하는 메서드
private void setProxy() {
    // https://free-proxy-list.net/ 해당 주소에서 무료 프록시 주소를 불러옴
    if(index <= 10) {
        //getHTMLSource메소드를 통해 무료 프록시 주소 10개를 파싱해옴
        getHTMLSource();
        index = 0;
    }
    System.out.println("changing proxy : "+index);

    // 사이트에서 파싱한 프록시 정보를 설정함
    try {
        System.setProperty("http.proxyHost", proxyIp[index]);
        System.setProperty("http.proxyPort", proxyPort[index]);
        index++;
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }
}
```

그림 14. setProxy 메서드

- getHTMLSource 메서드

무료 프록시 사이트인 <https://free-proxy-list.net/> 에서 IP주소와 Port주소를 파싱하여 배열에 저장하는 메서드

```
// Get Free Proxy information from https://free-proxy-list.net/
private void getHTMLSource() {
    int i = 0;
    try {
        String urlStr = "https://free-proxy-list.net/";

        Document doc = Jsoup.connect(urlStr).get();

        // 정규식을 이용하여 IP주소와 Port주소를 파싱함
        Pattern ipPattern = Pattern.compile("[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}");
        Pattern portPattern = Pattern.compile(">[0-9]{1,5}<");

        Matcher ipMatcher = ipPattern.matcher(doc.toString());
        Matcher portMatcher = portPattern.matcher(doc.toString());

        // 파싱한 값들을 각각 IP와 Port 문자열 배열에 넣어줌
        while (ipMatcher.find() && (i < 10)) {
            proxyIp[i++] = ipMatcher.group();
        }

        i = 0;
        while (portMatcher.find() && (i < 10)) {
            proxyPort[i] = portMatcher.group();
            proxyPort[i] = proxyPort[i].substring(1, proxyPort[i].length()-1);
            i++;
        }

    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }
}
```

그림 15. getHTMLSource 메서드

2.2.4 ParsingAndInesertMysql.java

다운받은 데이터를 파싱하고 DB에 저장하는 클래스

- 멤버 변수 및 생성자

```
package CertParsing;

import java.io.BufferedReader;

// 다운받은 데이터를 파싱하고 DB에 저장하는 클래스
public class ParsingAndInsertMysql {
    private String[][] ipNTime = new String[36876][];
    private int txtIndex = 0;
    private int identify = 1;
    private Connection conn = null;

    // 생성자에 의해 한번만 호출됨
    public ParsingAndInsertMysql() {
        txtParsing("C://BoBTest//CertDown");
        findCert("C://BoBTest//CertDown//ZipOut");
    }
}
```

그림 16. ParsingAndInesertMysql 클래스의 멤버 변수 및 생성자

- txtParsing 메서드

주어진 txt파일을 읽고 정규식을 이용하여 IP와 시간정보를 뽑아내는 메서드

```
// 주어진 txt파일을 읽고 정규식을 이용하여 아이피를 뽑아내고, 시간정보를 뽑아내는 메소드
private void txtParsing(String Path) {
    try {
        System.out.println("---txt Parsing & Unzip Files ---");
        //저장된 txt파일 객체 생성
        File file = new File(Path+"//IPList.txt");

        // 줄 단위로 읽을 수 있는 버퍼리더 객체 생성
        FileReader fileReader = new FileReader(file);
        BufferedReader bfr = new BufferedReader(fileReader);

        String line = new String();
        int i = 0;

        // txt파일의 끝까지 읽어서 탭 단위로 구분하여 시간정보를 뽑아내고, 정규식을 이용하여 아이피 주소를 추출해냄
        while ((line = bfr.readLine()) != null) {
            ipNTime[i] = line.split("\t");
            // 해당 파일의 정보와, 파일의 압축을 풀 경로를 decompressNmakeNewDir메서드로 보내줌
            decompressNmakeNewDir(Path+"\\"+ipNTime[i][0], Path+"\ZipOut");

