기능성과 보안성을 고려한 모바일 앱 개발 방법론 제시

송태한*, 오준석**, 최진영**
*고려대학교 컴퓨터정보통신공학과
**고려대학교 컴퓨터전파통신공학과
e-mail:crezo@naver.com
{jsoh, choi}@formal.korea.ac.kr

Mobile Apps Development Methodology in Considering the Functionality and Security

Tae-Han Song*
*Dept of Computer and Communication Engineering, Korea University
Joon-Seok Oh** Jin-Young Choi**
**Dept of Computer and Radio Communications Engineering, Korea University

요 약

스마트폰의 확산이 증대되고 있으며, 누구든지 모바일 앱을 개발할 수 있는 환경이 제공되고 있다. 모바일 앱을 개발하는 개인 개발자들은 접근성과 이해성이 좋고, 보안성이 고려된 표준 가이드를 적용하여 앱을 개발해야 한다. 이는 오류 발생률 감소 및 코드 수정에 소비되는 시간이 절약될 수 있는 모바일 앱을 개발하는데 도움이 된다. 하지만 개인개발자들을 위한 적절한 가이드가 존재하지 않는 실정이다. 본 논문에서는 개인 개발자들이 쉽게 접근 할 수 있으면서 보안성이 고려된 표준 가이드를 제시하기 위한 개발 방법론을 제시한다.

1. 서론

소프트웨어 개발 시 소프트웨어 특성에 맞는 개발 방법론을 적용하여 빠른 시간에 품질 높은 소프트웨어를 개발해야 한다.

그러나 현재 모바일 앱을 개발하는 개인 개발자들은 개발 전문가, 초급자, 비전문가 등 다양한 수준의 개발자들이 존재하며, 소규모 모바일 앱이라는 특성이 있다. 이런 특성에 맞는 개발 방법론을 개인 개발자가 선택하고, 모바일 앱 개발에 적용하기 어렵다.

이런 환경에서 모바일 앱을 개발하면 기능성과 보안성이 떨어지며, 배포 후 예상치 못한 문제점들이 발생하게 된다.

이런 문제점으로는 첫째, 구현한 모바일 앱의 오류 발생률이 높아 잦은 소스코드 수정이 발생하게 된다. 둘째, 기능 추가 시 기존 소스코드의 재사용이 어려워 주먹구구식으로 개발하게 된다. 셋째, 개인 개발자는 모바일 앱의 특성 및 기능과 상관없는 불필요한 개인 정보를 수집하여 개인정보를 악용, 침해하는 문제가 발생하게 된다.

본 논문에서는 모바일 앱을 개발하는 설계단계에서 모바일 앱의 기능성을 위해 OMG의 표준 방법론인 UML(Unified Modeling Language)을 이용하고 구현단계에서 보안성을 위한 Secure Coding 가이드 및 Code Inspection을 이용하여 개인 개발자들이 쉽게 접근 할 수 있는 개발 방법론을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 마이크로소프트(MS : Microsoft) 회사에서 개발한 MS-SDL(Secure Software Development Life Cycle)와 CWE(Common Weakness Enumeration) 및 CERT(Computer Emergency Response Team)의 시큐어 코딩 가이드에 대해 살펴보고, 3장에서는 개인 개발자들에게 적합한 개발 방법론을 제시하고, 4장에서는 결론 및 향후 연구 방향을 제시하여 본 논문을 마무리한다.

2. 관련 연구

2.1 MS-SDL 방법론

마이크로소프트는 오래 전부터 소프트웨어 설계와 테스트 과정에서 보안을 주요 이슈로 포함시켜 놓았다.

마이크로소프트는 시큐어 소프트웨어를 개발하기 위해 기본 원리로 “Secure by Design, Secure by Default, Secure in Deployment, and Communications(SD3+C)”를 정의했다[2].

