http://dx.doi.org/10.3745/KTCCS.2013.2.10.445

A Proposal for "Security Verification Method for Implementation of Secure Android Mobile Application"

Hur Hwan Seok† · Kang Sung Hoon†† · Kim Seung Joo†††

ABSTRACT

Mobile applications today are being offered as various services depending on the mobile device and mobile environment of users. This increase in mobile applications has shifted the spotlight to their vulnerability. As an effective method of security verification, this paper proposes "phase-wise security verification for the implementation of mobile applications." This method allows additional security verification by covering specific items across a wider range compared to existing methods. Based on the identified weaknesses, it detects the cause of vulnerability and monitors the related settings.

Keywords : Android App Vulnerability, App Security Verification, Mobile App Vulnerability Checklist

안전한 안드로이드 어플리케이션 개발을 위한 구현 단계별 보안성 검증 방안 제시

허 환 석† · 강 성 훈†† · 김 승 주†††

요 약

최근 증가하는 모바일 어플리케이션은 다양한 모바일 기기 및 사용자 별 모바일 환경에 따라 다양하게 서비스 되고 있다. 그러나 늘어나는 모바일 어플리케이션의 보안 취약성 문제점도 크게 대두되고 있는 실정이다. 이러한 보안 문제점을 효과적으로 검증하기 위해 본 논문에서는 "모바일 앱 구현 단계별 보안성 검증 방안"을 제안하고자 한다. 이는 기존 검증 방안 보다 더욱 구체적인 항목과 넓은 범위를 확인 할 수 있어 확인 되지 않았던 추가적인 항목에 대한 보안성 체크가 가능하며, 도출된 취약점을 바탕으로 해당 취약사항의 원인 및 발생환경 체크가 가능한 개선된 방법이다.

키워드 : 안드로이드 앱 취약점, 앱 보안 검증, 모바일 앱 취약점 점검리스트

1. 서 론

최근 각종 모바일 서비스가 활성화됨에 따라 스마트폰 사용이 증가하고, 새로운 모바일 어플리케이션(이하 앱)의 출시 및 사용이 급격하게 늘어나고 있다. 이에 늘어나는 스마트폰 앱의 보안 취약성 문제점도 크게 대두되고 있는 상황이다. 현재 많은 스마트폰의 모바일 앱이 사용자의 중요정보 및 개인정보를 이용하여 서비스 되고 있다. PC에서 웹 브라우저로 로그인시 사용되는 아이디/패스워드를 그대로 이용하거나, 금융 거래가 가능한 앱 및 모바일을 통한 쇼핑 앱에서 사용하는 인터넷 뱅킹 인증서 정보, 계좌정보, 패스워드 정보 등이 아무런 가공 없이 모바일에서도 동일하게 사용되고 있다. 이런 정보들 외에도 SNS(Social Network Service) 관련 앱의 문자 메시지 및 통화서비스 또한 개인의 정보를 포함한다. 이러한 정보들은 사용자의 의도와는 달리 스마트폰 단말기 내부에 남겨지기도 하며, 의도하지 않게 다른 앱에 노출되거나 보안에 안전하지 않은 네트워크 구간을 사용하여 송수신되기도 한다. 아무런 보안조치가 없이 개발되는 스마트폰 모바일 앱의 문제점은 개인의 프라이버시를 위협할 수도 있으며 경우에 따라 사회적, 경제적인 피해를 유발시킬 수도 있다. 따라서 스마트폰 앱의 취약성 검증을 통한 보안 대책이 시급히 요구되고 있다.

안드로이드 스마트폰에서 발생하는 취약점은 플랫폼 상(커널, 라이브러리)의 취약점, 변조된 앱(악성 앱)을 통한 취

※ 본 연구는 미래부가 지원한 2013년 정보통신·방송(ICT) 연구개발사업의 연구 결과로 수행되었음.
† 준 회 원 : 고려대학교 정보보호대학원 석사과정
†† 준 회 원 : 고려대학교 정보보호대학원 박사과정
††† 종신회원 : 고려대학교 정보보호대학원 정교수
논문접수 : 2013년 6월 5일
수 정 일 : 1차 2013년 8월 9일
심사완료 : 2013년 8월 12일
* Corresponding Author : Kim Seung Joo(skim71@korea.ac.kr)

약점으로 크게 2가지로 나눌 수 있으며, 취약점을 통한 정보 유출이 가장 큰 사회적 문제로 대두되고 있는 상황이다[1][2]. 이러한 취약점의 공통점은 결국 모바일 앱에서 사용하는 정보들을 유/노출 또는 악용 하는 것에 목적이 있다. 즉 모바일 앱에서 사용하는 정보의 관리가 보안 관점에서 제대로 수용 할 수 있는 범위라면 취약점에 노출되더라 하더라도 발생하는 위협을 최소화 할 수 있다.

본 논문에서는 안드로이드 스마트폰에서 발생하는 취약점들의 피해를 최소화 하기위해 안드로이드 모바일 앱에서 발생되는 잠재적인 취약사항을 검증하는 방안을 구현 단계별로 제안하며, 검증 방안을 통해 도출된 취약점에 따른 발생 원인 및 환경을 판단 할 수 있는 방법을 제안하고자 한다. 모바일 앱의 구현 단계인 분석/설계 단계에서부터 개발/테스트 단계까지 각각의 단계에 맞는 보안검증 방안을 제시하며, 각 단계에 맞는 적절한 방안을 통해 개발/테스트 단계에서 검증이 되지 않거나 검증되더라도 분석/설계를 다시 진행해야 하는 보안요구사항을 확인 할 수 있도록 하였다. 또한 단계별 보안검증을 통해 기존에 알려진 모든 안드로이드 앱의 취약점을 제거되어 구현이 완료되도록 하였다.

2. 안드로이드 모바일 앱의 위협현황

안드로이드 앱 동작 환경은 앱의 보안상 구조적/기능적 문제점과 함께 잠재적 위협의 발생 원인이 된다. 이러한 특성에 따라 안드로이드 앱의 개발환경, 앱의 서비스 환경, 앱의 이용환경으로 잠재적 위협의 발생 원인을 분류 할 수 있다. 이렇게 분류되는 보안 위협들은 설계 또는 개발 시점에서 발생하기도하며 앱이 서비스 되는 환경에서 발생하거나 사용자가 앱을 이용하는 단말기 환경에서 발생하기도 한다. 즉 Fig. 1의 형태로 각각의 보안 위협들은 다양한 환경에서 여러 가지 형태의 취약점으로 나타나고 있다[1][2].

2.1 리패키징 앱 제작 및 배포

안드로이드 앱은 악의적인 행위를 하는 코드를 삽입 한 후 리패키징 및 리사이닝 하여 임의로 배포가 가능하다. 이렇게 위/변조된 앱이 사용자에게 배포될 경우 사용자가 의도하지 않은 행위를 하게하며 사용자의 정보를 탈취하거나 자원 불법사용, 불법 과금을 발생 시킨다.

2.2 네트워크 스니핑/스푸핑

안드로이드 앱이 외부와 통신을 하는 기능을 포함하고 있다면 앱과 통신하는 시스템 간의 정보전달은 암호화되거나 안전한 방법으로 통신해야 한다. 그렇지 않다면 네트워크 구간의 스니핑 / 스푸핑 공격을 통해 중요정보가 노출되거나 패킷 재사용 등과 같은 네트워크상의 보안 문제점으로 전이 될 수 있다.