            Pattern dirPattern = Pattern.compile("([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3})");
            Matcher dirMatcher = dirPattern.matcher(ipNTime[i][0]);

            // 추출된 시간정보와 아이피주소는 ipNTime이라는 문자열에 저장됨
            while (dirMatcher.find()) {
                ipNTime[i][0] = dirMatcher.group(2);
            }

            i++;
        }
        txtIndex = i;
        bfr.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

그림 17. txtParsing 메서드

- parsingData 메서드

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

```
// 파싱한 txt파일정보와(IP, 시간)과 파싱한 cert데이터를 가지고 DB에 삽입
private void parsingData(String[] dirInfo, String text) {

    // 정규식을 통해 이름, 계좌정보, 은행정보, 발급국가를 파싱
    Pattern filePattern = Pattern
        .compile("^cn=([가-힣]{1,5})\\(\\(\\)[0-9]*),ou=([a-zA-Z]*),ou=[a-zA-Z]*,o=[a-zA-Z]*,c=([a-zA-Z]*)$");

    Matcher fileMatcher = filePattern.matcher(text);
    String buffer[] = new String[6];

    while (fileMatcher.find()) {
        buffer[0] = fileMatcher.group(1);
        buffer[1] = fileMatcher.group(2);
        buffer[2] = fileMatcher.group(3);
        buffer[3] = fileMatcher.group(4);
    }

    // 아이피주소와 시간정보는 인자값으로 받은 문자열을 통해 DB Insert문에 전달된 버퍼 배열 완성
    buffer[4] = dirInfo[1];
    buffer[5] = dirInfo[0];

    // 완성된 버퍼 배열을 Insert 해주는 메서드로 전달
    insert(buffer);
}

// 파싱된 아이피와 시간주소를 통합하여 DB에 삽입하는 메서드
private void findCert(String certPath) {
    System.out.println("---insert value into DB ---");
    // 인증서 갯수의 끝까지 반복문을 실행하며, 파싱된 인증서 경로를 읽어서 DB에 삽입해주는 메서드로 전달
    connect();
    for (int i = 0; i < txtIndex; i++) {
        String pathBuf = certPath + "\\\" + ipNTime[i][0] + "\\signCert.cert";
        parsingData(ipNTime[i], getFileData(pathBuf));
    }
}
```

그림 18. parsingData 메서드

- findCert 메서드

파싱된 IP와 시간정보를 통합하여 DB에 삽입하는 메서드

```
// 파싱된 아이피와 시간주소를 통합하여 DB에 삽입하는 메서드
private void findCert(String certPath) {
    System.out.println("---insert value into DB ---");
    // 인증서 갯수의 끝까지 반복문을 실행하며, 파싱된 인증서 경로를 읽어서 DB에 삽입해주는 메서드로 전달
    connect();
    for (int i = 0; i < txtIndex; i++) {
        String pathBuf = certPath + "\\\" + ipNTime[i][0] + "\\signCert.cert";
        parsingData(ipNTime[i], getFileData(pathBuf));
    }
}
```

그림 19. findCert 메서드

- connect 메서드

DB의 연결을 성립하기 위한 메서드

```
// DB의 연결을 성립하기 위한 메서드
private void connect() {
    // SQLite connection string
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/certdb?serverTimezone=UTC&useSSL=false",
            "root", "1a159753");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

그림 20. connect 메서드

- insert 메서드

DB에 값을 삽입하기 위한 메서드

```
// DB에 값을 삽입하기 위한 메서드
private void insert(String strArr[]) {
    String sql = "INSERT INTO cert(id, time, name, bank, account, ip, country) VALUES(?,?,?,?,?,?,?)";
    // 준비된 insert문에 value값을 각각 넣어 쿼리문 수행
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, String.valueOf(identify++));
        pstmt.setString(2, strArr[4]);
        pstmt.setString(3, strArr[0]);
        pstmt.setString(4, strArr[2]);
        pstmt.setString(5, strArr[1]);
        pstmt.setString(6, strArr[5]);
        pstmt.setString(7, strArr[3]);

