情報保護學會論文誌
第 22 卷 第 4 號, 2012. 8

모바일 시큐어코딩 자가평가(M-SCSA) 방법에 대한 연구

김 동 원,[1†] 한 근 희[2‡]
[1]고려대학교 정보보호대학원, [2]건국대학교 정보통신대학원

A Study on Self Assessment of Mobile Secure Coding

Dong-Won Kim,[1†] Keun-Hee Han[2‡]
[1]Graduate School of Information Security, Korea University, [2]Konkuk University

요 약

개발단계에서의 보안취약점 제거는 운영단계에서 실행하는 것보다 훨씬 더 효율적이고 효과적으로 적용될 수 있다. 소프트웨어에 내재된 보안취약점이 사이버 침해사고의 주요 원인이 되고 있어서 소스코드 수준에서의 보안취약점을 최소화하기 위한 일환으로 시큐어코딩이 주목받고 있다. 소프트웨어 개발과정에서 보안취약점을 제거하는 것이 보다 효과적이면서도 근본적인 해결책이 될 수 있다. 본 논문에서는 개인・단체・조직에서 모바일 시큐어코딩 적용에 따른 보안수준을 평가하기 위한 모바일 시큐어코딩 자가평가(Mobile-Secure Coding Self Assessment) 방법을 연구 제안한다.

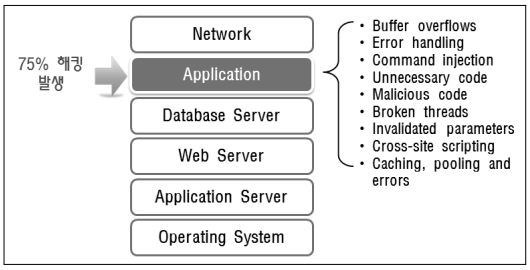
ABSTRACT

The removal of security vulnerabilities during the developmental stage is found to be much more effective and much more efficient than performing the application during the operational phase. The underlying security vulnerabilities in software have become the major cause of cyber security incidents. Thus, secure coding is drawing much attention for one of its abilities includes minimizing security vulnerabilities at the source code level. Removal of security vulnerabilities at the software's developmental stage is not only effective but can also be regarded as a fundamental solution. This thesis is a research about the methods of Mobile-Secure Coding Self Assessment in order to evaluate the security levels in accordance to the application of mobile secure coding of every individual, groups, and organizations.

Keywords: Secure Coding, Mobile Secure Coding Self Assessment(M-SCSA)

I. 서 론

최근의 사이버 공격은 보안패치가 발표되기 이전의 보안취약점을 악용하는 Zero Day 공격, 웹사이트 대상의 해킹 등이 주를 이루고 있으며, 이러한 공격은 소프트웨어 자체의 보안취약점을 이용하는 경우가 대부분이다. 〔그림1〕과 같이 최근의 가트너社 발표에 의하면, 사이버 공격의 약 75%가 응용 프로그램의 취약점을 악용한 것이라고 한다.〔1〕



〔그림 1〕 응용프로그램의 보안 수준(Gartner)

접수일(2012년 2월 28일), 수정일(2012년 6월 26일), 게재확정일(2012년 8월 6일)
† 주저자, blast.kim@hyundai-autoever.com
‡ 교신저자, hankeunhee@nate.com

"소프트웨어 개발보안"이란 소프트웨어 개발과정에서 개발자의 실수나 프로그래밍 오류 등으로 인해 소프트웨어에 내재된 보안취약점을 최소화하는 한편, 해킹 등 보안위협에 대응할 수 있는 안전한 소프트웨어를 개발하기 위한 일련의 과정을 의미한다. 넓은 의미에서 소프트웨어 개발보안은 소프트웨어 생명주기(SDLC, Software Development LifeCycle)의 각 단계별로 요구되는 보안활동을 모두 포함하며, 좁은 의미로는 소프트웨어 개발과정에서 소스코드를 작성하는 구현단계에서의 보안취약점을 배제하기 위한 "시큐어코딩(Secure Coding)"을 의미한다. 소프트웨어에 내재된 보안취약점은 사이버 침해사고의 주요 원인이 되고 있다. 한편, 이러한 유형의 공격 차단에 한계가 있으므로, 소프트웨어 개발과정에서 보안취약점을 배제하는 것이 보다 효과적이면서도 근본적인 해결책이 될 수 있다.[1]

