Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.4, Aug. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.4.827

안드로이드 간편결제 애플리케이션 보안 솔루션 결과값 변조를 통한 검증기능 우회 방법에 대한 연구*

유 재 욱,[1†] 한 미 정,[2] 김 규 헌,[3] 장 준 영,[3] 진 호 용,[4] 지 한 별,[5] 신 정 훈,[6] 김 경 곤[7‡]
[1]가천대학교, [2]조선대학교, [3]고려대학교, [4]세종대학교, [5]서울과학기술대학교,
[6]티오리, [7]고려대학교 정보보호대학원

A Study on Method for Bypassing Verification Function by Manipulating Return Value of Android Payment Application's Security Solution*

Jaewook You,[1†] Mijeong Han,[2] Kyuheon Kim,[3] Junyoung Jang,[3]
Hoyong Jin,[4] Hanbyeol Ji,[5] Jeonghoon Shin,[6] Kyounggon Kim[7‡]
[1]Gachon University, [2]Chosun University, [3]Korea University, [4]Sejong University,
[5]Seoul National University of Science and Technology, [6]THEORI
[7]Center for Information Security Technologies(CIST), Korea University

요 약

2014년도부터 금융권의 규제가 완화됨으로써 간단한 인증만으로 대금을 결제 할 수 있는 모바일 간편결제 시장이 확대되어 다양한 간편결제 서비스가 등장하고 있다. 모바일 간편결제 애플리케이션의 보안 위협을 막기 위해 여러 검증 기능을 가진 보안 솔루션들이 애플리케이션에 적용되었지만, 보안 솔루션의 적용 구조상 취약점은 여전히 발생할 수 있다. 본 논문에서는 간편결제 애플리케이션과 보안 솔루션을 프로세스 관점에서 분석하여, 검증 결과 값을 변조하는 것만으로도 각 보안 기능의 상세 분석 없이도 보안 솔루션의 검증 기능을 쉽게 우회 할 수 있음을 실제 간편결제 애플리케이션들을 대상으로 한 실험을 통해 증명한다. 그리고 본 논문에서 제시한 우회 방법의 대응방안을 세 가지 시점으로 나누어 제시함으로써 국내 간편결제 서비스의 보안성을 향상시킬 수 있도록 기여한다.

ABSTRACT

Since 2014, ease of regulations on financial institutions expanded the mobile payment market based on simple authentication, and this resulted in the emergence of various simple payment services. Although several security solutions have been used to mitigate possible security threats to payment applications, there are vulnerabilities which can still be found due to the structure in which the security solution is applied to the payment service. In this paper, we analyze the payment application and security solution from the process perspective, and prove through experimentation that verification functions of security solutions can be bypassed without detailed analysis of each security function, but by simply manipulating the verification result value. Finally, we propose methods to mitigate the bypass method presented in this paper from three different perspectives, and thereby contribute to the improvement of security level of the payment service.

Keywords: Mobile Payment, Android Security, Security Solution, Android Security Verification

Received(12. 20. 2017), Modified(1st: 05. 17. 2018, 2nd: 07. 10. 2018), Accepted(07. 11. 2018)
* 본 연구는 KITRI 차세대 보안리더 양성 프로그램(Best of the Best)의 연구 지원으로 수행되었습니다.
† 주저자, jaewook376@naver.com
‡ 교신저자, anesra@gmail.com(Corresponding author)

I. 서 론

금융 산업에 IT 기술이 접목된 핀테크(fintech) 산업이 활성화되고 있는 현재, 국내에서는 송금, 지급결제, P2P(Peer-to-Peer) 금융 플랫폼, 금융데이터 분석 등 다양한 분야에서 핀테크 기술이 성장하고 있다[1]. 이 중 모바일 환경에서의 지급결제는, PC 기반 위주의 온라인 쇼핑이 모바일 기반으로 전환됨에 따라 2017년 상반기 전년 동기 대비 41.2%가 증가하는 등 빠른 성장세를 보이고 있다[2].

이러한 모바일 지급결제 서비스는 인증을 간편화하여 결제 할 수 있기 때문에 '간편결제 서비스'로 불린다. 하지만 역공학이 쉬운 Java 코드로 구현된 안드로이드 애플리케이션의 특성상 금융 애플리케이션은 여러 보안 위협에 노출 되어있다. 모바일 보안 솔루션 개발사 ARXAN의 조사 결과, 고객의 자산과 직결된 금융 분야 모바일 애플리케이션의 95%가 해킹당한 이력이 있음을 확인할 수 있다[3].

모바일 애플리케이션을 대상으로 한 해킹 공격은, 애플리케이션의 위·변조를 통한 재배포 공격, 역분석을 통한 기능 분석 및 검증 우회, 통신 상 변조 공격 등으로 나타나고 있다[4]. Fig. 1.과 같이, 모바일 뱅킹 애플리케이션을 가장한 악성 애플리케이션이 지속적으로 나타나고 있는 상황이다[5]. 특히 북한의 사이버 부대가 전 세계의 금융 기관을 공격 표적으로 삼아 외화를 확보하고 있는 현재[6], 금융 애플리케이션에 대한 보안성의 확보가 더욱 중요해지고 있다.

이러한 상황에서 2014년도부터 핀테크 산업의 활성화를 위한 규제 완화로 인증이 간편화되어[7], 인증 방식의 간편화와 함께 보안성을 확보할 수 있는 방법에 대한 연구의 필요성이 늘어나고 있다. 현재의

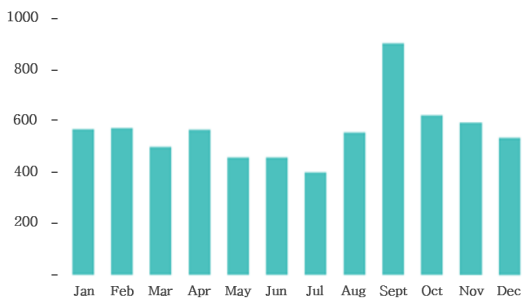


Fig. 1. Monthly breakdown of fake banking applications malware discovered in 2015

간편결제 애플리케이션은 애플리케이션의 위·변조와 악성 애플리케이션의 탐지, 역분석을 어렵게 하는 등의 보안 기능을 수행하는 보안 솔루션을 탑재하여 보안성을 확보하고 있다.