        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

그림 21. insert 메서드

- getFileData 메서드

인증서 파일을 읽어들이기 위한 메서드

```
// 인증서 파일을 읽어들이기 위한 메서드
String getFileData(String FilePath) {
    try {
        // 인자로 받은 인증서 경로를 받아 인증서 내용 파싱
        File file = new File(FilePath);

        FileReader fileReader = new FileReader(file);
        BufferedReader bfr = new BufferedReader(fileReader);

        return bfr.readLine();
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
        return null;
    }
}
```

그림 22. getFileData 메서드

- decompressNmakeNewDir 메서드

파일명을 받아 새로운 디렉터리를 생성하고, 해당 디렉터리에 압축을 푸는 메서드

```
private void decompressNmakeNewDir(String zipFileName, String outPutPath) {
    // 인자로 받은 파일명을 통해 File객체 생성
    File zipFile = new File(zipFileName);
    // 압축을 푼 파일의 각각 디렉터리를 생성하기 위해 임시 문자열 배열
    String nameBuf = zipFile.getName();
    // 저장할 디렉터리와, 파일명을 통해 새로운 디렉터리 경로로 파일 객체 생성
    File dirFile = new File(outPutPath+"\\\\"+nameBuf.substring(0, nameBuf.length()-4));

    //파일 압축 풀기를 위한 객체들 생성
    FileInputStream fis = null;
    FileOutputStream fos = null;
    ZipInputStream zis = null;
    ZipEntry zipentry = null;
    int size = 0;
    byte[] buffer = new byte[256];

    try {
        // 압축된 파일의 디렉터리 생성
        dirFile.mkdir();

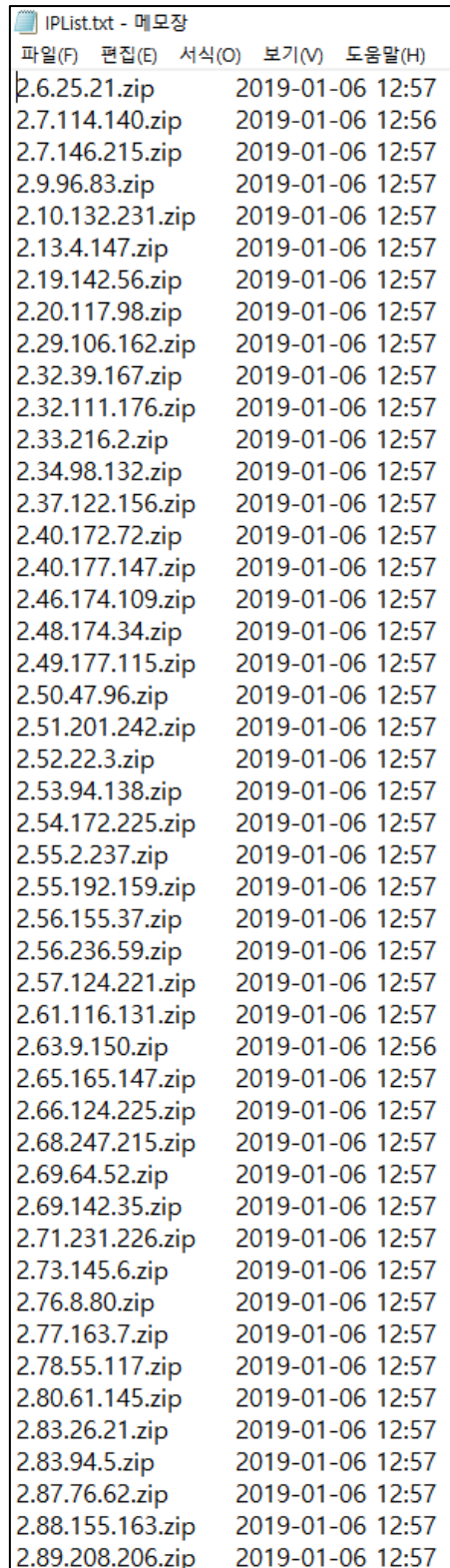
        fis = new FileInputStream(zipFile);
        zis = new ZipInputStream(fis);
        fos = new FileOutputStream(dirFile+"\\signCert.cert");
        // 압축 파일의 끝까지 탐색
        while ((zipentry = zis.getNextEntry()) != null)

            // 탐색한 파일을 파일로 만들
            while ((size = zis.read(buffer)) > 0) {
                // byte로 파일 만들기
                fos.write(buffer, 0, size);
            }
        fis.close();
        zis.close();
        fos.close();
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }
}
```

