KECCAK을 이용한 자바 난독화 프로그램 개발

한국 멀티미디어학회

2022년 11월 18일

동명대학교 정보보호학과

김재원, 김재형, 김태원, 김재현

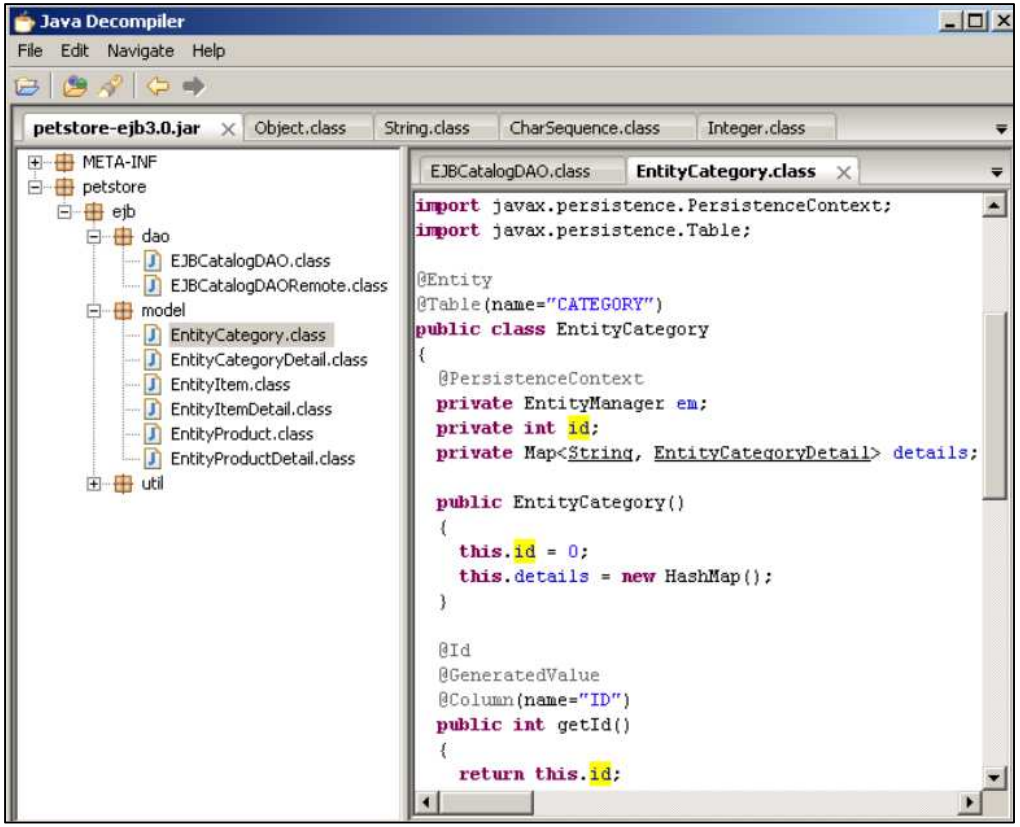
0. 목 차

Global IT Brain

동명대학교

TONGMYONG UNIVERSITY

1. 서 론



출처 : http://sjava.net/2010/01/another-java-decompiler/

- 자바로 개발된 프로그램은 디컴파일러를 통해 **원형의 코드로 복원**된다.
- 코드의 재사용 및 수정을 통한 저작권 침해와 **게임 핵, 크랙 버전 생성** 등으로 인한 피해를 발생시킬 수 있다.
- 본 논문에서는 이러한 문제점을 개선하기 위하여 자바로 작성된 프로그램의 악용사례를 줄이고자 KECCAK을 이용한 코드 난독화 방법을 제안하고 **프로그램을 개발한다.**