○ Secure by Design : 아키텍처・설계・구조에서 보안 고려, 설계와 기능 명세시 위협 모델링 생성 및 완화 방법 적용, 알려진 취약점 제거, 표준에 적합한 코딩으로 보안성 강화

○ Secure by Default : 최소 권한으로 모든 컴포넌트 실행, 보수적인 기본 설정, 위험을 야기하는 기본 설정 변

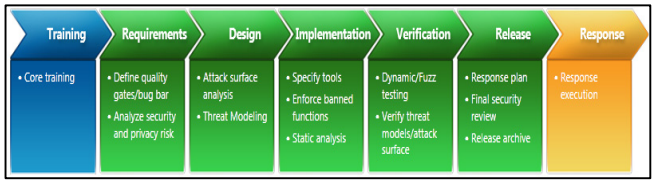
경 회피, 사용하지 않는 기능은 기본적으로 서비스 중지

○ Secure in Deployment : 배포가이드 규정, 분석과 관리 툴 사용, 패치 배포 툴 제공

소프트웨어 개발 생명주기 동안 안전한 소프트웨어를 개발하기 위해 소프트웨어 개발 생명주기에 보안과 관련된 실무를 결합하여 SDL방법론을 만들기 시작했고 2002년 보안 푸시(Secure Push)를 거쳐 SDL방법론이 만들어졌다.[1][2]

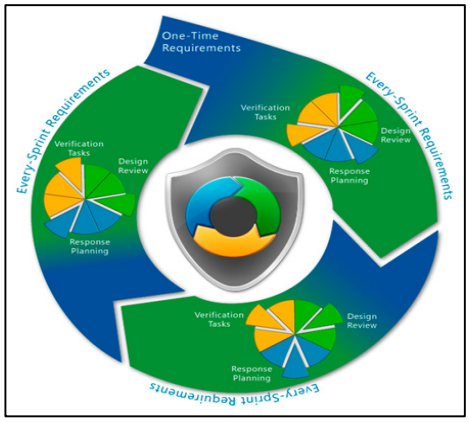
MS-SDL은 교육, 요구사항, 설계, 구현, 검증, 릴리즈, 대응 단계로 구성되며 각 단계별로 필수 보안활동과 권고 보안활동 두 단계로 구분한다.

2010년 4월에 SDL 5.0이 릴리즈 되었고 기존 모델인 그림 1에서 요구사항, 설계, 구현, 검증 단계에 필수 보안 활동들이 추가되어 보안성이 강화되었다[2].



(그림 1) MS SDL 프로세스

이에 더해서 SDL과 애자일 방법론의 원칙을 유지하면서 SDL과 애자일 방법론이 융합될 수 있는 그림 2의 SDL-Agile 프로세스 모델을 추가 하였다[3].



(그림 2) MS SDL 프로세스

2.2 CWE

미 국토안보부(The U.S. Department of Homeland Security)에서 관리하고 있는 CWE는 2009년에 "TOP 25 Most Dangerous Programming Errors" 프로젝트를 시작하여 현재까지 지속적으로 연구하고 있다[4].

이 프로젝트는 소프트웨어 산업에서 가장 빈번하게 발생하고 위험한 25개의 오류 목록을 정리하였다. 소프트웨어의 취약점을 뷰, 카테고리, 취약점, 복합요소를 기준으로 분류하여 개발자가 쉽게 접근하여 이용할 수 있도록 구성되어 있다.

2.3 CERT

카네기 멜론 대학교의 소프트웨어 공학연구소에서 관리하는 CERT는 시큐어 코딩에서도 특히 코딩 규칙 및 가이드의 표준화 작업에 활발한 활동을 하고 있다.

CERT는 프로그래밍 언어 별 특징을 기준으로 시큐어 코딩 표준을 분류하여 언어 별 사용자 및 학습자가 용이하게 접근하도록 하고 있다.

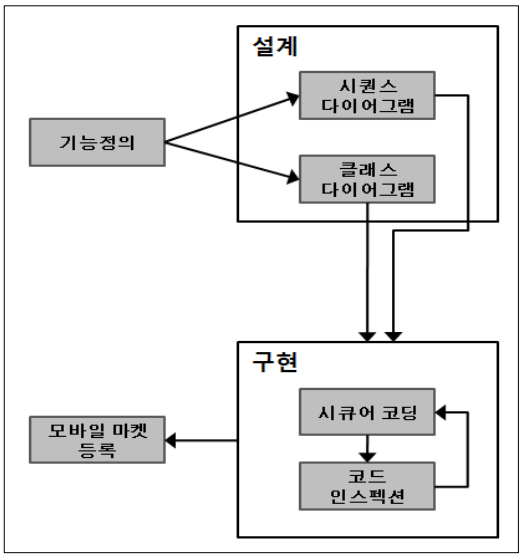
현재는 C, Java, C++, TIOBE Sofrware 언어에 대한 코딩 규칙을 생성하며 개발자를 위한 다양한 가이드를 제시하고 있다[5].