스마트폰의 확산으로 인해 다양한 종류의 스마트폰 앱이 개발되고 있다. 이러한 앱들은 기존의 PC에서 수행하던 간단한 작업들을 대신할 수 있으나 앱이 대형화되고 복잡해짐에 따라, 앱 개발과정에서 내재된 소프트웨어 보안취약점의 비율이 높아지고 있다. 이러한 취약점은 컴파일러에 의해서는 탐지되지 않고, 개발자가 놓치기 쉽기 때문에 소프트웨어에 오랫동안 존재한다.[2] 이러한 스마트폰 앱에 내재된 보안취약점은 외부 공격에 이용되어 스마트폰에 저장되어 있는 정보유출이나 사용자가 원하지 않는 통신요금을 유발시키는 등 다양한 공격에 악용될 수 있다.

또한 최근 정부에서는 2012년 10월부터 추진하는 정보화사업에 시큐어코딩을 의무적으로 적용한다. 시큐어코딩을 개발단계부터 적용하도록 "정보시스템 구축•운영 지침 개정안을 마련하며, 소프트웨어 개발보안 연구센터를 개설하는 등 투자가 활발하게 진행되고 있다.[3]

II. 모바일 시큐어코딩 자가평가(M-SCSA)

모바일 시큐어코딩 자가평가(Mobile-Secure Coding Self Assessment : 이하 M-SCSA)는 개인・단체・기관에서 모바일용 응용프로그램의 시큐어코딩 적용에 따른 보안수준을 스스로 평가하기 위한 도구이다. 본 논문에서 제안하는 M-SCSA 방법은 "안드로이드 앱 시큐어코딩 표준"[4]을 목록화하여 스마트폰 보안위협의 정성적 분류에 따라 시큐어코딩 분류체계를 도출한 후, "심각도", "영향도", "발생가능성"의 산정기준을 정의하여 새로운 시큐어코딩 기준표를 도출한다. 도출된 기준표와 M-SCSA를 매핑하기 위한 식별번호를 정의한 후 3단계 등급(필수, 선택, 권고)으로 분류하여 M-SCSA의 평가표인 M-SCSA Matrix를 정의한다. 이를 이용해 개인・단체・기관에서 스스로 시큐어코딩 적용 여부에 따른 보안수준을 자가평가할 수 있을 것이다.

2.1 M-SCSA 분류체계

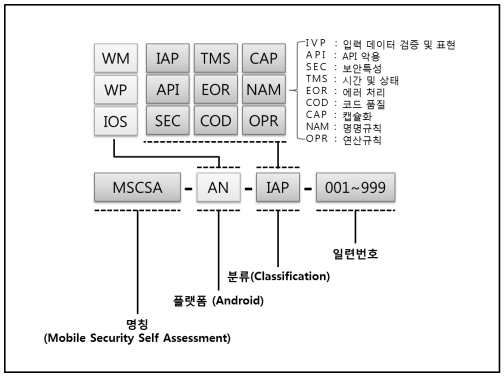
시큐어코딩의 취약점에 따라 분류하여 공통되는 항목 단위로 분류할 것이다. 분류체계는 기존의 시큐어코딩을 목록화한 후 공통되는 분류항목을 도출한다. 제시된 보안취약점 유형은 CWE[5] 분류방법론 중의

[표 1] 시큐어코딩 분류표

No	분류항목	설명
1	입력 데이터 검증 및 표현	입력을 검증 없이 그대로 받아들여 사용하면 많은 보안위험에 노출된다.[6]
2	API 악용	API 오용 및 취약점이 알려진 API의 사용은 개발효율성, 유지보수성의 저하 및 보안상의 심각한 위협요인이 될 수 있다.[6]
3	보안 특성	부적절한 보안특성의 사용은 오히려 성능이나 부가적인 문제를 불러올 수도 있다.[6]
4	시간 및 상태	프로그램의 동작 과정에서 시간적 개념을 포함한 시스템 상태에 대항 정보에 관련한 취약점을 유발할 수 있다.[6]
5	에러 처리	에러를 불충하게 처리하지 않을 때나 에러 정보에 과도하게 많은 정보를 포함할 경우 공격자에게 악용될 수 있다.[6]
6	코드 품질	프로그램 코드가 너무 복잡하면 관리성, 유지보수성, 가독성이 떨어지며, 안전성을 위협할 취약점들이 코드 안에 숨겨져 있을 가능성이 높다.[6]
7	캡슐화	중요한 데이터나 기능성을 불충분하게 캡슐화하는 경우, 인가된 데이터와 인가되지 않은 데이터를 구분하지 못하여 허용되지 않는 사용자들 간의 데이터 유출이 가능해진다.[6]
8	명명 규칙	좋은 명명규칙은 프로그램의 가독성이 있어서 핵심적인 요소이다. 루프 인덱스와 상태변수 같은 특정한 종류의 변수들은 특정한 고려사항이 요구된다.[7]
9	연산 규칙	잘못된 연산을 수행하면 오버플로를 유발할 수 있고, 경우에 따라 공격자가 임의 코드를 수행시킬 수 있게 허용하는 결과를 초래한다.[8]