그림 23. decompressNmakeNewDir 메서드

3. 코드 실행

3.1 용의자의 사이트에서 HTML소스 파싱 및 TXT파일로 출력



파일(F)	편집(E)	서식(O)	보기(V)	도움말(H)
2.6.25.21.zip				2019-01-06 12:57
2.7.114.140.zip				2019-01-06 12:56
2.7.146.215.zip				2019-01-06 12:57
2.9.96.83.zip				2019-01-06 12:57
2.10.132.231.zip				2019-01-06 12:57
2.13.4.147.zip				2019-01-06 12:57
2.19.142.56.zip				2019-01-06 12:57
2.20.117.98.zip				2019-01-06 12:57
2.29.106.162.zip				2019-01-06 12:57
2.32.39.167.zip				2019-01-06 12:57
2.32.111.176.zip				2019-01-06 12:57
2.33.216.2.zip				2019-01-06 12:57
2.34.98.132.zip				2019-01-06 12:57
2.37.122.156.zip				2019-01-06 12:57
2.40.172.72.zip				2019-01-06 12:57
2.40.177.147.zip				2019-01-06 12:57
2.46.174.109.zip				2019-01-06 12:57
2.48.174.34.zip				2019-01-06 12:57
2.49.177.115.zip				2019-01-06 12:57
2.50.47.96.zip				2019-01-06 12:57
2.51.201.242.zip				2019-01-06 12:57
2.52.22.3.zip				2019-01-06 12:57
2.53.94.138.zip				2019-01-06 12:57
2.54.172.225.zip				2019-01-06 12:57
2.55.2.237.zip				2019-01-06 12:57
2.55.192.159.zip				2019-01-06 12:57
2.56.155.37.zip				2019-01-06 12:57
2.56.236.59.zip				2019-01-06 12:57
2.57.124.221.zip				2019-01-06 12:57
2.61.116.131.zip				2019-01-06 12:57
2.63.9.150.zip				2019-01-06 12:56
2.65.165.147.zip				2019-01-06 12:57
2.66.124.225.zip				2019-01-06 12:57
2.68.247.215.zip				2019-01-06 12:57
2.69.64.52.zip				2019-01-06 12:57
2.69.142.35.zip				2019-01-06 12:57
2.71.231.226.zip				2019-01-06 12:57
2.73.145.6.zip				2019-01-06 12:57
2.76.8.80.zip				2019-01-06 12:57
2.77.163.7.zip				2019-01-06 12:57
2.78.55.117.zip				2019-01-06 12:57
2.80.61.145.zip				2019-01-06 12:57
2.83.26.21.zip				2019-01-06 12:57
2.83.94.5.zip				2019-01-06 12:57
2.87.76.62.zip				2019-01-06 12:57
2.88.155.163.zip				2019-01-06 12:57
2.89.208.206.zip				2019-01-06 12:57

그림 24. 용의자의 사이트에서 파싱한 IP주소와 시간값

3.2 인증서 다운로드

GetFileFromURLWithProxy 클래스의 생성자를 수행함으로써 자동으로 인증서가 다운로드 된다.

다운로드가 완료된 인증서는 [그림 28]과 같다.