2. 관련 동향

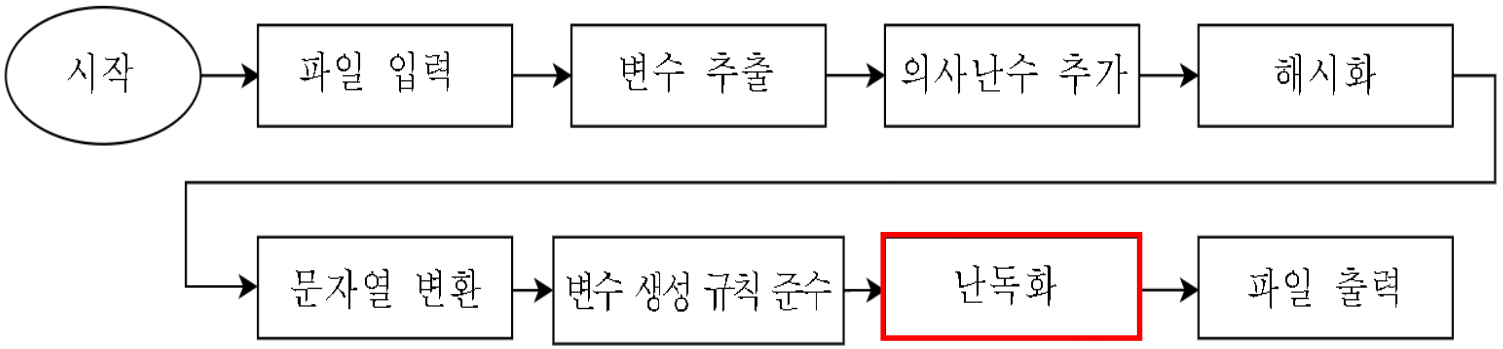
Original source	Converted source
while (true) { (Omission) Class localClass = Class.forName(**"android.telephony.SmsManager"**); (Omission) Method localMethod1 = localClass.getDeclaredMethod (**"sendTextMessage"**, (Class[])localObject); Object[] arrayOfObject = new Object[5]; arrayOfObject[0] = "821012345678"; arrayOfObject[2] = "TEST"; (Omission)	while (true) { (Omission) Class localClass = Class.forName(".**Yjin2.iEBFviut$NCeimUJXZ wsv**"); (Omission) Method localMethod1 = localClass.getDeclaredMethod (**"qPt6SYsDXBMJ 5tj"**, (Class[])localObject); Object[] arrayOfObject = new Object[5]; arrayOfObject[0] = "821012345678"; arrayOfObject[2] = "TEST"; (Omission)

- 참고 -

- 기존 난독화 기법에선 함수명을 암호화 하는 방식을 사용하였다.
- 해당 기법은 변수명을 난독화 하는데 사용 할 경우 **코드의 가독성에 큰 영향이 없다**.
- 또한, 제3자가 정보를 수집한다면 난독화 한 코드를 **복호화 할 가능성 또한 존재한다.**

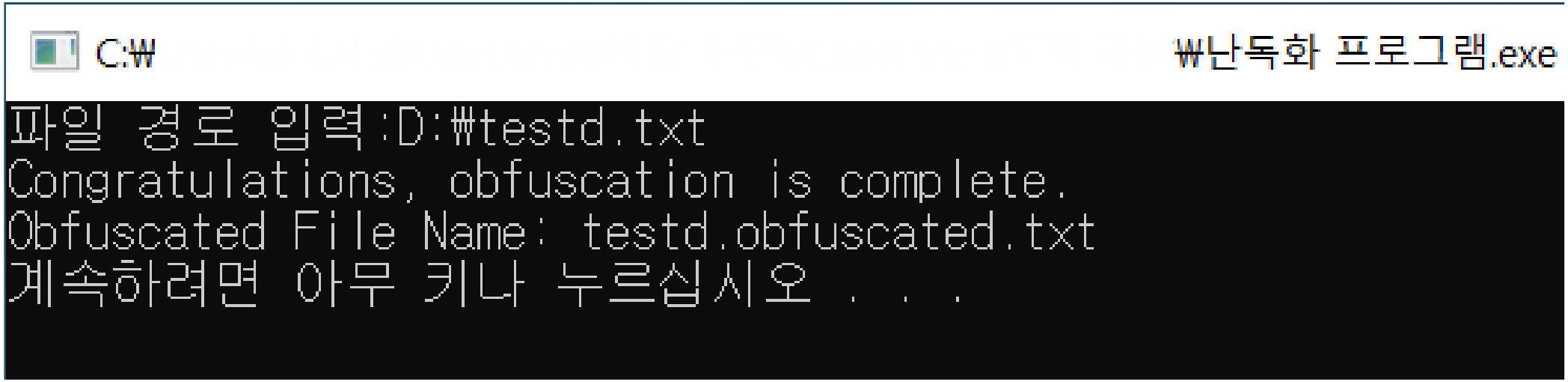
참고문헌 : 이주혁 외 1명, "리플렉션과 문자열 암호화를 이용한 안드로이드 API 난독화 도구", 정보처리학회논문지/컴퓨터 및 통신 시스템 제4권 제1호, 2015. 1. , pp 23