하지만 간편결제 애플리케이션들을 분석한 결과, 애플리케이션의 기능이 모두 완성된 이후 보안 솔루션을 탑재하여, 보안 솔루션과 애플리케이션이 독립적으로 동작하는 것을 확인할 수 있었다. 이러한 독립적 구조에서는 보안 솔루션의 검증 결과를 애플리케이션에 반환할 때, 이를 위·변조 하여 정상 응답값으로 수정하는 것만으로도 검증기능을 우회할 수 있는 것을 실제 간편결제 애플리케이션에 탑재된 보안 솔루션을 대상으로 한 실험을 통해 확인하였다.

본 논문에서는 이러한 독립적 구조상의 문제점과, 이로 인해 발생할 수 있는 검증기능 우회 취약점을 밝히며, 이에 대한 대응방안을 구체적으로 제시함으로써 현 간편결제 서비스의 보안성 향상에 기여하고자 한다.

본 논문의 2장에서는 애플리케이션의 보안성을 향상시키기 위한 기존의 기법들에 대한 연구와 그 한계점을 설명한다. 3장에서는 애플리케이션에 탑재된 보안 솔루션의 구조와 독립적 구조상 발생할 수 있는 검증기능 우회 취약점, 그리고 그 진단 방법을 설명한다. 4장에서는 3장에서 설명한 방법으로 실제 간편결제 애플리케이션을 대상으로 한 실험과 그 결과를 밝힌다. 5장에서는 본 논문에서 밝힌 취약점에 대한 대응방안을 제시하고, 끝으로 6장에서 결론과 향후 연구방향에 대해 서술한다.

II. 관련 연구

본 장에서는 안드로이드 애플리케이션의 보안성을 향상시키기 위한 기술들에 대한 선행 연구들을 알아본다.

정상 안드로이드 애플리케이션의 위·변조를 통하여 중요정보를 탈취할 수 있는 악성코드를 삽입하거나 거래루틴 수정을 통한 부정 거래 등의 공격을 수행할 수 있다. 이러한 공격을 막고 애플리케이션의 보안성을 향상시키기 위한 보안 검증기능 우회 취약점과 그 대응방안에 대한 연구들이 진행되고 있다.

금융 애플리케이션이 실행될 때 함께 실행되는 백신 애플리케이션의 검사 지점 우회 취약점과[8], 애플리케이션의 위·변조를 탐지하기 위한 무결성 검증기능 우회 취약점[9]등 보안 검증기능을 우회할 수

있는 취약점에 대한 연구가 실제 뱅킹 애플리케이션을 대상으로 한 실험을 통해 진행되었다.

특히 뱅킹 애플리케이션에서 무결성 검증기능을 우회하고, 리패키징을 통해 사용자가 송금하고자 하는 대상을 공격자로 변경하여 부정 거래를 진행하는 시나리오 기반의 실험을 통하여 금융 애플리케이션이 위·변조 되었을 때의 위험성을 제시하는 연구 또한 진행되었다[10].

애플리케이션의 보안 검증기능을 우회할 수 있는 취약점에 대한 연구가 진행됨과 동시에, 이러한 취약점에 대응하기 위한 여러 기법들에 대한 연구 또한 진행되었다. APK 파일의 클래스를 동적으로 로딩하여 정적 분석과 변조를 어렵게 하는 연구와[11], 동적으로 정해지는 메모리에 적재된 클래스 주소를 Key 값으로 해시 계산을 하는 기법을 통해 검증기능을 강화하는 등의 연구가 제시되었다[12].

또한 의사 명령어와 메소드 진입 분기문을 중요 클래스의 메소드 앞에 삽입함으로써 역공학 도구들의 분석을 방해하는 연구와[13], 단말 내의 이벤트를 추출하여 악성 애플리케이션의 루팅 공격을 탐지하는 기법에 대한 연구[14]등, 안드로이드 애플리케이션의 보안 검증기능 향상에 대한 다양한 연구가 진행되고 있다.

그리고 최근, 금융 애플리케이션의 실행 환경을 검증하는 자가 보호 기능(SDM, Self Defence Machine) 중 루팅 체크와 무결성 체크 기능의 상세 실행 흐름과 자가 보호 기능의 실행 흐름을 분석하고, 이 기능의 코드 위치를 탐색할 수 있는 방안에 대한 연구가 제시되었다[15].

하지만, 본 논문에서 제시하는 '애플리케이션과 보안 솔루션의 독립적 구조상 발생하는 검증기능 우회 취약점'은, 검증기능 그 자체에 대한 취약점이 아닌, 애플리케이션과 보안 솔루션 탑재 프로세스에서 발생하는 독립적 구조로 인하여 정교한 검증기법에 대한 상세한 루틴 분석 없이도 검증기능의 우회가 가능한 것이기 때문에, 이에 대한 상세한 연구와 독립적 구조를 해결할 수 있는 대응책이 필요한 실정이다.

III. 보안 솔루션의 적용 구조 및 검증기능 우회 취약점

본 장에서는 애플리케이션에 보안 솔루션이 적용된 구조와, 검증 결과값 위·변조를 통한 검증기능 우회 취약점에 대해 기술한다.

3.1 안드로이드 애플리케이션의 보안 솔루션 적용 구조

안드로이드 애플리케이션의 APK 파일은 Java로 구현된 코드와 리소스 파일이 컴파일 되어 구성된다. 애플리케이션의 Java class 파일은, 안드로이드의 달빅(dalvik) 가상머신이 인식할 수 있도록 DEX(Dalvik Executable) 파일로 변환된다. 또한 안드로이드는 JNI(Java Native Interface)를 통해 Java가 아닌 C/C++, 혹은 어셈블리어로 작성된 네이티브 라이브러리를 달빅 가상머신에서 사용할 수 있도록 지원한다.