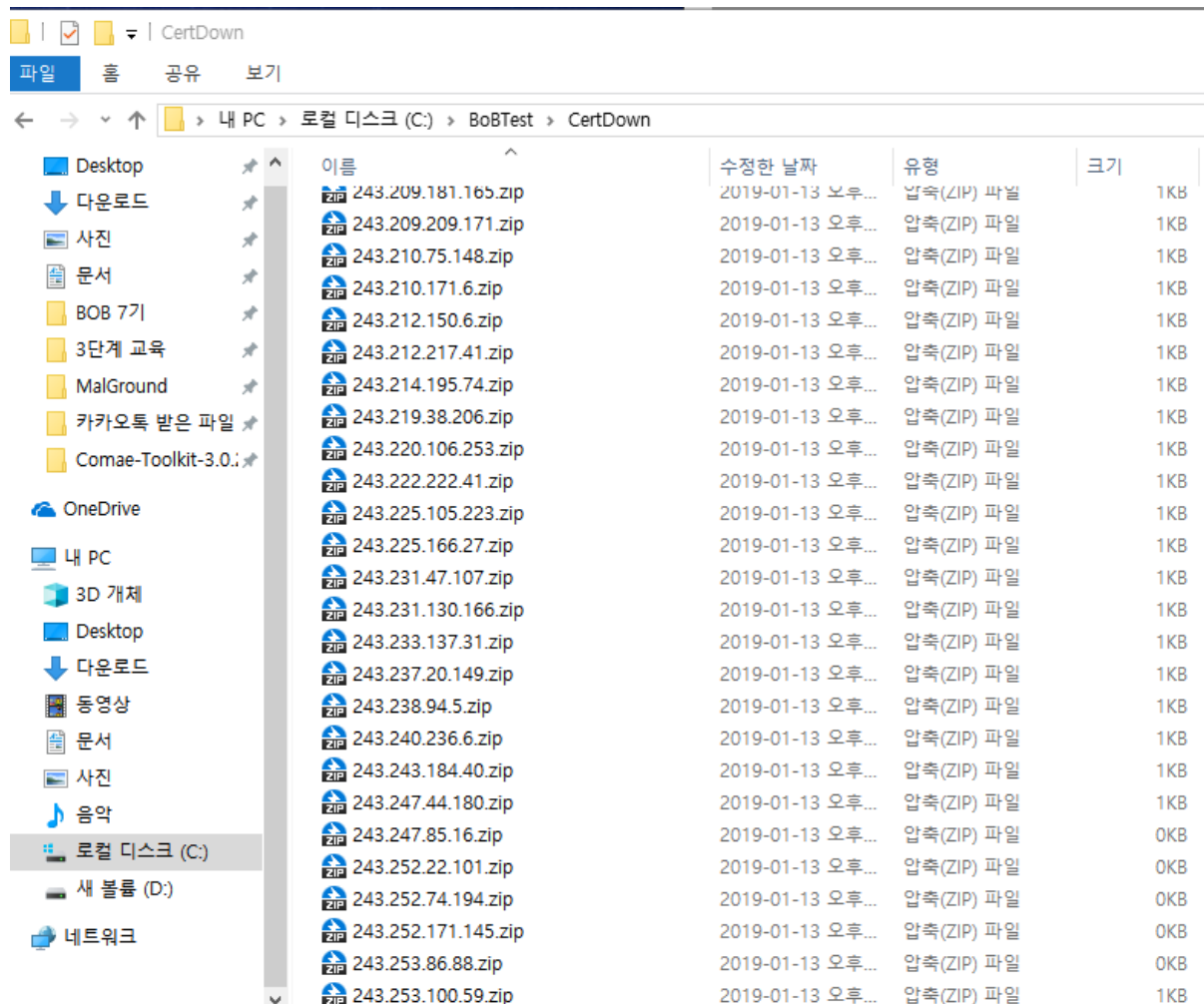


그림 25. 다운로드가 완료된 인증서들

총 36,876개의 인증서가 다운로드 되었다.