3. 기능성과 보안성을 고려한 모바일 앱 개발 방법론 제시

본 장에서는 UML, 시큐어 코딩, 코드 인스팩션을 이용하여 기능성과 보안성을 고려한 모바일 앱 개발 방법론을 제시한다. 또한 방법론의 절차를 간소화하여 개인 개발자들이 쉽게 접근 할 수 표준 가이드를 제시한다.

본 논문에서 제시하는 개발 방법론은 모바일 앱 개인 개발자들 중 초급자 및 비전무가들을 위한 표준 가이드로 전제한다.

개발 방법론은 그림 3과 같이 기능정의, 설계, 구현, 그리고 모바일 마켓 등록으로 총 4단계로 나눈다. 올바른 기능성을 위해 설계 단계에서 UML을 사용하며, 모바일 앱의 보안성을 위해 구현 단계에서는 시큐어 코딩 가이드를 준수하고 코드 인스펙션 단계를 거쳐야 한다.



(그림 3) 모바일 앱 개발 방법론 제시

3.1 설계 단계

본 논문에서 제시하는 설계 단계는 초급자와 비전문가

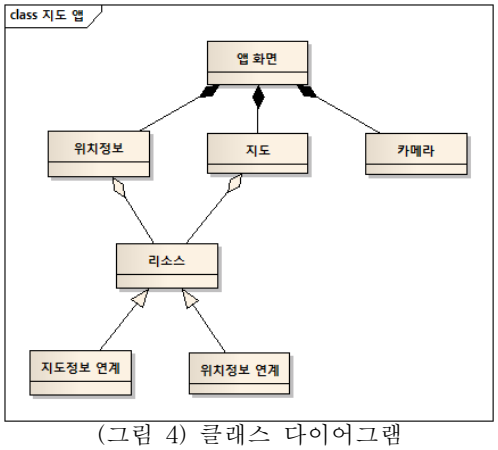
의 개인 개발자의 수준에 맞춰 애플리케이션의 구조와 흐름을 파악할 수 있도록 가이드한다. 기능정의 단계에서 도출된 기능을 바탕으로 UML 기반의 클래스 다이어그램과 시퀀스 다이어그램을 작성하면서 분석 겸 설계 활동을 수행한다.

설계 단계를 거치지 않고 바로 개발자의 생각대로 구현에 들어가면 기능흐름이 명확한 상태가 아니기 때문에 잦은 변경이 발생하며 소스코드의 구조가 복잡해져 애기치 못한 오류가 발생할 수 있다. 또한 기능 버전업을 위해 소스 재사용시 설계 자료가 없어 기존 소스를 재분석하고 이해를 해야하는 시간 낭비가 발생한다.

이러한 상황을 방지하고 애플리케이션 품질 향상을 위해 반드시 설계 단계의 클래스 다이어그램과 시퀀스 다이어그램을 작성하여 애플리케이션의 이해도를 높이고 구현 단계로 진행해야 한다.