현재 이러한 안드로이드 애플리케이션의 구조에서 보안 솔루션은 일반적으로 Fig. 2.와 같이 적용되어 동작한다. 애플리케이션의 구동 시, DEX 파일 내에서 주 서비스는 보안 솔루션을 호출한다. 보안 솔루션은 검증 기능을 네이티브 라이브러리에서 수행한 후, 검증 결과값을 반환한다. 반환된 결과값을 토대로 무결성 훼손, 변조된 OS(루팅), 디버깅 환경에서 애플리케이션 실행 등 보안성을 저해할 수 있는 상황을 식별하여 애플리케이션을 제어함으로써 보안 위협을 차단한다.

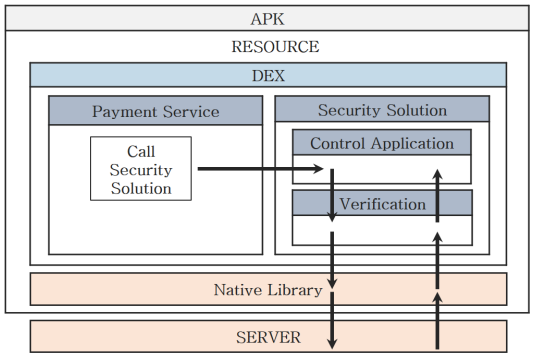


Fig. 2. Flow of Android Application's Security Solution

3.2 검증 결과값 위·변조를 통한 검증기능 우회 취약점

3.1절에서 설명한 바와 같이 애플리케이션 보안 솔루션은 전달받은 반환값을 토대로 보안 검증과 제어를 수행한다. 때문에 보안 검증에 실패할지라도, 네이티브 라이브러리나 DEX의 반환값을 정상 응답값으로 위·변조한다면 보안 솔루션의 검증 기능이 쉽게 우회될 수 있다.

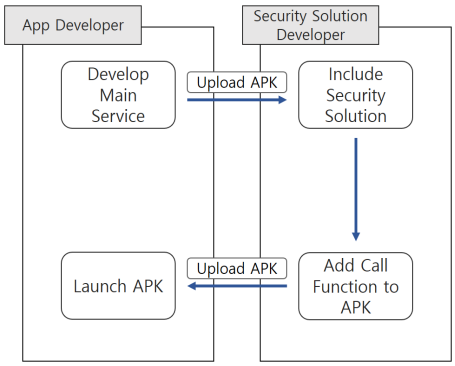


Fig. 3. Process of Adapting Security Solution on Application

현재 일반적인 보안 솔루션의 적용 과정은 Fig. 3.와 같다. 애플리케이션 개발사는 메인 서비스의 개발을 완료한 후, 보안성의 확보를 위해 원본 APK파일을 보안 솔루션 제공업체에 전달한다. 보안 솔루션 제공업체는 개발사에서 전달받은 APK 파일에 보안 검증기능을 수행하는 네이티브 라이브러리나 DEX 코드를 포함하고, 이를 애플리케이션 구동시 호출하게 하여 보안 솔루션을 적용되었는지 확인한다.

이와 같은 보안 솔루션 적용 과정에선 애플리케이션의 메인 서비스와 보안 솔루션은 독립적이다. 개발 과정에서 보안 검증기능에 대한 고려가 없었기 때문에, 보안 솔루션을 우회 하더라도 메인 서비스의 동작에는 아무런 지장이 없다.

본 장에서 제시한 취약점은 이러한 독립적 구조로 인하여 발생한다. 만약 보안 솔루션이 정상적으로 보안 검증기능을 수행하지 않았을 시 애플리케이션 서비스에서 종속적으로 사용하는 네이티브 라이브러리의 코드가 실행되지 않는다면, 검증 결과값을 정상값으로 우회한다고 하더라도 애플리케이션이 정상적으로 작동하지 않을 수 있다.

IV. 검증기능 우회 취약점 진단 방법 및 실험 결과

본 장에서는 3장에서 기술한 검증 결과값 위·변조를 통한 검증기능 우회 취약점을 진단하기 위한 방법과 실제 간편결제 애플리케이션을 대상으로 한 검증기능 우회 실험 결과를 설명한다.

4.1 검증기능 우회 취약점 진단 방법

본 절에서는 보안 솔루션의 검증기능을 우회하기 위한 방법을 순서와 분석 대상에 따라 Fig. 4.와 같이 크게 세 단계로 나누어 제시한다.

4.1.1 APK 파일 디컴파일을 통한 소스코드 분석

분석 대상 애플리케이션의 보안 솔루션 적용 구조를 파악하기 위해 Fig. 5.와 같이 APK 파일을 디컴파일하여 소스코드를 분석한다. 3.1절에서 설명했듯이, Java 클래스 파일은 달빅 가상머신이 인식할 수 있는 DEX 파일로 변환된다. 이 DEX 파일을 다시 Java 코드로 변환하는 디컴파일 과정을 거쳐 애플리케이션의 소스코드를 분석한다.

애플리케이션의 구동 시, 메인 서비스가 동작하는 DEX에서 보안 솔루션을 호출하기 때문에, 디컴파일된 코드 분석을 통해 보안 솔루션의 호출, 제어, 검증 과정을 파악할 수 있다.