2.3 앱에서 사용되는 중요정보의 노출

안드로이드 앱은 실행되는 동안 사용되는 중요정보는

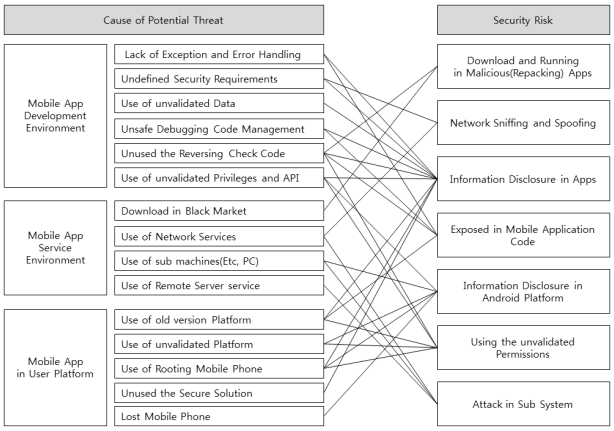


Fig. 1. Security Threats of Mobile App

IPC(Inter Process Communication)를 통해 전달되기도 하며 메모리상에 저장되기도 한다. 또한 앱의 4가지 구성요소(Activity, Content Provider, Service, Broadcast receiver)의 통신 수단인 인텐트를 통해 정보가 전달된다. 이러한 정보들은 권한 또는 앱의 구조적 문제로 인해 악의적인 사용자에게 노출 될 수 있다.

2.4 앱의 주요 구조/기능 노출

기본적으로 자바로 구현이 가능한 안드로이드 앱의 특성상 소스코드의 디컴파일이 가능하다. 이로 인해 안드로이드 앱을 소스코드 레벨에서 쉽게 분석 가능하게 하여 코드 상에 포함된 중요정보가 노출되며, 앱의 서비스 상 주요 구조/기능이 노출 될 수 있다.

2.5 단말기에 저장된 중요정보의 노출

안드로이드 단말기 내부에는 모바일 앱이 사용하는 데이터 파일, 데이터베이스 파일 등의 자원이 포함되어 있으며, 그 외에도 기본적으로 사용자의 전화번호목록, 메시지, 이메일 정보 등이 저장되어 있다. 이러한 자원들이 악의적인 행위를 하는 앱 또는 악성코드를 통해 특정 앱으로 전달되거나 네트워크를 통해 외부로 유출 될 수 있다.

2.6 부적절한 권한 도용 및 상승

안드로이드 앱이 서비스 하는 목적 및 기능에 불필요한 권한이 할당 되어 있거나 앱의 구현상의 문제점으로 인해 권한 상승이 된다면 이를 이용하여 부적절한 행위가 이루어지거나 권한도용 및, 중요한 정보가 외부로 노출 될 수 있다.

2.7 연동/연계 시스템 공격

안드로이드 앱이 서비스 되는 환경은 서버와 통신을 하거나, 개인PC에 설치된 프로그램과 연결되어 자원을 공유하기도 한다. 이러한 연동/연계되는 시스템에 대한 정보가 단말기에 쉽게 노출되거나 취약점을 통해 확인이 가능하다면 악의적인 사용자는 해당 시스템에 직접적으로 공격이 가능하다.

3. 기존 모바일 앱 취약점 검증 방안

현재 스마트폰 모바일 앱의 보안성을 검증하는 방안으로는 안전행정부와 한국정보보호진흥원에서 진행하는 1. “모바일 전자정부서비스 앱 검증 신청기관을 위한 앱 소스코드 보안성 검증” 방안과 금융감독원에서 시행하는 2. “스마트폰 금융 안전대책 이행실태 점검”을 통한 방안이 있다[3]. 이 두 가지 방안은 모두 구현이 완료된 앱을 대상으로 한다. 앱을 동작시키거나 개발 소스코드를 확인하는 방법으로 보안검증을 하고 있으며, 정리된 보안검증 체크리스트를 바탕으로 검증을 수행하는 방식이다.

3.1 모바일 전자정보서비스 앱 검증 신청기관을 위한 앱 소스코드 보안성 검증

“모바일 전자정부서비스 앱 소스코드 보안성 검증 체계”는 국가기관에서 개발되는 모바일 앱에 대해 소스코드 수준의 보안약점을 사전에 진단, 제거하여 모바일 전자정부서비스의 안전성, 신뢰성을 제고하기위해 진행되고 있다. 국가기관에서 개발되는 모바일 서비스를 대상으로 적용하고 있으며 현재는 중앙행정기관 및 지방자치단체에 한해서 검증이 시행되고 있다[3].

모바일 전자정부서비스 앱 소스코드에 대한 보안성 검증은 1) 소스코드 보안약점 2) 기능보안 취약점 이렇게 2개로 분류되며 각각 아래와 같은 점검 항목을 가진다[4][5].

1) 소스코드 보안약점

소스코드 보안약점의 경우 ″정보시스템 구축, 운영 지침(안전행정부고시 제2012-25호)″ 별표3. 소프트웨어 보안약점 기준(43개 항목) 및 소프트웨어 개발보안 가이드2) 등을 준수하여 구현언어 기반의 소스코드 보안약점과 공개영역인 CWE(Common Weakness Enumeration) 등을 통해 알려진 보안약점을 제거하도록 되어있다[4].

2) 기능보안 취약점

기능 보안취약점의 경우 해당 모바일 앱의 기능에 대해서 보안위협 및 보안취약점 점검하게 된다. ①적절한 기능 존재 유무, ②기능동작에 대한 최소권한 부여, ③외부 입력정보에 대한 유효성 검증, ④중요정보의 안전한 저장 및 관리, ⑤모바일 플랫폼 보안모델, ⑥사용 및 공개 모듈에 대한 안전성 보증, 이렇게 6가지로 구분되어 15가지의 세부항목을 통해 보안 취약점을 제거하도록 되어 있다[5].

3.2 스마트폰 금융 안전대책 이행실태 점검

금융감독원은 스마트 금융서비스를 제공하는 금융회사(은행, 카드, 보험, 증권 등)를 대상으로 안전대책 이행 실태점검이 매년 진행하고 있다. ①스마트폰 금융보안대책, ②앱 위/변조 방지대책, ③기타, 이렇게 3가지 영역으로 구분 후 13가지의 점검항목을 토대로 16가지의 세부 내용에 대해 점검을 실시한다. 특히 백신프로그램이나 보안키패드와 같은 입력정보 보호기능 등 스마트폰 금융 보안대책을 제대로 적용했는지, 스마트폰 임의 개조를 막을 대책은 있는지, 그 외 취약점은 없는지를 중점적 점검하고 있다.

4. 안드로이드 모바일 앱 구현시 단계별 보안검증 방안 제시

모바일 앱 보안위협에 대응하기 위해 앞에서 확인한 두 가지 방법은 이미 구현이 완료된 앱을 대상으로 보안검증을 진행하고 있어 보안취약점이 존재할 경우 코드 수정을 하거나 설계 단계부터 다시 구현을 해야 하기 때문에 비용 및 시간적인 면에서 효율적이지 못하다. 또한 개발 소프트웨어를 모두 포함하는 소스코드 보안검증 항목이기 때문에 안드로이드 모바일 앱의 소스코드의 특징을 고려한 점검 방안이 아니라고 할 수 있으며, 보안성 검증 대상이 되는 범위가 공공기간이나 금융권 서비스에 한정적으로 맞춰져 있기 때문에 점검 리스트의 기준이 제한적이라고 할 수 있다. 이러한 문제점에 대해 안드로이드 모바일 앱 구현시 각 구현 단계별로 보안검증 방안을 제시하여 위에 언급된 문제점을 보완하는 것은 물론, 각 취약점 별 VCG[Vulnerability Cause Graphs][6]을 도출하여 취약사항에 대해 발생원인 및 환경을 쉽게 판단하도록 하며 현재까지 알려진 모든 안드로이드 모바일 앱 취약점을 검증 할 수 있도록 하고자 한다.