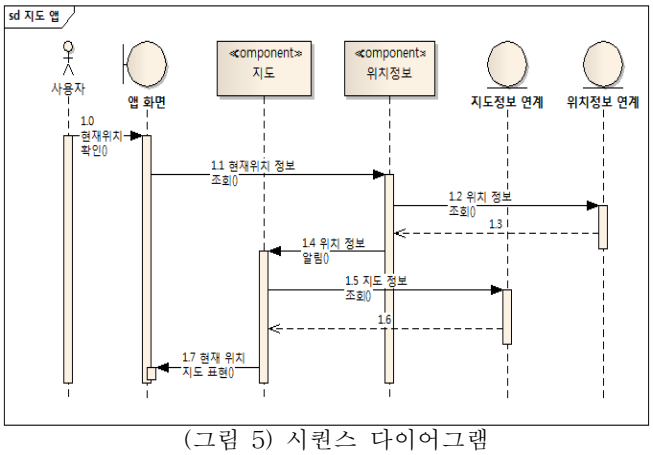
본 논문의 설계 단계에서 UML을 적용한 이유는 다음과 같다. 첫째, UML은 애플리케이션을 개발할 때 애플리케이션을 쉽게 이해할 수 있도록 도와주는 여러 가지 유형의 다이어그램을 제공한다[6]. 이 중에서 클래스 다이어그램과 시퀀스 다이어그램은 애플리케이션의 흐름과 클래스의 구조를 쉽게 파악하는데 사용된다. 둘째, 구현하고자 하는 앱의 전체상을 다양한 관점에서 UML을 통해 시각화하여 기능 전체를 파악하고 설계를 명확히하여 개발자가 미처 파악하지 못한 문제점을 발견할 수 있다. 셋째, 향후 앱의 버전 업이나 다른 개발언어로 재개발할 경우 정형화·표준화된 UML을 사용한 기존 설계를 재사용 할 수 있다. 넷째, 설계나 구현 시 다른 사람들에게 공유나 조언을 구할 때 표준화된 UML을 사용한 설계를 통하여 의견 교환이나 공통적인 이해를 돕는데 효율적이다.

설계시 클래스 다이어그램은 그림 4와 같이 개발자가 육안으로 알아보기 쉽게 작성한다. 이를 통해 애플리케이션 소스의 뼈대와 기틀을 마련하고 복잡한 소스 구조를 미연에 방지하는 효과를 준다.



(그림 4) 클래스 다이어그램

클래스 다이어그램을 작성 후 클래스 다이어그램에서 도출된 클래스들을 사용하여 애플리케이션 기능 흐름파악을 위해 그림 5와 같이 시퀀스 다이어그램을 작성한다.



(그림 5) 시퀀스 다이어그램

개발자는 시퀀스 다이어그램을 통해 기능 흐름상 오류가 발생하는 부분을 발견하거나 잘못된 기능 흐름을 파악하여 애플리케이션의 오류를 방지하는 효과를 준다.

3.2 구현 단계

본 논문의 구현 단계에서는 개발 단계부터 소스코드의 보안 취약점을 고려해 코딩하도록 CWE와 CERT에서 제공하는 시큐어코딩 가이드를 참고하여 개발해야 한다.

구현단 계에서 초급자와 비전문가의 개인 개발자가 시큐어코딩을 적용하지 않고 개발하는 경우 다음과 같은 문제들을 야기한다. 첫째, 모바일 앱은 인터넷에 접근이 가능하고 스마트폰에 존재하는 정보(예를 들면 통화내역, 연락처, 위치정보, 이메일 정보, 전자결재, 기밀정보 등 개인 및 업무관련 중요 정보 등)에 접근이 가능하여 소스코드의 취약점이 존재하는 경우 공격자가 이를 악용하여 개인의 중요정보를 유출하는 문제가 있다. 둘째, 개인 개발자는 앱 구현 시 폰의 정보를 접근하는 권한에 대해서는 중요치 않게 생각하여 실제 앱의 기능과 상관없는 정보에 접근하여 인터넷에 불필요한 개인 정보를 유출하는 문제가 있다. 셋째, 개인 개발자들의 코딩 스타일이 다양하기 때문에 소스코드에 취약점이 있는 앱을 배포할 경우 앱스토어에서 취약점을 발견하기에는 한계가 있기 때문에 개발자스스로 취약점이 없는 소스코드를 작성해야한다.

이러한 문제점들을 구현 후에 보완하는 것이 아닌 구현 단계에서 예방하도록 시큐어코딩을 준수하여 보안침해 사고를 막고 시큐어한 코드를 개발하도록 가이드한다.

개인 개발자는 개발언어의 특징에 따라 시큐어 코딩가이드를 적용해야 하며 CERT에서 제공하는 시큐어 코딩

가이드를 참고해야 한다. 아이폰 앱을 개발할 경우는 "the CERT C Secure Coding Standard, Version 2.0"[5], 안드로이드폰 앱을 개발할 경우는 "the CERT Oracle Secure Coding Standard for Java"[5] 가이드를 참고하여 코딩을 한다.