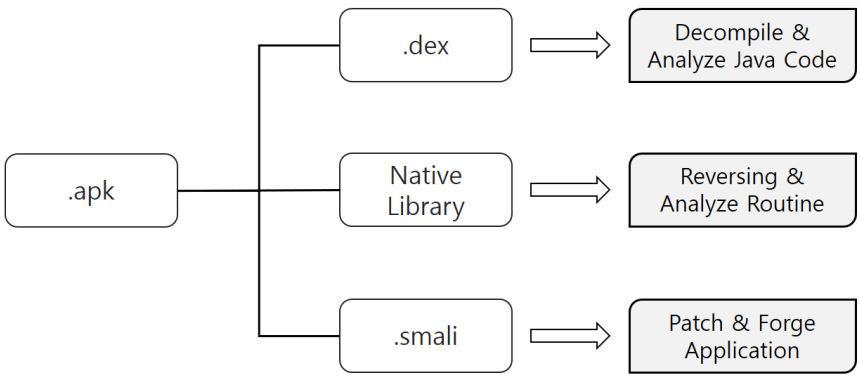


Fig. 4. Analyzing Step to Bypass Security Solution

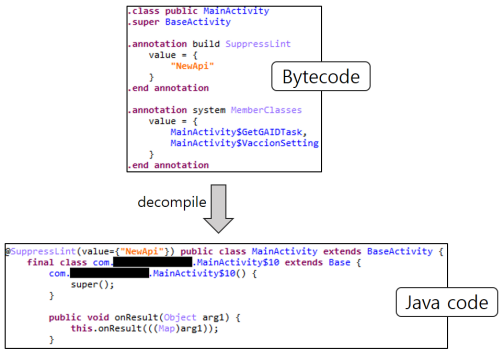


Fig. 5. Bytecode & Decompiled Java Code

4.1.2 네이티브 라이브러리 역분석

일반적으로 보안 솔루션의 실질적인 보안 검증은 네이티브 라이브러리에서 이루어진다. 때문에 보안 검증 기능을 수행하는 네이티브 라이브러리를 역분석하여 보안 솔루션의 검증 및 결과값 반환 과정을 파악한다. Fig. 6.와 같이 디컴파일된 자바코드에서 어떤 네이티브 라이브러리를 호출하는 지 파악하고, 해당 라이브러리의 분석을 수행한다.

Fig. 7.은 네이티브 라이브러리 내에서 루팅과 무결성을 검증하고, 그 결과값을 반환하는 부분의 역분석된 소스코드이다. 변수 R1에 검증 함수의 반환값

```
static {
    String v3 = null;
    MainActivity.c3[redacted]9e = false;
    MainActivity.c2[redacted]d8 = v3;
    MainActivity.c5[redacted]f5 = new ArrayList();
    MainActivity.cc[redacted]fd = false;
    System.loadLibrary("android-anti-debug");
```

Fig. 6. Load Native Library with DEX Code

```
return_value = (char *)APKPath_Check(v28);
if ( return_value )
{
  v9 = "APK Path verify failed";
}
else
{
  return_value = (char *)GetSOPath(v42, 100);
  if ( !return_value )
  {
    ...(Omission)...
    R1 = Rooting_And_Integrity_Check(v31, v32, "1",
v28, v42, v44, v35, 0, &v41, v11, &v38, &v36,
&v37, 2, 0);
    return_value = R1
    ...(Omission)...
```

Fig. 7. Verification Routine in Native Library

을 저장하고, 네이티브 라이브러리에서 DEX단 실행 환경으로 반환할 변수에 이를 저장한다. 네이티브 라이브러리의 반환값은 DEX에서 네이티브 라이브러리를 호출한 함수의 반환값이 되어 이를 바탕으로 애플리케이션 구동 환경을 판단한 후에 애플리케이션을 제어하는 것을 확인할 수 있었다.

4.1.3 코드 패치를 통한 검증 결과 반환 값 변조

DEX와 네이티브 라이브러리의 분석을 통해 호출/검증 루틴을 파악한 후, smali나 네이티브 라이브러리의 루틴에서 정상 값을 반환하도록 코드를 패치하여 검증 기능을 우회할 수 있다.

Fig. 8.에서 본래의 코드는, 검증기능을 수행하고 그 결과를 v0에 저장하여(move-result-object v0) 이를 반환(return-object)한다. 그러나 만약 이 v0의 값을 정상 응답값으로 변조(const-string/jumbo v0, "0000")한 후 반환하면, 보안 솔루션은 검증 결과가 정상이라고 판단할 것이다.

이러한 검증 결과값 변조를 통해 무결성 검증기능을 우선적으로 우회하면, Fig. 9.과 같이 루팅 검사, 안티 디버깅, 악성 애플리케이션 탐지 등의 보안 검증기능을 아예 호출하지 않거나, 검증 루틴 내부를 모두 제거하여 기능을 무력화 하는 등의 코드 패치 또한 가능해진다. 본 논문에서 제시하는 방법으로 보안 솔루션을 우회한다면, 검증 기능 루틴의 상세 분석 없이도 손쉽게 검증기능을 모두 우회 혹은 무력화 할 수 있다.