4.1 분석/설계 단계에서의 보안검증 방안

안드로이드 앱 구현시 일반적으로 분석/설계 단계에서는 모바일 앱의 기능 및 UI, 그리고 데이터 프로세스 및 흐름에 대하여 요구사항을 분석하고 그에 맞게 설계하고 정의한다. 이때 모바일 앱이 구동되는 환경과 서비스에 따라 보안 요구사항을 정의하고 그에 따른 보안 기능의 설계가 필요하다. Table 1에서 제시하는 보안검증 항목을 바탕으로 보안 요구사항을 정의하거나 구현되는 기능에 대해 잠재적 보안 취약성을 사전에 제거 할 수 있다[7].

4.2 개발 단계에서의 보안검증 방안

안드로이드 앱 구현시 개발 단계에서는 분석/설계 단계에서 정의한 모바일 앱의 기능 및 UI, 그리고 데이터 프로세스 및 흐름에 대해 실제 코드를 작성하고 환경을 구성하게 된다. 이때 코드 작성 시 적용해야 할 보안로직 및 기능 구현에 관련 코드 작성이 필요하다. 실제 작성한 코드 및 환경을 기반으로 Table 2의 보안검증 방안을 확인하여 코드 상의 보안성을 검증함으로써 코드 상에서 도출되는 잠재적 보안 취약성을 사전에 제거 할 수 있다[8].

4.3 테스트 단계에서의 보안검증 방안

테스트 단계에서는 실제 구현된 안드로이드 앱의 동작 및 서비스 환경을 테스트한다. 이때 구현이 완료된 앱을 대상으로 앱이 동작하는 적절한 환경에 맞춰 실제 보안 점검이 필요하다. Table 3에서 제시하는 테스트 단계에서의 보안점검 항목을 바탕으로 실제로 테스트를 하여 도출되는 취약사항을 제거 할 수 있다[7~9].

Table 1. List of Security Verification in Analysis and Design Process

Type	Code	Security Verification List	CWE
[A-1] Adequate storage of important resources (data)	A-1.1	Verification of storage section according to logic and level of importance for stored data in mobile device and server	/
	A-1.2	Verification of storage of important information in DB files and password adequacy	CWE-312
	A-1.3	Verification of storage of important information in data files and password adequacy	CWE-312
	A-1.4	Verification of storage of important information in external storage device (SD card) and password adequacy	/
[A-2] Restricted access to important resources (data)	A-2.1	Adequacy of access control to communication resources in Android such as IPC and Intent	CWE-732
	A-2.2	Assignment of UID control/management for sharing of resources between apps	CWE-732
[A-3] Adequate resource (data) control over transmission section	A-3.1	Implementation of security network (SSL, HTTPS, SFTP, etc.) over transmission section	CWE-319
[A-4] Adequacy of app authorization and resource reference	A-4.1	Unnecessary use of app authorization	CWE-250
	A-4.2	Adequate use of Open Library/API code	CWE-250
	A-4.3	Adequate reference and loading for resources and codes	CWE-829
[A-5] Adequacy of cryptograph	A-5.1	Use of securely encrypted algorithm	CWE-327
	A-5.2	Adequacy of password key length	CWE-327
	A-5.3	Use of cryptographically secure random number	CWE-327
	A-5.4	Use of cryptographically secure hash	CWE-327 CWE-759
[A-6] Adequacy of login authentication logic	A-6.1	Implementation of login authentication control rules	CWE-307
	A-6.2	Implementation of adequate password combination rules	/
	A-6.3	Implementation of secure lifecycle for cookies/sessions and encryption	CWE-306 CWE-863
[A-7] Check the Reverse engineering and integrity verification	A-7.1	Implementation of app integrity check	/
	A-7.2	Implementation of code obfuscation	/
[A-8] Handling of Error Message and log management	A-8.1	Adequate exception handling during errors	/
	A-8.2	Implementation of user input value confirmation logic	/
	A-8.3	Log management and log pattern implementation	/
[A-9] Distribution and update, signature management	A-9.1	Safe app update and adequate security measures for in-app billing	/
	A-9.2	Distribution through official Android market	/

Table 2. List of Security Verification in Implementation Process

Type	Code	Security Verification List	CWE	Development Type		
				Mobile WEB	Hybrid APP	Mobile APP
[D-1] Code verification for app authorization and use	D-1.1	Adequacy of app authorization in AndroidManifest.xml	CWE-250	X	O	O
	D-1.2	Adequacy of integrity verification code for running apps	/	X	O	O
[D-2] Code verification for app resource access control	D-2.1	Sharing of adequate UID through SharedUserId	CWE-732	X	O	O
	D-2.2	Use of adequate mode during IPC file generation	CWE-732	X	O	O
	D-2.3	Adequate use of broadcast Intent	CWE-829	X	O	O
[D-3] Code verification for input resource (data) filtering	D-3.1	Verification of unintended input values including special characters	CWE-22 CWE-807 CWE-862	O	O	O
	D-3.2	Verification of SQL query insertion	CWE-89 CWE-807	O	O	O
	D-3.3	Verification XSS script insertion	CWE-79 CWE-352 CWE-807	O	O	O
	D-3.4	Verification of command insertion	CWE-78 CWE-807	O	O	O
	D-3.5	Availability of file upload option	CWE-434	O	O	X
	D-3.6	Availability of file download option	CWE-494	O	O	X
	D-3.7	Availability of URL redirection	CWE-601 CWE-862	O	O	X
[D-4] Verification of Native code and Library code	D-4.1	Availability of memory management/control	CWE-120 CWE-131 CWE-134 CWE-190	X	O	O
	D-4.2	Use of vulnerable library/API	CWE-676	X	O	O
[D-5] Code verification for transmitted resources (data)	D-5.1	Use of secure network transmission code during HTTP protocol use	CWE-319	O	O	O
	D-5.2	Use of secure socket communication code	CWE-319	O	O	O
	D-5.3	Use of Localhost network port	CWE-319	X	O	O
[D-6] Verification of information exposure in source code	D-6.1	Deletion of debugging code during distribution	/	X	O	O
	D-6.2	Hard coding of important information	CWE-798	O	O	O
	D-6.3	Comment code the important information	CWE-798	O	O	O
[D-7] Information disclosure in code	D-7.1	Adequacy of error codes	/	O	O	O
[D-8] Code verification for use of encrypted algorithm	D-8.1	Use of cryptographically secure random number generation code	CWE-327	X	O	O
	D-8.2	Use of cryptographically secure password key generation code	CWE-327	X	O	O
	D-8.3	Use of cryptographically secure hash code	CWE-327 CWE-759	X	O	O
	D-8.4	Use of KeyStore mechanism during password key storage	CWE-327	X	O	O

Table 3. List of Security Verification in Testing Process

APP Status	Type	Code	Security Verification List
Static status app check	[V-1] Handling of important information	V-1.1	Plain text storage of important information in file data
		V-1.2	Plain text storage of important information in DB data
		V-1.3	Plain text storage of important information in SDCARD
	[V-2] File access authorization	V-2.1	Inappropriate permission of file data
	[V-3] Virus / Malware Check	V-3.1	Use of security process (solution)
	[V-4] Phone hacking (rooting) identification	V-4.1	Check the Hacking(rooting) Mobile device
	[V-5] Check the Reverse engineering and code identification	V-5.1	Use of source code Obfuscation
		V-5.2	Storage of important information in source code
		V-5.3	Abuse of resource data
		V-5.4	Checking of app forgery
Dynamic status app check	[V-6] Safe use of important information	V-6.1	Plain text transmission of important information over communication section
		V-6.2	Information exposure via Intent
	[V-7] Debugging information	V-7.1	Log storage of important information and checking for excessive log storage
	[V-8] Authorization abuse	V-8.1	Inappropriate use of app authorization
		V-8.2	Abuse of authorization via network
	[V-9] Input value verification	V-9.1	Use of filtering for unsatisfactory input value
		V-9.2	Execution of random input code and command
		V-9.3	Exposure of error message

5. 보안성 검증 방안의 실효성 및 구현 단계별 연관관계 확인

앞에서 제시한 구현 단계별 보안검증 방안은 일반적인 소프트웨어 취약점 분류체계인 CWE/SANS Top 25를 기준으로 모바일 앱 구현 단계별 환경에 맞게 제시하였으며 최종적으로 모바일 환경에 맞는 도출된 취약점의 원인 및 환경을 분석 할 수 있다. 즉, 각각의 취약점 별 VCG[6]을 나타내어 분석/설계 단계 및 개발/테스트 단계별 연관관계를 확인 할 수 있다.