CERT의 시큐어 코딩가이드를 참고하여 코딩하면서 CWE에서 제공하는 표 1의 가장 빈번하게 발생하고 위험한 25개의 오류 목록들 중 모바일 앱 기능과 관련있는 부분을 참고하여 코딩의 보안성 및 애플리케이션의 품질을 향상시켜야 한다.

RANK	ID	Name
[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[2]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	CWE-352	Cross-Site Request Forgery (CSRF)
[5]	CWE-285	Improper Access Control (Authorization)
중간생략		
[20]	CWE-494	Download of Code Without Integrity Check
[21]	CWE-732	Incorrect Permission Assignment for Critical Resource
[22]	CWE-770	Allocation of Resources Without Limits or Throttling
[23]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[24]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[25]	CWE-362	Race Condition

(표 1) CWE/SANS 2010년 25개의 가장 위험한 소프트웨어 에러

개인 개발자들은 시큐어 코딩 가이드를 준수하고 소프트웨어 취약점인 주요 오류를 예방하여 코딩하는 것이 쉽지 않다. 하지만 시큐어한 코딩 습관을 익힘으로서 소프트웨어의 보안성과 품질을 향상하여 사용자에게 발생하는 보안침해 사고를 줄일 수 있고 개인 개발자 스스로의 코딩 실력도 향상 된다.

소스 코드를 작성 후에는 코드 인스펙션 툴을 통해 예상치 못한 소스코드의 취약점을 분석하고 수정하여 소프트웨어의 보안과 품질을 향상한다. 코드 인스펙션[7]의 주요 기능으로는 표 2의 내용들이다. 주로 문법 오류 검사, 소스 코드 실행시 발생 가능한 오류 검사, 사용되지 않는 소스 코드 검사 등이다.

주요기능	설명
Syntax Error Inspection	작성한 소스 코드의 Syntax 오류를 검사하는 기능을 제공한다.
Logical Error Inspection	작성한 소스 코드에서 실행 시 발생 가능한 오류를 찾아내는 기능을 제공한다.
Reference Inspection	작성한 소스 코드에서 실행 시 사용되지 않는 부분을 찾아내는 기능을 제공한다.
Reporting	인스펙션을 수행한 결과를 Excel, HTML등이 문서 형식으로 제공하는 기능을 제공한다.
Rule Customizing	Rule을 정의하는 API를 사용하여 새로운 Rule을 정의하여 추가할 수 있는 기능을 제공한다.

(표 2) 코드 인스펙션 주요기능

4. 결론 및 향후 연구

본 논문에서 제안한 방법론은 설계단계에서 UML을 사용하여 애플리케이션의 기능을 명확히 하고, 구현단계에서 시큐어 코딩과 코드 인스펙션을 통해 소스코드의 취약점을 보완하고 품질 및 보안을 향상 시킬 수 있는 특징이 있다. 또한 개발 절차를 간소화 하여 개인 개발자들이 방법론을 쉽게 접하고 적용 할 수 있는 장점이 있다.

향후 연구로는 본 논문에서 제시한 방법론을 실제 모바일 앱 개발에 적용하여 취약성으로부터 안전한 모바일 앱을 개발하고자 한다. 추가적으로 설계 단계에서 나온 산출물을 통해 코딩을 효과적으로 할 수 있는 방안을 연구하고자 한다.

참고문헌

[1] The Trustworthy Computing Security Development Lifecycle, "http://msdn.microsoft.com/en-us/library/ms-995349.asp", 2010

[2] Microsoft Security Development Lifecycle (SDL) - version 5.0, "http://msdn.microsoft.com/en-us/library/c-c307748.aspx", 2010

[3] SDL for Agile Development, "http://msdn.microsoft-.com/en-us/library/ee790621.aspx", 2010

[4] CWE, "2010 CWE/SANS Top 25 Most Dangerous Software Errors", "http://cwe.mitre.org/top25/", June 29, 2010

[5] CERT, Secure Coding, "http://www.cert.org/secur-e-coding/"

[6] Donald Bell, "UML의 기초: Unified Modeling Lang-uage 소개", "http://www.ibm.com/developerworks/kr/li-brary/769.html", 2003.11.25

[7] Code Inspection, 전자정부표준프레임워크, "http://w-ww.egovframe.org/wiki/doku.php?id=egovframework:dev:i mp:inspection"