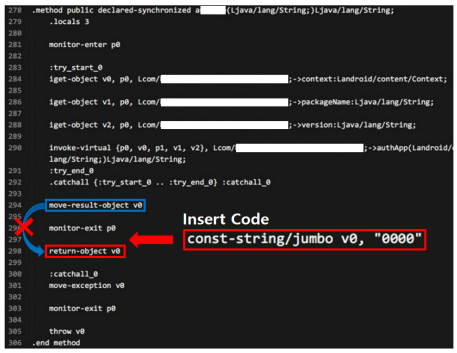


Fig. 8. Patch Smali Code to Bypass Security Verification Routine

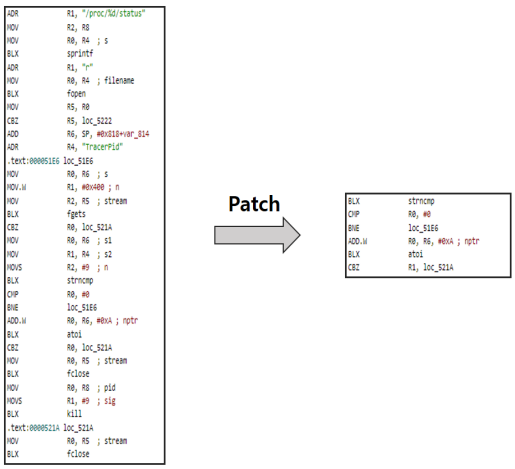


Fig. 9. Patch Native Library Code to Bypass Security Verification Routine

4.1 검증기능 우회 실험 결과

4.1.1 우회 실험 대상 선정 및 탑재 보안 솔루션 기능 조사

간편결제 서비스는 이를 제공하는 기업의 업종과 플랫폼 제공 방식에 따라 그 성격이 나뉜다. 본 논문에서 제시한 검증 결과값 위·변조를 통한 검증기능 우회 취약점의 진단 실험은 서비스사의 업종을 인터넷 서비스업, 은행업, 유통업, 단말 제조업 4개의 분야로 나누고, 각 분야의 애플리케이션 가운데 이용률이 높은 상위 5개 애플리케이션 중, 무작위로 총 7개 애플리케이션을 선정하였다. 실험 환경은 루팅된 단말과 루팅되지 않은 단말 모두에서 실험을 진행하였으며, 안드로이드 버전 6.0.1(Marshmallow)과 7.0(Nougat)에서 실험을 진행하였다.

간편결제 애플리케이션들을 분석한 결과, Table 1.과 같이 한 개의 간편결제 애플리케이션에 복수의 보안 솔루션이 탑재되어있는 경우가 많았다. 탑재된 보안 솔루션들은 Table 2.와 같이 안티 디버깅, 루팅 체크, 무결성 체크, 악성코드 검사, 에뮬레이터 체크 등의 보안 검증 기능을 각각 제공하고 있음을 확인할 수 있었다.

4.1.2 검증기능 우회 실험

본 실험에서는, 4.1절에서 제시한 분석 방법으로, 각 간편결제 애플리케이션의 디컴파일된 코드와 보안 솔루션의 네이티브 라이브러리를 역분석하여 검증 결과값 변조를 통해 검증 기능을 우회할 수 있는 지점을 확인하였다. 이러한 지점 중 먼저 애플리케이션의 무결성을 검증하는 지점을 코드 재작성을 통해 우회하고, 이후 다른 보안 검증 기능들은 수행하지 않게 하거나 정상 검증 결과값을 반환하게 함으로써 무력화하였다. 뿐만 아니라, 이렇게 보안 검증 기능을 우회하도록 리패키징된 간편결제 애플리케이션의 결제, 송금 등의 기능이 정상 동작되고 이를 사용하는데 아무 문제가 없었으므로 검증 기능의 우회 및 무력화의 성공을 확인했다.

G 애플리케이션을 제외한 모든 애플리케이션은 jeb[16]와 apktool[17]을 통하여 디컴파일 및 리패키징을 할 수 있었다. 이는 애플리케이션과 보안 솔루션의 독립적 구조로 인해 보다 손쉽게 보안 솔루션을 우회할 수 있음을 의미한다. 또한 리패키징을 통해 소스코드를 분석할 때 소스코드에 대해 기본적인 난독화만이 적용되었기 때문에 루틴 분석에 어려

Table 1. Current State of Payment Application's Security Solution and Test Result of Bypass

Payment Application	Security Solutions	Decompile	Repackaging	Bypass Points	Bypass& Use service
A	a, b	O	O	DEX	O
B	a, b, c	O	O	DEX	O
C	a, b, d, e	O	O	DEX	O
D	b	O	O	DEX	O
E	a, b, c, f	O	O	Native Library, DEX	O
F	h, i	O	O	Native Library, DEX	O
G	a, b, g	X	X	X	X

Table 2. Verification Function for Each Security Solution

Security Solutions	Rooting Check	Integrity Check	Debugging Check	Detecting Malicious App.	Emulator Check	Bypass Solutions
a	O	O	O	O	X	O
b	O	O	O	X	O	O
c	O	X	X	O	X	O
d	O	O	O	X	X	O
e	O	O	O	X	X	O
f	O	O	X	X	X	O
g	O	O	O	X	O	X
h	O	O	O	X	O	O
i	O	X	X	O	X	O

움이 없었고, 디컴파일 방지기법이 적용되지 않아 java 코드와 smali 코드를 쉽게 획득할 수 있었으며, 네이티브 라이브러리의 패킹이 적용되어있지 않아 곧바로 역분석이 가능했기 때문이기도 하다. 이에 반해 G 애플리케이션과 그에 탑재된 g 솔루션의 경우, 디컴파일과 리패키징이 불가했기 때문에 java 코드와 smali 코드의 분석 및 패치가 불가능하였고, 네이티브 라이브러리가 패킹되어 있어 이에 대한 역분석 또한 불가능하였다. 해당 G 애플리케이션과 g 솔루션의 경우 본 논문에서 제시한 방법과는 별개의 방법으로 우회하였기 때문에 본 논문에서는 해당 솔루션의 우회 방법에 대해서 소개하지 않도록 한다.

4.1.3 검증기능 우회 실험 결과

본 실험 결과, 실험대상으로 선정한 7개의 애플리케이션에 탑재되어 있는 9개의 보안 솔루션 중 8개의 보안 솔루션의 모든 보안 검증 기능을 모두 우회하여 신뢰되지 않는 환경(루팅된 단말, 에뮬레이터, 위변조된 애플리케이션 등)에서 정상적으로 사용할 수 있도록 보안 솔루션을 무력화 할 수 있었다.

본 논문에서 제시한 우회 방법이 대부분의 솔루션에 적용될 수 있었던 가장 큰 이유는 3.2에서 설명한 바와 같이 애플리케이션의 서비스와 보안 솔루션이 독립적으로 동작하기 때문이다. 보안 솔루션이 실제 검증기능을 수행할 때 비정상임을 탐지하더라도, 그 결과만을 변조하여 애플리케이션을 제어하는 메서드에 정상 결과값을 전달하는 것으로 보안 솔루션을 손쉽게 우회할 수 있었다. 심지어 일부 보안 솔루션의 경우, 해당 솔루션의 검증 기능을 수행하는 네이티브 라이브러리를 아예 호출하지 않게끔 라이브러리 호출구문을 삭제하여도, 애플리케이션 동작에 아무 지장을 주지 않아 결과값을 변조할 필요도 없이 검증 기능을 모두 우회할 수 있었다.