현재까지 CVE를 통해 알려진 안드로이드 관련 취약점은 349개이며 그중 안드로이드에서 실행되는 모바일 앱 자체에서 도출된 취약점은 총 60개 이다. 이렇게 알려진 취약점 60개와 모바일 앱 환경에 따른 잠재적인 취약사항 5개를 모두 포함하여 9가지 항목으로 구분 지을 수 있으며 세부적으로 15가지의 취약점 항목으로 나누어 각 항목별 VCG를 정의할 수 있다[6], Table 4.

5.1 중요정보 취급상태(식별코드 V-1, V-2)

단말기 내부에 저장되는 데이터 파일의 저장방식 및 접근권한을 확인하여 중요정보에 대한 유출 가능성을 체크한다. 이때 분류되는 저장 공간 및 방식은 DB(Database)파일, 일반 데이터 파일(XML, TXT 등), 그리고 외부저장장치 인 SDCARD(Secure Digital CARD)로 분류되며 각 파일에 부여된 접근권한에 따라 악성 앱에서 접근, 연결되는 단말기(PC)에서의 접근, 그리고 루팅된 단말기에서의 환경적 요인으로 인한 임의 접근으로 나눌 수 있다. “단말기 내 저장된 정보유출 취약점”의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 2, Table 5와 같다.

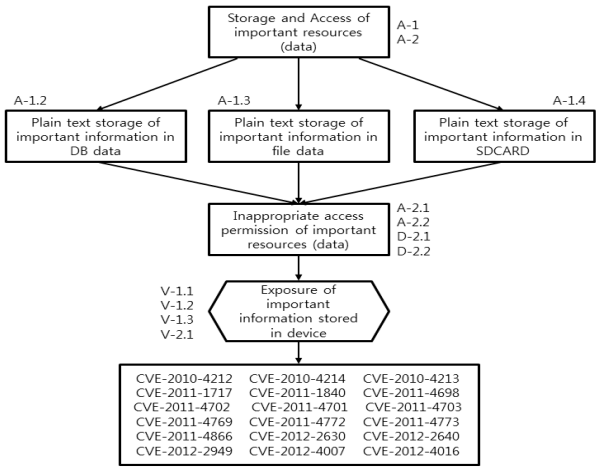


Fig. 2. VCG of “File Information handling Threat Vulnerability”

Table 4. Vulnerability List of Android Mobile App

App Status	Code	Type	Vulnerable List
Static environment	V-1 V-2	Handling of important information	Exposure of important information stored in device
	V-3 V-4	Maintenance of security process	Execution on Rooting devices
			Lack of security process
	V-5	Source code identification	Exposure of important information in source code
			Lack of source code Obfuscation
	V-5	Check the Reverse engineering	App forgery
			Abuse of resource data
Dynamic environment	V-6	Utilization of important information	Information exposure via intent
	V-6 V-8	Handling of network information	Exposure of important information via network
			Abuse of authorization over network
	V-7	Debugging information	Exposure of important information in debug log
	V-8	App authorization abuse	Inappropriate use of app authorization
	V-9	Input value verification and Inappropriate code execution	Execution of code and command
			Possibility of DoS attacks
			Error message exposure

Table 5. Security Verification Flow Table in Mobile App development Process of "File Information handling Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Storage of important resources (data)	A-1.1 A-1.2 A-1.3 A-1.4
	Restricted access to important resources (data)	A-2.1 A-2.2
Development Process	App authorization and resource reference	D-2.1 D-2.2
Test Process	Handling of important information	V-1.1 V-1.2 V-1.3
	File access authorization	V-2.1

Table 6. Security Verification Flow Table in Mobile App development Process of "Unused Security Platform and Programs Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Handling of Error Message and log management	A-8.2
Development Process	Code verification for input resource (data) filtering	D-3
Test Process	Virus / Malware Check	V-3.1
	Check the Hacking(rooting) Mobile device	V-4.1

5.2 최소한의 보안프로세스 유지(식별코드 V-3, V-4)

금융 서비스(상품결제, 계좌이체 등)를 제공하는 모바일 앱의 경우 계좌번호, 인증서 비밀번호 등의 중요정보를 입력하게 된다. 이때 보안키보드 및 실시간 백신 탐지 기능이 실행되고 있지 않을 경우 악성 앱에 의해 중요정보가 유출되는 잠재적 취약점이 존재한다. 특히 검증되지 않는 플랫폼(루팅된 안드로이드 단말기, 커스터마이징 된 안드로이드 OS 등)을 사용하고 있을 경우 이러한 잠재적 위협이 더욱 커지게 되므로 금융정보 등의 사용자 중요정보에 대한 보호를 위해 최소한의 보안프로세스를 유지하는 것이 중요하다. 이에 해당하는 "보안프로세스 미사용 취약점", "루팅된 단말기에서의 실행 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 3, Table 6와 같다.

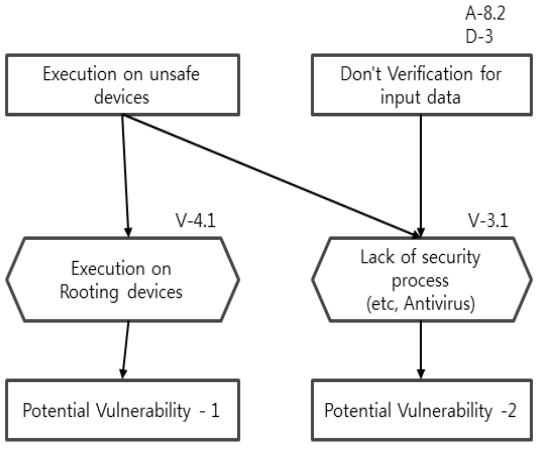


Fig. 3. VCG of "Unused Security Platform and Programs Vulnerability"

5.3 소스코드 식별(식별코드 V-5)

자바로 개발되는 안드로이드 앱의 경우 자바코드 특성에 따라 소스코드 디컴파일이 가능하다. 이때 디컴파일 후 소스코드 상에 포함되어 있는 중요정보가 노출될 가능성의 잠재적 취약점이 존재하며, 이를 위해 소스코드의 난독화 적용이 필요하다. 이에 해당하는 "코드 내 작성된 중요정보 노출 취약점" 및 "소스코드 난독화 미적용 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 4, Table 7와 같다.