3.2 인증서 파싱 및 DB삽입

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

```
mysql> select count(*) from cert;
+-----+
| count(*) |
+-----+
|      36691 |
+-----+
1 row in set (0.13 sec)

mysql> select count(*) from cert;
+-----+
| count(*) |
+-----+
|      36876 |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from cert;
+-----+
| count(*) |
+-----+
|      36876 |
+-----+
1 row in set (0.03 sec)
```

그림 29. MySQL DB 데이터의 증가

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

```
mysql> select * from cert;
```

id	time	name	bank	account	ip	country
1	2019-01-06 12:57:00	서서현	daegu	14680878735558714258	2.6.25.21	kr
2	2019-01-06 12:56:00	최보규	citi	25656834666483126081	2.7.114.140	kr
3	2019-01-06 12:57:00	심신영	gyungnam	83723163528356156800	2.7.146.215	kr
4	2019-01-06 12:57:00	송인영	daegu	67753664462287714605	2.9.96.83	kr
5	2019-01-06 12:57:00	주나정	jeju	53861342837386664512	2.10.132.231	kr
6	2019-01-06 12:57:00	권나현	nh	18441142716473563422	2.13.4.147	kr
7	2019-01-06 12:57:00	문재규	gyungnam	18552302230544142533	2.19.142.56	kr
8	2019-01-06 12:57:00	남수혜	gwangju	16026674517811444883	2.20.117.98	kr
9	2019-01-06 12:57:00	홍희주	gwangju	10872616645258268382	2.29.106.162	kr
10	2019-01-06 12:57:00	정유미	shinhan	12664005748772174515	2.32.39.167	kr
11	2019-01-06 12:57:00	진승민	ibk	26304162081163157134	2.32.111.176	kr
12	2019-01-06 12:57:00	박준혁	kookmin	16847027778458177465	2.33.216.2	kr
13	2019-01-06 12:57:00	정수혜	kookmin	65867845151210064680	2.34.98.132	kr
14	2019-01-06 12:57:00	남궁지윤	jeju	47678250040750658535	2.37.122.156	kr
15	2019-01-06 12:57:00	유수빈	daegu	75708500433572008628	2.40.172.72	kr
16	2019-01-06 12:57:00	송준현	jeju	80516117183860713862	2.40.177.147	kr
17	2019-01-06 12:57:00	주은실	jeju	10711464236412073610	2.46.174.109	kr
18	2019-01-06 12:57:00	주지원	busan	51485060826133585110	2.48.174.34	kr

그림 30. MySQL에 삽입된 데이터

4. DB 분석

은행별 유출된 공인인증서의 개수 파악은 다음 쿼리문을 통해 파악할 수 있다

```
SELECT bank, count(*) as bankCount
```

```
FROM cert
```

```
GROUP BY bank
```

```
ORDER BY bankCount desc;
```

```
mysql> select bank, count(*) as bankCount
-> from cert
-> group by bank
-> order by bankCount desc;
```

bank	bankCount
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

```
13 rows in set (0.03 sec)
```

그림 31. 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

해당 과정은 다음과 같은 쿼리문을 통해 진행 하였다.

```

1 SELECT * FROM cert
2 INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/db2csv.csv'
3 CHARACTER SET euckr
4 FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
5 ESCAPED BY '\\'
6 LINES TERMINATED BY '\n'
7

```

그림 32. DB데이터의 CSV추출을 위한 쿼리문

```

mysql> 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';
Query OK, 36876 rows affected (0.10 sec)