또한 보안 솔루션에서의 검증 결과를 클라이언트에서 판단하는 것이 아닌, 서버로 해당 결과를 전송하여 애플리케이션과 구동환경의 상태가 정상인지 서버 단에서 이를 판단하고, 그 결과값을 다시 클라이언트로 전달하는 경우도 존재하였다. 이러한 경우 서버에서 전달되는 결과값을 네트워크 단에서 가로채 정상 결과값으로 바꾸는 스푸핑 공격을 통해서도 보안 솔루션의 검증 기능을 우회할 수 있었다[18].

각 솔루션은 난독화를 통해 그 소스코드를 역분석하기 어렵게 구현하였지만, 애플리케이션의 서비스와 보안 솔루션이 독립적으로 동작하여 보안 솔루션의 기능 수행과는 별개로 애플리케이션 서비스가 동작하기 때문에 검증 기능에 대한 자세한 분석 없이도 어렵지 않게 우회할 수 있다.

이러한 실험 결과로 애플리케이션의 보안 검증 기능이 모두 우회됐기 때문에, 루팅된 OS에서의 애플리케이션 동적 분석과 애플리케이션 위·변조를 통하여, 정보 탈취, 부정 거래, 허위 결제, 인증 우회 등 간편결제 애플리케이션에서의 취약점이 추가로 발생할 수 있다.

간편결제 애플리케이션과 같은 금융 애플리케이션은, 사용자의 자산과 직접적으로 연관되어있기 때문에 보안 솔루션을 통한 검증으로 더 높은 수준의 보안성을 확보하려 했지만, 본 논문에서 제시한 방법을 통해 검증 기능을 쉽게 우회할 수 있다는 사실을 확인할 수 있었다. 보안 솔루션을 우회할 수 있다는 것

은 이후 공격자가 애플리케이션 단에서 아무런 통제 없이 공격을 시도할 수 있다는 의미이기에, 간편결제 서비스 및 모바일 금융 서비스의 보안 위협이 증가하는 것으로 해석할 수 있다.

때문에 실험 이후, 본 논문에서 제시한 우회 취약점을 관련 기관 및 기업에 제보하여 현 문제점을 알렸고, 현재 관련 기관에선 이러한 문제점을 해결하기 위해 대응방안을 적용 중에 있다.

V. 대응 방안

본 장에서는 본 논문에서 제시한 검증기능 우회 취약점을 막기 위한 대응방안을 설계 시, 개발 시, 운영 시의 세가지 관점에서 나누어 제시한다.

5.1 설계 시 대응방안

본 논문에서 제시한 취약점을 대응하는 방법으로 초기 설계 시 보안 요건들을 적용하는 것이다. 본 논문에서 검토한 취약점 발생의 주요 원인 중 하나는 애플리케이션과 보안 솔루션이 독립적으로 동작하기 때문이다. 3.2절에서 설명한 보안 솔루션의 탑재 과정을 보면, 애플리케이션 서비스를 개발할 때 보안 솔루션 적용에 대해 고려하지 않고 서비스가 먼저 개발된다. 따라서, 보안 솔루션이 존재하지 않는다고 가정하여도 애플리케이션 서비스는 정상적으로 동작할 수 있다.

하지만 애플리케이션과 보안 솔루션이 종속적으로 동작한다면 본 논문에서 제시한 방법으로 검증기능을 우회하여 항상 정상 값을 반환하더라도, 검증 결과가 아닌 과정을 기반으로 검증기능의 수행을 보장할 수 있다. 만약 검증 과정에 문제가 있다면 종속성이 부여된 애플리케이션 주요 서비스의 동작을 막고 에러 상황을 유발함으로써 검증 기능을 보호할 수 있다. 애플리케이션 서비스 기능의 일부가 보안 솔루션에서 구현되도록 하거나, 기능을 사용하는데 필요한 자원을 보안 솔루션을 통해 가져오는 등의 방식으로, 애플리케이션 서비스와 보안 솔루션의 종속성을 확보할 수 있다.

이러한 대응방안을 적용하기 위해선 애플리케이션 서비스의 개발 전, 설계단계부터 보안 솔루션의 적용 방안에 대해 고려해야한다. 애플리케이션 구현 시 주요 서비스의 어느 부분을 보안 솔루션이 수행할 것인지 미리 설계한 후 구현하여야 보안 솔루션과 네이티브 라이브러리 사이에 종속성을 부여하면서도 서비스 기능 수행에 지장이 가지 않도록 구현할 수 있다.

5.2 구현 시 대응방안

애플리케이션의 구현 시에도 애플리케이션 서비스와 보안 솔루션 간에 종속성을 줄 수 있는 방안에 대해 고려할 수 있다.

Fig. 10.은 토큰을 이용하여 요청을 검증하는 프로세스에 대한 개요도이다. 4.2.3의 실험 결과에서 서버와 통신하여 검증 결과를 판단하는 경우가 있다고 설명하였다. 이러한 경우 검증 결과값을 클라이언트로부터 받은 서버는 정상이라고 판단되면 토큰을 발급하여 클라이언트에게 검증 결과와 토큰을 함께 전달한다. 이후 애플리케이션의 주요 서비스를 이용할 때 서버로의 요청과 함께 이전 보안성 검증 과정에서 발급받은 토큰을 함께 담아 보내고, 서버에선 토큰을 확인하여 유효한 토큰이 아니라면 요청을 거절한다[19].

이러한 과정을 통해 검증을 정상적으로 수행하지 않고 단순히 결과값을 변조하는 경우에는 애플리케이션은 정상 동작하더라도 주요 서비스는 이용하지 못하도록 막을 수 있다.