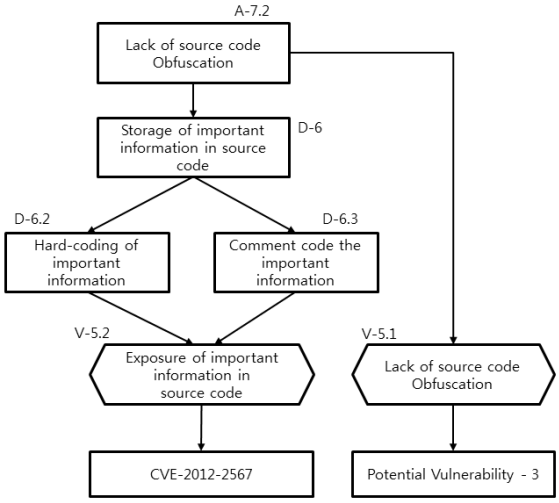


Fig. 4. VCG of "Source Code Decompile Vulnerability"

Table 7. Security Verification Flow Table in Mobile App development Process of "Source Code Decompile Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Check the Reverse engineering and integrity verification	A-7.2
Development Process	Information disclosure in code	D-6.2 D-6.3
Test Process	Check the Repacking App and Code verification	V-5.1 V-5.2

5.4 역공학 대응(식별코드 V-5)

기본적으로 역공학이 가능한 안드로이드 앱의 경우 사용되는 리소스 자원의 노출과 같은 잠재적 취약점이나 리패키징을 통한 악성코드 삽입 등의 위험에 노출되어 있다. 이러한 위협에 대해 안드로이드 앱 실행시 해당 앱의 무결성 검증에 대한 보안로직이 필요하며, html 코드 또는 스크립트 등의 독립 실행이 가능한 파일자원 로직을 앱 실행 바이너리에 포함시킬 수 있도록 구현하거나 서버 작동 로직으로 포함시켜 독립적인 실행에 대한 보안조치가 필요하다. "앱 위/변조 취약점" 및 "리소스 자원 악용 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 5, Table 8와 같다.

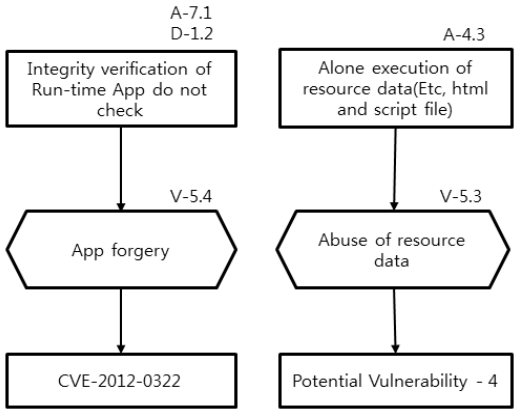


Fig. 5. VCG of "Reverse Engineering Vulnerability"

Table 8. Security Verification Flow Table in Mobile App development Process of "Reverse Engineering Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Adequacy of app authorization and resource reference	A-4.3
	Check the Reverse engineering and integrity verification	A-7.1
Development Process	Inappropriate use of app authorization	D-1.2
Test Process	Check the Repacking App and Code verification	V-5.3 V-5.4

5.5 네트워크 정보취급(식별코드 V-6, V-8)

안드로이드 앱과 서버 간에 네트워크를 통한 정보의 송수신에 대해 중요정보 유출 가능성 및 네트워크 자원(세션, 포트 번호 등)의 악용 가능성을 확인한다. 이때 사용되는 네트워크 프로토콜에 따라 평문으로 송수신이 가능한 프로토콜(http, ftp 등)의 경우 중요정보가 노출될 가능성이 존재하며, 서버와 통신시 사용되는 세션 및 포트 등의 자원에 대한 보안관리가 정상적으로 되고 있지 않을 경우 권한을 도용하거나 중요정보에 대한 임의의 접근이 가능하다. 이에 따른 "네트워크를 통한 중요정보 노출 취약점" 및 "네트워크를 통한 권한 도용 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 6, Table 9와 같다.

5.6 중요정보 활용(식별코드 V-6)

안드로이드 앱은 각 컨포넌트 자원간의 통신을 위해 인텐트를 사용하게 된다. 이때 사용되는 인텐트 정보를 가로채거나 조작하여 중요정보를 획득 가능성을 확인한다. 앱이 인텐트를 사용할 경우 중요정보를 인텐트를 통해 다루지 말아야 하며, 부득이 할 경우 브로드캐스트 방식의 인텐트 사용은 신중하게 설계해야 한다. 이러한 "인텐트를 이용한 정보유출 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 7, Table 10와 같다.

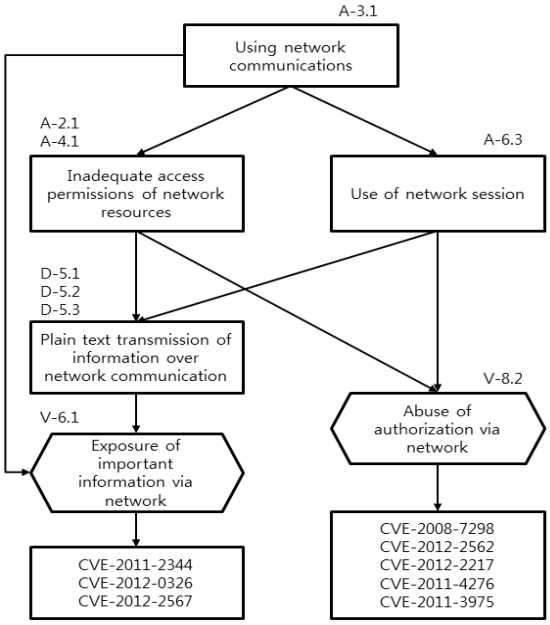


Fig. 6. VCG of "Network flow Information Vulnerability"

Table 9. Security Verification Flow Table in Mobile App development Process of "Network flow Information Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Adequate resource (data) control over transmission section	A-3.1
	Restricted access to important resources (data)	A-2.1
	Adequacy of app authorization and resource reference	A-4.1
	Adequacy of login authentication logic	A-6.3
Development Process	Code verification for transmitted resources (data)	D-5.1 D-5.2 D-5.3
Test Process	Handling of important information	V-6.1
	Authorization abuse	V-8.2

5.7 디버그 정보(식별코드 V-7)

안드로이드 앱의 경우 개발 시 디버깅을 위해 디버깅 코드(Log Class)를 사용한다. 이후 개발이 완료된 시점에서 디버깅 코드로 사용한 Log Class의 삭제 없이 컴파일 및 배포가 된다면 단말기 내부에는 해당 앱의 디버깅 코드가 실행되게 된다. 이때 디버깅 코드에 개인정보와 같은 중요정보 및 서버 인증 정보, 앱 로직상의 중요정보 등이 포함되어 있다면 악성 앱에 의해 손쉽게 외부로 유출이 가능하다. 이러한 취약사항에 해당하는 "디버깅 로그에서 중요정보 노출 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 8, Table 11와 같다.

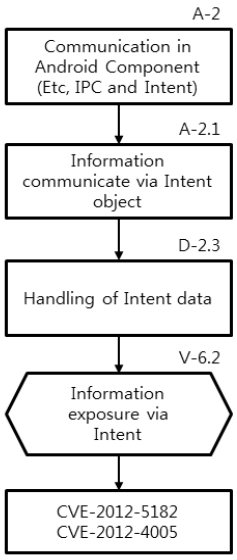


Fig. 7. VCG of "Intent Handling Vulnerability"

Table 10. Security Verification Flow Table in Mobile App development Process of "Intent Handling Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Restricted access to important resources (data)	A-2.1
Development Process	Code verification for App resources (data)	D-2.3
Test Process	Safe use of important information	V-6.2

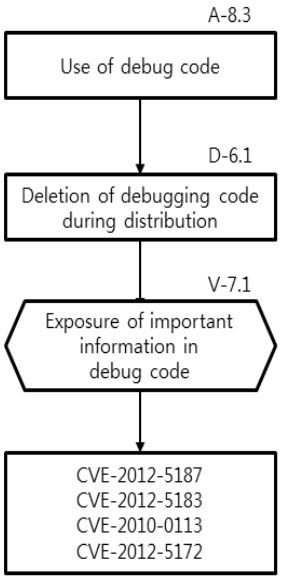


Fig. 8. VCG of "Undeleted Debugging Code Vulnerability"

Table 11. Security Verification Flow Table in Mobile App development Process of "Undeleted Debugging Code Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Handling of Error Message and log management	A-8.3
Development Process	Information disclosure in code	D-6.1
Test Process	Debugging information	V-7.1

5.8 입력 값 검증 및 부적절한 코드 실행(식별코드 V-9)

사용자의 입력 값을 통해 특정 기능을 실행하거나 정보를 제공하는 앱의 경우 사용자의 입력 값에 대한 검증이 필요하다. 임의의 코드 및 스크립트 코드가 앱 내부에 삽입되거나 앱과 연동되는 서버로 유입될 경우 악성 코드가 실행되거나 서비스에 대한 DoS 공격, 그리고 에러메시지 노출의 잠재적 위협에 노출된다. 특히 웹서버와 연동되는 앱의 경우 이러한 문제점을 반드시 검증해야 한다. 이에 해당하는 "코드 및 명령어 실행 취약점", "DoS 공격 취약점", "에러메시지 노출 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 9, Table 12와 같다.