```

그림 33. MySQL데이터의 CSV추출을 위한 쿼리문 수행

1	2019-01-06 12:57	서서현	daegu	1.47E+19	2.6.25.21	kr
2	2019-01-06 12:56	최보규	citi	2.57E+19	2.7.114.140	kr
3	2019-01-06 12:57	심신영	gyungnam	8.37E+19	2.7.146.215	kr
4	2019-01-06 12:57	송인영	daegu	6.78E+19	2.9.96.83	kr
5	2019-01-06 12:57	주나정	jeju	5.39E+19	2.10.132.231	kr
6	2019-01-06 12:57	권나현	nh	1.84E+19	2.13.4.147	kr
7	2019-01-06 12:57	문재규	gyungnam	1.86E+19	2.19.142.56	kr
8	2019-01-06 12:57	남수혜	gwangju	1.6E+19	2.20.117.98	kr
9	2019-01-06 12:57	홍희주	gwangju	1.09E+19	2.29.106.162	kr
10	2019-01-06 12:57	정유미	shinhan	1.27E+19	2.32.39.167	kr
11	2019-01-06 12:57	진승민	ibk	2.63E+19	2.32.111.176	kr
12	2019-01-06 12:57	박준탁	kookmin	1.68E+19	2.33.216.2	kr
13	2019-01-06 12:57	정수혜	kookmin	6.59E+19	2.34.98.132	kr
14	2019-01-06 12:57	남궁지윤	jeju	4.77E+19	2.37.122.156	kr
15	2019-01-06 12:57	유수빈	daegu	7.57E+19	2.40.172.72	kr
16	2019-01-06 12:57	송준현	jeju	8.05E+19	2.40.177.147	kr
17	2019-01-06 12:57	주은설	jeju	1.07E+19	2.46.174.109	kr
18	2019-01-06 12:57	주지원	busan	5.15E+19	2.48.174.34	kr
19	2019-01-06 12:57	백민하	woori	5.03E+19	2.49.177.115	kr
20	2019-01-06 12:57	조백천	citi	8.89E+19	2.50.47.96	kr
21	2019-01-06 12:57	진숙자	citi	3.07E+19	2.51.201.242	kr
22	2019-01-06 12:57	심보리	busan	1.27E+19	2.52.22.3	kr
23	2019-01-06 12:57	탁유비	jeju	3.71E+19	2.53.94.138	kr
24	2019-01-06 12:57	주인철	busan	6.88E+19	2.54.172.225	kr
25	2019-01-06 12:57	홍경희	jeju	8.02E+19	2.55.2.237	kr
26	2019-01-06 12:57	정경애	busan	1.66E+19	2.55.192.159	kr
27	2019-01-06 12:57	이영아	shinhan	6.38E+19	2.56.155.37	kr
28	2019-01-06 12:57	주진서	gyungnam	4.04E+19	2.56.236.59	kr
29	2019-01-06 12:57	성유람	jeonbuk	3.83E+19	2.57.124.221	kr
30	2019-01-06 12:57	추섭	gyungnam	2.11E+19	2.61.116.131	kr

그림 34. MySQL DB에서 추출된 CSV파일의 일부

6. 결론 및 후기

수사관의 관점에서 피해상황을 최소화 하고자 코드를 제작하여 자동으로 인증서를 폐기하기 위해 DB에 삽입하는 코드를 구현하였다. 이전에 진행하였던 과제였으나, 다양한 이슈가 발견하였으나 다시한번 코드를 구현하며 모두 해결할 수 있었다.

해결한 문제 중 이전에 DB로 사용하였던 SQLite에서 이름 데이터에 한글자에서 네글자까지만 처리하게 하여 3개의 튜플에 NULL값이 삽입되었던 문제가 있었다. 해당 에러를 이번 과제에서 파악하여 한글자에서 다섯글자까지 처리하게 하여 모든 데이터가 정상적으로 Insert될 수 있도록 조치하였다.

또한 손으로 직접 IP값을 드래그 하여 TXT파일을 만드는 것을 자동화 하였으며, 압축을 푸는 것 또한 압축 프로그램을 통해 압축을 푸는 방식이 아닌, 코드를 구현하여 압축풀기를 자동화 하였다. 무엇보다, Proxy연결을 설정 및 연결이 확실한 Proxy만 불러와 사용하기 때문에 비교적 안정적으로 수사관의 IP주소를 숨겨 파일을 다운받을 수 있었다.

그리하여, 용의자의 사이트에서 다운받을 파일들을 파싱 및 TXT파일로 출력하고, 출력된 TXT파일을 통해 프록시를 설정하여 다운로드 받은 뒤, 압축을 풀고, 풀어진 각 인증서의 데이터를 파싱하여 DB에 삽입하는 전 과정을 자동화 하였다.