또한 4.2절에서 언급했듯이, 애플리케이션 분석에 별다른 제약이 없었기 때문에 검증기능을 쉽게 우회할 수 있었다. APK 디컴파일을 통해 공격자가 세부적인 분석을 하기 어렵도록 디컴파일 시 가장 많이 사용되는 툴인 apktool에서 강제로 에러를 유발하도록 하는 방법이 있다[20]. 또한, 공격자가 애플리케이션의 디컴파일 및 리패키징에 성공하더라도, 소스코드에서 사용되는 중요한 문자열을 암호화하여 공격자로부터 루틴을 파악하기 어렵게 하는 등, 쉽게

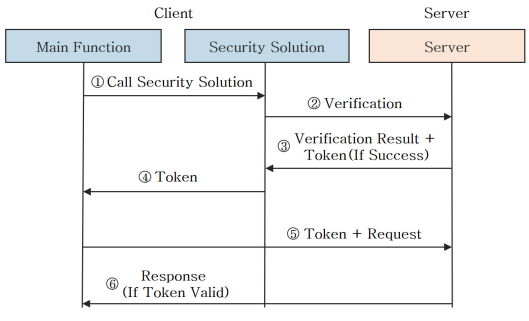


Fig. 10. Process of Token Issurance

구조를 파악할 수 없도록 소스코드 난독화를 강화해야한다.

5.3 운영 시 대응방안

간편결제 서비스가 운영되고 있는 과정에서 보안을 강화하기 위한 방법으로 부정 사용을 탐지하는 FDS(Fraud Detection System)와 정기적인 업데이트를 통한 분석 무효화 등이 있다.

첫 번째 방식인 FDS는 사용자의 평소 거래 데이터를 수집한 뒤, 이 정보들의 상관관계를 분석한다. 이후, 사용자가 평소 패턴 및 행위와 다른 형태의 결제가 이뤄질 때 이러한 거래가 정상인지, 부정인지를 판단하는 부정거래 탐지 시스템이다[21]. 이를 5.1절에서의 설계 시 대응방안과 연계하여, 정상적으로 검증이 수행되지 않았는데 트랜잭션 요청이 들어올 시, 비정상 거래로 간주하고 요청을 거부하는 등의 탐지기법을 활용할 수 있다.

또한 공격자가 난독화를 해제하여 애플리케이션의 루틴을 분석하고 주요 기능을 이미 파악했을 수도 있다. 이런 경우, 정기적인 업데이트를 통해 문자열 암호화 알고리즘, 검증 루틴 등을 추가로 변경하여, 기존의 분석을 무효화하고 다시 처음부터 분석하게끔 할 수 있다. 이러한 방식으로 공격자가 역분석 및 우회를 업데이트 주기 내에 성공해야만 하게끔 시간적 제약을 줌으로써 검증 기능의 우회를 더욱 어렵게 만들 수 있다.

VI. 결 론

본 논문에서는 안드로이드 애플리케이션 보안 솔루션의 독립적 구조로 인하여 발생하는 검증 결과값 위·변조를 통한 우회 취약점과 진단 방법에 대해 살펴보고, 이를 실제 간편결제 애플리케이션을 대상으로 한 실험을 통하여 취약점의 실효성과 위험성을 증명하였다. 또한 이에 대한 대응방안을 설계, 구현, 운영, 세 가지 시점으로 나누어 제시함으로써 국내의 간편결제 서비스의 보안성 향상에 기여하고자 하였다.

본 논문에서 밝힌 취약점의 핵심은 보안 솔루션의 상세 검증 루틴 분석 없이 손쉽게 우회가 가능하다는 것이다. 해당 취약점은 비단 간편결제 서비스 뿐 만 아니라, 보안 솔루션을 적용한 많은 모바일 금융 서비스에서도 동일하게 발생할 수 있다. 따라서 보안성 및 무결성 검증이 제대로 수행되지 않기에, 국제 해커 그룹이나 북한의 사이버 부대 등의 공격자가 이러한 보안 취약점을 통해 언제라도 국내 모바일 금융 서비스를 공격하여 국민의 자산과 중요정보를 탈취할 수 있는 상황이다. 국가경제의 기반이 되는 금융시장이 흔들릴 시, 그 여파는 금융 분야를 넘어 전 분야로 확산될 것이기에 모바일 금융 애플리케이션의 보안에 대한 연구와 대처가 시급한 것으로 보인다.

하지만 자유도가 높은 안드로이드 OS의 특성상 OS 자체가 분석과 공격하기 쉬운 환경이고, 이러한 환경에서 상세 검증 루틴을 파악하고 여러 공격 기법을 사용한다면 본 논문에서 제시한 대응방안 또한 우회가 가능하다. 때문에 향후에는 분석과 패치만으로는 검증기능을 우회할 수 없게끔 안드로이드 트러스트존(Trust-Zone)을 이용한 애플리케이션의 보안성을 향상시킬 수 있는 기법에 대한 연구를 진행할 계획이다.

References

[1] “Survey on Mobile Payment Service (Fintech1),” Korea consumer Agency, pp. 1-2, May. 2016

[2] “Payment trend in the first half of 2017,” The Bank of Korea, pp. 2, Sep. 2017

[3] “State of Security in the App Economy: Mobile Apps Under Attack,” ARXAN, Vol. 1, Research Report, Aug. 2012

[4] Kyounggon Kim, “Study on Security Diagnosis Method for Android Mobile App,” Master’s Thesis, Korea University, Feb. 2015

[5] “HPE Security Research - Cyber Risk Report 2016,” Hewlett Packard Enterprise, pp. 42, Feb. 2016

[6] Timothy W.Martin, “North Korea’s Army of Hackers Has a New Target: Bank Accounts,” The Wall Street Journal, Jul. 2017

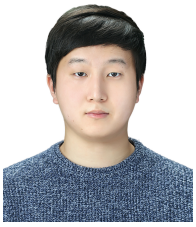
[7] Heesok Seo, “Status of Legal Regulations on Electronic Payments in Korea,” Journal of Consumer Law,