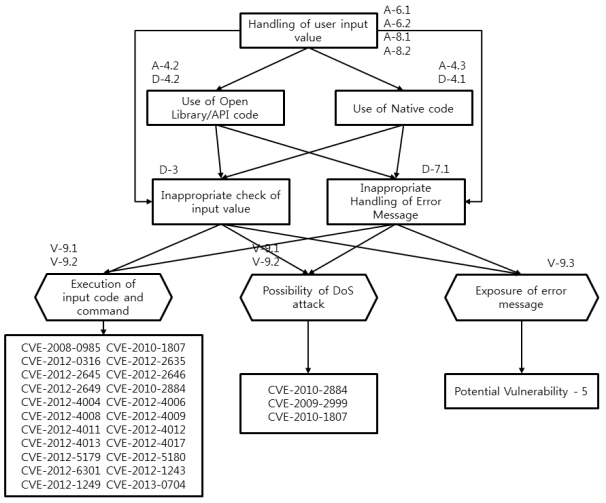


Fig. 9. VCG of "Input data Check and Malicious Code execute Vulnerability"

5.9 앱 권한 오남용(식별코드 V-8)

사용되는 앱의 원래 목적인 특정 서비스의 권한보다 과도한 권한을 포함하거나 사용되는 앱 기능이 악성 앱 또는 사용자에 의해 악용 될 수 있다면 앱 서비스의 신뢰성 하락 및 권한 악용에 대한 문제점이 발생 할 수 있다. 이에 따라 AndroidManifest.xml 파일에 부여된 권한의 적정성을 확인하고, 부여된 기능의 악용 가능성을 점검해야 한다. 이러한 취약사항에 해당하는 "앱 권한의 부적절한 사용 취약점"의 VCG에 따른 모바일 앱 구축 단계별 취약 원인 및 환경 흐름은 Fig. 10, Table 13와 같다.

Table 12. Security Verification Flow Table in Mobile App development Process of"Input data Check and Malicious Code execute Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Adequacy of login authentication logic	A-6.1 A-6.2
	Handling of Error Message and log management	A-8.1 A-8.2
Development Process	Code verification for input resource (data) filtering	D-3.1 D-3.2 D-3.3 D-3.4 D-3.5 D-3.6 D-3.7
	Verification of Native code and Library code	D-4.1 D-4.1
	Adequacy of error codes	D-7.1
Test Process	Input value verification	V-9.1 V-9.2 V-9.3

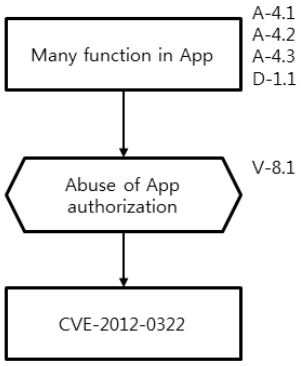


Fig. 10. VCG of "Permission abuse Vulnerability"

Table 13. Security Verification Flow Table in Mobile App development Process of "Permission abuse Vulnerability"

Process	Security Verification	Code
Analysis / Design Process	Adequacy of App authorization and resource reference	A-4.1 A-4.2 A-4.3
Development Process	Inappropriate use of App authorization	D-1.1
Test Process	Authorization abuse	V-8.1

6. 모바일 앱 보안검증 방안 비교

상위 목차인 "4. 안드로이드 모바일 앱 구현시 단계별 보안검증 방안 제시", "5. 보안검증 방안의 실효성 및 구현 단계별 연관관계 확인"에서 제안한 모바일 앱 보안검증 방안의 실효성을 입증 하기위해 기존의 모바일 앱의 보안성을

Table 14. Check List of Android App Vulnerability

Type	Vulnerable Checklist
Handling of important information	Exposure of important information stored in Mobile device
	Inappropriate permission of file data
Maintenance of security process	Lack of security process
	Check the Hacking(rooting) Mobile device
Check the Reverse engineering and code identification	Lack of source code Obfuscation
	Storage of important information in source code
	Abuse of resource data
	Checking of app forgery
Safe use of important information	Plain text transmission of important information over communication section
	Information exposure via Intent
Debugging information	Log storage of important information and checking for excessive log storage
Authorization abuse	Inappropriate use of app authorization
	Abuse of authorization via network
Input value verification	Use of filtering for unsatisfactory input value
	Execution of random input code and command
	Exposure of error message

검증하는 방안인 “모바일 전자정부서비스 앱 검증 신청기관을 위한 앱 소스코드 보안성 검증”, “스마트폰 금융 안전대책 이행실태 점검” 방안 2가지를 기반으로 실제 사용되고 있는 모바일 앱의 보안검증을 실시하여 모바일 앱의 취약점을 도출 후 비교, 분석을 진행한다. 이때 사용되는 점검 취약점 항목 리스트는 “5. 보안검증 방안의 실효성 및 구현 단계별 연관관계 확인”을 통해 나온 과거 모든 취약점을 포함하는 취약점 목록 리스트와 VCG를 통해 확인된 잠재적 취약점 목록 리스트를 모두 포함하여 모바일 앱에서 도출 될 수 있는 모든 취약점을 수렴하도록 한다[3~8], Table 14.

6.1 기존 방안별 모바일 앱 보안 점검

모바일 앱 보안 점검은 ‘금융권 모바일 앱’, ‘공기업 모바일 앱’, 그리고 ‘일반 기업 모바일 앱’ 3가지를 대상으로 한다.

1) “모바일 전자정부서비스 앱 소스코드 보안성 점검”을 기반으로 한 보안 점검 결과

Table 14에서 정의한 모바일 앱 취약점 항목을 기준으로 “모바일 전자정부서비스 앱 소스코드 보안성 점검”의 경우 점검항목에 해당하지 않는 항목이 3개가 존재하며 이를 고려하여 진행한 보안 점검 결과는 일반기업 모바일 앱(A사 모바일 앱)의 경우 양호 10개, 취약 3개의 결과가 나왔으며, 금융권 기업 모바일 앱(B사 모바일 앱)은 양호 9개, 취약 4개이다. 그리고 공기업 모바일 앱(C사 모바일 앱)의 경우 양호 10개, 취약 3개로 나타났다(Table 15).

2) “스마트폰 금융 안전대책 이행실태 점검”을 기반으로 한 보안 점검 결과

Table 14에서 정의한 모바일 앱 취약점 항목을 기준으로 “스마트폰 금융 안전대책 이행실태 점검”의 경우 점검항목에 해당되지 않는 항목이 4개가 존재하며 이를 고려하여 진행한 보안 점검 결과는 일반기업 모바일 앱(A사 모바일 앱)의 경우 양호 10개, 취약 2개의 결과가 나왔으며, 금융권 기업 모바일 앱(B사 모바일 앱)은 양호 9개, 취약 3개이다. 그리고 공기업 모바일 앱(C사 모바일 앱)의 경우 양호 9개, 취약 3개로 나타났다(Table 16).