난독화 프로그램 - 시스템 순서도



- 파일내의 코드에서 변수명을 추출함.
- 레인보우 테이블을 이용한 공격을 방지하기 위해 변수명 앞에 의사난수를 추가함.
- 의사난수를 추가한 변수명을 KECCAK알고리즘을 사용하여 난독화 함.

난독화 프로그램 - 프로그램 실행화면



- 프로그램 실행 시 콘솔창 형태로 실행된다.
- **난독화 할 파일은 TXT형식으로 지정**하고 해당 파일의 경로를 입력받는다.
- 난독화가 완료된 파일은 프로그램과 같은 경로에 저장된다.

난독화 프로그램 - 난독화 비교분석

```
import java.util.Scanner;

public class test {
   public static void main(String[] args) {
      String name;
      int age;
      double height;
      String intro;
      String buffer;

      Scanner sc = new Scanner(System.in);
      System.out.println("이름을 입력하세요");
      name = sc.next();
      System.out.println("나이를 입력하세요");
      age = sc.nextInt();
      System.out.println("키를 입력하세요");
      height = sc.nextDouble();
      System.out.println("자기소개를 입력하세요");
      buffer = sc.nextLine();
      intro = sc.nextLine();

      System.out.println("이름은 "+name+"나이는 "+age+",키는 "+height+"입니다.");
      System.out.println(intro);
   }
```

before Obfscasion

```
import java.util.Scanner;public class test {public static void main(String[] args){String
Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B;int
ZA30B94FFF640802A90C381787F78CA24558182F50B484350D9705205A2C91FB3;double
MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39;String
M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55;String
B6D016A00D02FCA010E4DE350CD53403FA5D8E411F475CB9E915D4637937D9557;Scanner
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725 = new Scanner
(System.in);System.out.println("이름을
입력하세요");Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.next
();System.out.println("나이를
입력하세요");ZA30B94FFF640802A90C381787F78CA24558182F50B484350D9705205A2C91FB3 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextInt
();System.out.println("키를
입력하세요");MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextDouble
();System.out.println("자기소개를
입력하세요");M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextLine
();B6D016A00D02FCA010E4DE350CD53403FA5D8E411F475CB9E915D4637937D9557 =
NF7E37773635FACB58C898AB32234CB515444A7F96FE0F92F030E7CAE21D18725.nextLine
();System.out.println("이름은
"+Y5788E682E1AD8B3C0CC9F67D2A82016D42236153F5B9BFABBE617ECAE110411B+"나이는
"+ZA30B94FFF640802A90C381787F78CA24558182F50B484350D9705205A2C91FB3+",키는
"+MC5E3AB367959A198849C50EB906738B71DEDB4234899D36FC8D72AC47278BA39+"입니다.");Syste
m.out.println(M9CC47D5CA9FEEE88F3FB8CBF8C50284829FD503AB91B720FF8BD2927971FEB55);}}
```

after Obfscasion

4. 결 론

- 본 논문에서는 기존의 난독화 기법을 기반으로 하여, KECCAK을 이용한 **코드 난독화 프로그램**을 개발하였다.

- **KECCAK 알고리즘**을 사용하여 난독화를 진행하기 때문에 역상저항성에 강하여 본래 해시값으로 기존 변수명을 찾기가 매우 힘들다. 하지만, 악의적인 사용자가 정보를 지속적으로 수집 할 경우 기존 변수명을 찾을 확률이 존재하기 때문에 KECCAK알고리즘을 사용하기 전 변수명에 **난수를 추가**하여 프로그램을 실행할 때마다 **같은 변수명**이라도 다른 해시값을 출력한다.

- **주석과 줄 바꿈을 제거**함으로써 가독성을 떨어뜨려 소스코드 분석을 통한 프로그램 **무단복제와 같은 악용사례를 예방한다.**