2(2), pp. 155-176, Sep. 2016
〔8〕 Woojin Lee and Kyungho Lee, "A Study on the Vulnerability of Using Intermediate Language in Android: Bypassing Security Check Point in Android-Based Banking Applications," Journal of The Korea Institute of Information Security & Cryptology, 27(3), pp. 549-562, Jun. 2017
〔9〕 Soonil Kim, Sunghoon Kim, and Dong Hoon Lee, "A study on the vulnerability of integrity verification functions of android-based smartphone banking applications," Journal of The Korea Institute of Information Security & Cryptology, 23(4), pp. 743-755, Aug. 2013
〔10〕 Jin-Hyuk Jung, Ju Young Kim, Hyeong-Chan Lee, and Jeong Hyun Yi, "Repackaging Attack on Android Banking Applications and Its Countermeasures," Wireless Personal Communications, Vol. 73, Issue. 4, pp. 1421-1437, Dec. 2013
〔11〕 Hyunjo Kim and Jin-Young Choi, "Research on Secure Coding and Weakness for Implementation of Android-based Dynamic Class Loading," Journal of Korea Multimedia Society, 19(10), pp. 1792-1807, Oct. 2016
〔12〕 Jeong-min Kim, "A study on the vulnerability strengthening of android banking app using dynamic key value," Master's Thesis, Hannam University, Feb. 2017
〔13〕 Chanhee Lee, Yoon-Sik Jeong, and Seong-Je Cho, "A Method to Protect Android Applications against Reverse Engineering," Journal of Security Engineering, 10(1), pp. 41-50, Feb. 2013
〔14〕 Hyung-Woo Lee, "Android based Mobile Device Rooting Attack Detection and Response Mechanism using Events Extracted from Daemon Processes," Journal of The Korea Institute of Information Security & Cryptology, 23(3), pp. 479-490, Jun. 2013
〔15〕 Taehun Kim, Hyeonmin Ha, Seoyoon Choi, Jaeyeon Jung, and Byung-Gon Chun, "Breaking Ad-hoc Runtime Integrity Protection Mechanisms in Android Financial Apps.," Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 179-192, Apr. 2017
〔16〕 JEB decompiler software, "JEB", https://www.pnfsoftware.com/, Oct. 2017
〔17〕 A tool for reverse engineering Android apk open source, "apktool", https://github.com/iBotPeaches/Apktool, Oct. 2017
〔18〕 Kyounggon Kim, "Countermeasure of e-payment app security solution problem," The Korea Contents Association, 16(2), pp. 14-19, Jun. 2018
〔19〕 Kyuheon Kim, Mijeong Han, Jaewook You, Junyoung Jang, Hoyong Jin, Hanbyeol Ji, Kyounggon Kim, and Jeonghoon Shin, "A Study on Countermeasure for Bypassing Android Security Solution through Manipulating Return Value," Proceedings of the Korea Institutes of Information Security and Cryptology Conference, Dec. 2017
〔20〕 Tim Strazzere, "Dex Education: Practicing Safe Dex," Blackhat USA 2012, Jul. 2012
〔21〕 Financial Security Institute, "E-Finance And Financial Security," Financial Security Institute, 1(15), pp. 67-98, Jul. 2015

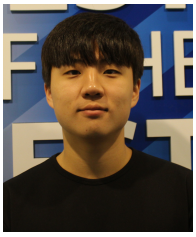
〈저자소개〉

유 재 욱 (Jaewook You) 학생회원
2012년 3월~현재: 가천대학교 컴퓨터공학과
2018년 4월~현재: 라온화이트햇 전임연구원
〈관심분야〉 정보보호, 보안컨설팅, 모의해킹, 금융보안

한 미 정 (Mijeong Han) 정회원
2017년 8월: 조선대학교 컴퓨터공학과 졸업
〈관심분야〉 정보보호, 보안컨설팅, 금융보안, 웹/앱 보안



김 규 헌 (Kyuheon Kim) 학생회원
2015년 3월~현재: 고려대학교 정보보호학부
〈관심분야〉 정보보호, 모바일보안, 시스템보안



장 준 영 (Junyoung Jang) 학생회원
2015년 3월~현재: 고려대학교 정보보호학부
〈관심분야〉 역공학, 시스템보안, 앱보안

진 호 용 (Hoyong Jin) 학생회원
2016년 3월~현재: 세종대학교 정보보호학과
〈관심분야〉 시스템보안, 금융보안

지 한 별 (Hanbyeol Ji) 학생회원
2014년 2월~현재: 서울과학기술대학교 산업정보시스템공학과
2017년 4월~현재: 라온화이트햇 전임연구원
〈관심분야〉 보안컨설팅, 핀테크보안, 금융보안, 웹/앱 보안

신 정 훈 (Jeonghoon Shin) 정회원
2014년 8월: 세종대학교 컴퓨터공학과 졸업
2014년 9월~2017 12월: 국방과학연구소 연구원
2015년 7월~현재: KITRI Best of the Best 멘토
2017년 12월~현재: 티오리(THEORI) 연구원
〈관심분야〉 소프트웨어 버그 리서치

김 경 곤 (Kyounggon Kim) 정회원
2008년 2월: 숭실대학교 컴퓨터공학과 졸업
2015년 2월: 고려대학교 정보보호대학원 석사
2003년 8월~2006 2월: A3 시큐리티 컨설턴트
2006년 3월~2007 12월: SK 인포섹 시니어컨설턴트
2008년 1월~2011 12월: 삼일회계법인(Samil PwC) 매니저
2011년 12월~2017 8월: 딜로이트 코리아 시니어매니저
2014년 7월~현재: KITRI Best of the Best 멘토
2016년 9월~현재: 고려대학교 정보대학 산학협력중점교수
〈관심분야〉 기업보안컨설팅, 정보보안 교육 개발, 악성코드분석, 인공지능보안