3) 제안하는 “모바일 앱 구현 단계별 보안성 점검” 방안을 이용한 모바일 앱 보안 점검

Table 14에서 정의한 모바일 앱 취약점 항목을 기준으로 “모바일 앱 구축 단계별 보안성 점검” 방안은 현재까지 알려진 모든 모바일 앱 취약점을 수렴하고 있으며 잠재적 취약사항까지 점검이 가능하다(Table 17).

6.2 각 방안별 점검 결과 비교

각 보안성 점검 방안별 점검 결과 “모바일 전자정부서비스 앱 소스코드 보안성 점검” 방안과 “스마트폰 금융 안전대책 이행실태 점검” 방안의 경우 점검되지 않는 항목이 존재하며, 그에 따라 Table 18과 같이 점검 결과 또한 상이한 것을 알 수 있다.

Table 15. Security Verification Result of Android Mobile App - 1

Type	Vulnerable Checklist	A Company Mobile App	B Company Mobile App	C Company Mobile App
Handling of important information	Exposure of important information stored in device	secure	**Vulnerable**	secure
	Inappropriate permission of file data	secure	secure	secure
Maintenance of security process	Lack of security process	**Not checked**	**Not checked**	**Not checked**
	Check the Hacking(rooting) Mobile device	**Vulnerable**	**Vulnerable**	secure
Check the Reverse engineering and code identification	Lack of source code Obfuscation	**Vulnerable**	**Vulnerable**	**Vulnerable**
	Storage of important information in source code	secure	secure	secure
	Abuse of resource data	**Not checked**	**Not checked**	**Not checked**
	Checking of app forgery	**Not checked**	**Not checked**	**Not checked**
Safe use of important information	Plain text transmission of important information over communication section	secure	secure	**Vulnerable**
	Information exposure via Intent	secure	secure	secure
Debugging information	Log storage of important information and checking for excessive log storage	**Vulnerable**	**Vulnerable**	**Vulnerable**
Authorization abuse	Inappropriate use of app authorization	secure	secure	secure
	Abuse of authorization via network	secure	secure	secure
Input value verification	Use of filtering for unsatisfactory input value	secure	secure	secure
	Execution of random input code and command	secure	secure	secure
	Exposure of error message	secure	secure	secure

Table 16. Security Verification Result of Android Mobile App - 2

Type	Vulnerable Checklist	A Company Mobile App	B Company Mobile App	C Company Mobile App
Handling of important information	Exposure of important information stored in device	secure	**Vulnerable**	secure
	Inappropriate permission of file data	secure	secure	secure
Maintenance of security process	Lack of security process	secure	secure	secure
	Check the Hacking(rooting) Mobile device	**Vulnerable**	**Vulnerable**	양호
Check the Reverse engineering and code identification	Lack of source code Obfuscation	**Vulnerable**	**Vulnerable**	**Vulnerable**
	Storage of important information in source code	secure	secure	secure
	Abuse of resource data	**Not checked**	**Not checked**	**Not checked**
	Checking of app forgery	secure	secure	**Vulnerable**
Safe use of important information	Plain text transmission of important information over communication section	secure	secure	**Vulnerable**
	Information exposure via Intent	**Not checked**	**Not checked**	**Not checked**
Debugging information	Log storage of important information and checking for excessive log storage	**Not checked**	**Not checked**	**Not checked**
Authorization abuse	Inappropriate use of app authorization	**Not checked**	**Not checked**	**Not checked**
	Abuse of authorization via network	secure	secure	secure
Input value verification	Use of filtering for unsatisfactory input value	secure	secure	secure
	Execution of random input code and command	secure	secure	secure
	Exposure of error message	secure	secure	secure

Table 17. Security Verification Result of Mobile App development Process based

Type	Vulnerable Checklist	A Company Mobile App	B Company Mobile App	C Company Mobile App
Handling of important information	Exposure of important information stored in device	secure	**Vulnerable**	secure
	Inappropriate permission of file data	secure	secure	secure
Maintenance of security process	Lack of security process	secure	secure	secure
	Check the Hacking(rooting) Mobile device	**Vulnerable**	**Vulnerable**	secure
Check the Reverse engineering and code identification	Lack of source code Obfuscation	**Vulnerable**	**Vulnerable**	**Vulnerable**
	Storage of important information in source code	secure	secure	secure
	Abuse of resource data	**Vulnerable**	secure	secure
	Checking of app forgery	secure	secure	**Vulnerable**
Safe use of important information	Plain text transmission of important information over communication section	secure	secure	**Vulnerable**
	Information exposure via Intent	secure	secure	secure
Debugging information	Log storage of important information and checking for excessive log storage	**Vulnerable**	**Vulnerable**	**Vulnerable**
Authorization abuse	Inappropriate use of app authorization	secure	secure	secure
	Abuse of authorization via network	secure	secure	secure
Input value verification	Use of filtering for unsatisfactory input value	secure	secure	secure
	Execution of random input code and command	secure	secure	secure
	Exposure of error message	secure	secure	secure

Table 18. Security Verification Comparison

	Security Checklist Type	Verification List available	Verification results
A Company Mobile App	Security Verification Guide for Mobile e-Government service App Source code	13/16	Secure(10), Vulnerable(3)
	Security Verification of SmartPhone Finance App	12/16	Secure(10), Vulnerable(2)
	Security Verification Result of Mobile App development Process based	16/16	Secure(12), Vulnerable(4)
B Company Mobile App	Security Verification Guide for Mobile e-Government service App Source code	13/16	Secure(9), Vulnerable(4)
	Security Verification of SmartPhone Finance App	12/16	Secure(9), Vulnerable(3)
	Security Verification Result of Mobile App development Process based	16/16	Secure(12), Vulnerable(4)
C Company Mobile App	Security Verification Guide for Mobile e-Government service App Source code	13/16	Secure(10), Vulnerable(3)
	Security Verification of SmartPhone Finance App	12/16	Secure(9), Vulnerable(3)
	Security Verification Result of Mobile App development Process based	16/16	Secure(12), Vulnerable(4)

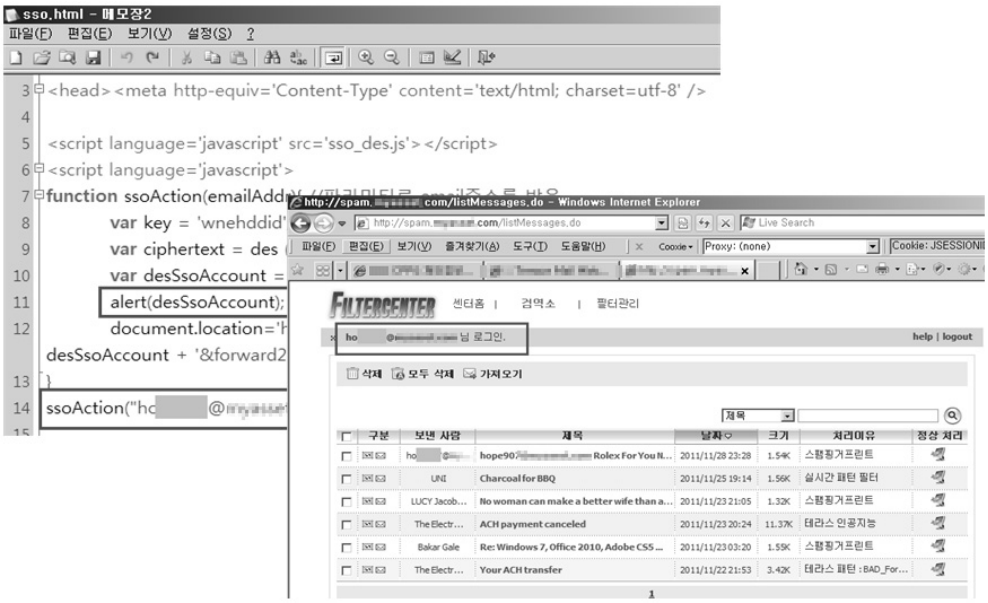


Fig. 11. Abuse of resource data Vulnerability

Fig. 12. Undeleted debug code Vulnerability

1) 일반기업 모바일 앱

일반기업 모바일 앱은 “모바일 전자정부서비스 앱 소스코드 보안성 점검” 방안과 “스마트폰 금융 안전대책 이행실태 점검” 방안으로 보안성 점검을 진행 할 경우 ‘리소스 자원 악용 가능성’ 취약점을 도출 하지 못한다. 이 취약점은 “모바일 앱 구축 단계별 보안성 점검” 방안으로 Fig. 11과 같이 취약점 도출이 가능하다.

2) 금융권 모바일 앱

금융권 모바일 앱의 경우 “스마트폰 금융 안전대책 이행실태 점검” 방안으로 보안성 점검을 진행 할 경우 ‘중요정보 로그 저장 및 과다한 로그 저장 여부 확인’ 취약점을 도출 하지 못한다. 이 취약점은 “모바일 앱 구축 단계별 보안성 점검” 방안으로 Fig. 12와 같이 취약점 도출이 가능하다.

3) 공기업 모바일 앱

공기업 모바일 앱의 경우 “모바일 전자정부서비스 앱 소스코드 보안성 점검” 방안으로 보안성 점검을 진행 할 경우 ‘앱 위/변조 체크 가능 여부 확인’ 취약점을 도출 하지 못한다. 또한 “스마트폰 금융 안전대책 이행실태 점검” 방안으로 보안성 점검을 진행 할 경우 ‘중요정보 로그 저장 및 과다한 로그 저장 여부 확인’ 취약점을 도출 하지 못한다. 이 두 가지 취약점은 “모바일 앱 구축 단계별 보안성 점검” 방안으로 Fig. 13, Fig. 14와 같이 취약점 도출이 가능하다.

Fig. 13. Undeleted debug code Vulnerability

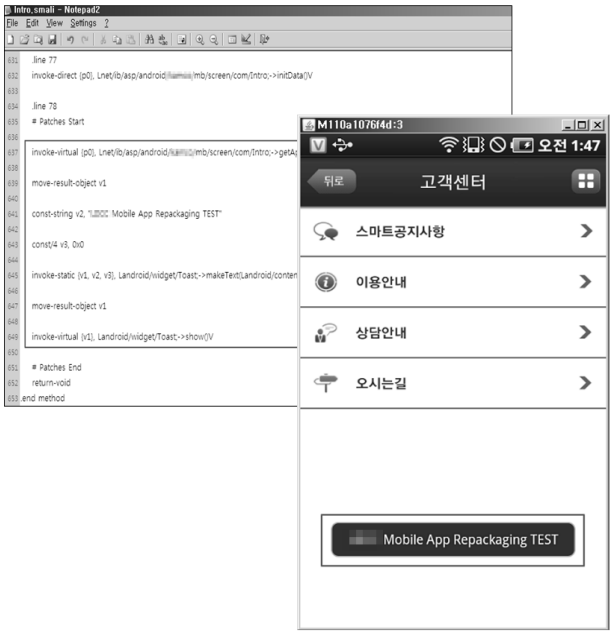


Fig. 14. App Repackaging Vulnerability

7. 결 론

본 논문에서 제안하는 “모바일 앱 구현 단계별 보안검증” 방안의 경우 안전행정부와 한국정보보호진흥원에서 진행하는 “모바일 전자정부서비스 앱 검증 신청기관을 위한 앱 소스코드 보안성 검증” 방안과 금융감독원에서 시행하는 “스마트폰 금융 안전대책 이행실태 점검 체크리스트”를 이용하는 방안과는 다르게 앱 제작 주체(기업, 개인 등)의 성격이나 앱 서비스 방향을 구분하지 않고 모두 포괄적으로 수용하여 보안성 체크가 가능하며, 안드로이드 공식 마켓 및 각 통신사 마켓에서 유통되는 모든 종류의 앱과 기업 등에서 자체적으로 사용하는 오피스 앱을 포함하여 보안성 검증을 수행 할 수 있다. 또한 각 검증 방안 보다 더욱 구체적인 항목과 넓은 범위를 확인 할 수 있어 확인 되지 않았던 추가적인 항목에 대한 보안성 체크가 가능하며, 도출된 취약점을 바탕으로 해당 취약사항의 원인 및 발생환경 체크가 가능하다. 점점 발전 하는 모바일 앱의 서비스 환경에 따라 앱을 제공하는 기업 또는 개인의 입장에서는 해당 앱의 고유 서비스뿐만 아니라 안전성, 무결성, 가용성, 신뢰성과 같은 보안성도 제공해야 한다고 생각한다. 본 논문에서 제안한 “안전한 안드로이드 모바일 앱 구현을 위한 개발 단계별 보안검증 방안 제안”을 바탕으로 이러한 문제의 해결방안을 찾을 수 있으며 모바일 앱의 잠재적인 위협요인을 사전에 예방 할 수 있을 것이다.

참 고 문 헌

[1] ArXan Technologies Inc, “State of Security in the App Economy : Mobile Apps Under Attack”, ArXan Technologies Research Report. Vol.1, pp.2-15, 2012.

[2] TrendLabs, “2012 Mobile Threat and Security Roundup Repeating History”, TrendMicro Inc, pp.2-12, 2012.

[3] MOSPA, “Mobile e-Government service App Source code Security Verification Guide(V2.0)”, 2012.

[4] MOSPA, “Security Weakness verification Guide of Software ”, 2012.

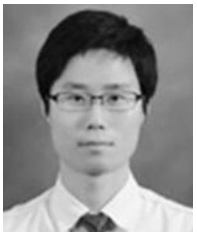
[5] MOSPA, “Security Verification Guide of Software Develop -ment”, 2012.

[6] Shanai Ardi and Nahid Shahmehri, “Introducing Vulnerability Awareness to Common Criteria’s Security Targets”, IEEE Computer society, In Proceedings of the Fourth International Conference on Software Engineering Advances(ICSEA), pp. 421-423, 2009.

[7] William Enck, Machigar Ongtang, and Patrick McDaniel - “Understanding Android Security”, IEEE Security and Privacy 7(1), pp.50-57, 2009.

[8] Jesse Burns, “Developing Secure Mobile Applications for Android”, ISEC PARTNERS, pp.4-26, 2008.

[9] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri, “A Study of Android Application Security”, USENIX Security Symposium, pp.2-28, 2011.



허 환 석

e-mail : eits1st@korea.ac.kr

2009년 영남대학교 컴퓨터공학과(학사)

2011년~현 재 고려대학교 정보보호 대학원 석사과정

2013년~현 재 현대카드 정보보안기술 모의해킹, 취약점분석 담당

관심분야 : Mobile Security, Network Security, Hacking & Security Research

강 성 훈

e-mail : korhoon@korea.ac.kr
2010년 서원대학교 컴퓨터공학과(학사)
2013년 고려대학교 정보보호대학원(석사)
2013년~현 재 고려대학교 정보보호 대학원 박사과정
관심분야 : 정보보증, 정보보호제품 보안성 평가, 보안공학 Usable Security

김 승 주

e-mail : skim71@korea.ac.kr
1994년 성균관대학교 정보공학과(학사)
1996년 성균관대학교 정보공학과(석사)
1999년 성균관대학교 정보공학과(박사)
1998년 12월~2004년 2월 KISA(舊한국정보보호진흥원) 팀장
2004년 3월~2011년 2월 성균관대학교 정보통신공학부 조교수, 부교수
2011년 3월~현 재 고려대학교 사이버국방학과 정보보호대학원 정교수
2012년 3월~2012년 6월 선관위 디도스 특별검사팀 자문위원
관심분야 : 보안공학, 암호이론, 정보보증, 정보보호제품 보안성 평가, Usable Security