하나인 '7 Pernicious Kingdom'〔6〕 분류체계를 준용한 것으로 개발자 및 보안전문가가 소프트웨어 보안 취약점을 소스코드 관점에서 이해하기 쉽도록 유형을 분류하였다.

〔그림 2〕는 시큐어코딩 분류체계를 용이하게 식별하고, 모든 기준표 및 기준자료와 매핑하기 위한 수단으로 사용하기 위한 식별번호 체계이다.



〔그림 2〕 식별번호 체계

2.2 M-SCSA 위험평가

위험평가 기준은 심각도, 영향도, 발생가능성 3개의 평가요소로 구성된다. 심각도(Severity)는 시큐어코딩 규칙을 무시했을 때의 결과가 얼마나 심각한지를 평가하기 위한 요소이다. 영향도(Impact)는 시큐어코딩 규칙을 무시했을 때의 결과가 프로그램에 얼마나 큰 영향을 끼치는지 평가하기 위한 요소이다. 발생가능성(Possibility)은 시큐어코딩 규칙을 무시해 악용당할 가능성을 평가하기 위함이다. 각 평가요소에는 1에서 3 사이의 값이 부여된다.

위험평가를 구성하는 세 가지 평가요소에 대한 판단 기준과 평가 점수는 주관적일 수밖에 없다. 동일한 위험에 대해서도 개인이 느끼는 바와 기관이 느끼는 바가 다르고, 기관의 규모에 따라 다르며, 또 정보자산의 규모에 따라 달라질 수 있기 때문이다. 이를 객관화하기 위하여 연간 기대 손실액(ALE : Annualized Loss Expectancy)을 계산하는 방법〔10〕 등을 사용할 수 있으며, 위험평가 평가요소의 근거를 보완하기 위해서 보안전문가들의 주관적 평가점수를 합산하고 평균을 내어 기준을 평가하는 방식인 델파이 법으로 알려진 방식을 적용할 수도 있다.〔11〕

위험평가의 세 가지 평가요소 중에 심각도에 대한

〔표 2〕 위험평가 요소

평가요소	범위		설명
심각도 (Severity)	1	낮음	규칙을 무시했을 때의 결과가 얼마나 심각한가?
	2	보통	
	3	높음	
영향도 (Impact)	1	낮음	규칙을 무시했을 때의 결과가 프로그램에 얼마나 큰 영향을 끼치는가?
	2	보통	
	3	높음	
발생가능성 (Possibility)	1	낮음	규칙을 무시해 해킹에 악용당할 가능성은 어떤가?
	2	보통	
	3	높음	

판단기준은 다음 표와 같다.〔11〕

〔표 3〕 심각도에 대한 판단기준

구분	1점	2점	3점
위험 수준	낮음	보통	높음
판단 기준	초래된 위험을 어플리케이션 자체에서 즉시 극복할 수 있음	초래된 위험을 어플리케이션 자체에서 상당한 노력을 기울여야 극복할 수 있음	초래된 위험을 어플리케이션 자체에서 극복할 수 없음
위험사례	해당 어플리케이션의 오류로 보안수준(정보유출, 조작 등)에 영향이 있으며, 오류수정이 1일 이하 소요	해당 어플리케이션의 오류로 보안수준(정보유출, 조작 등)에 영향이 있으며, 오류수정이 1일 이상 소요	해당 어플리케이션 오류로 보안수준(정보유출, 조작 등)에 영향이 있으며, 단말기의 작동불능을 초래하며 오류 수정이 3일 이상 소요

위험평가 평가요소의 세 가지 평가요소 중에 영향도에 대한 판단기준은 다음 표와 같다.〔11〕

〔표 4〕 영향도에 대한 판단기준

구분	1점	2점	3점
위험 수준	낮음	보통	높음
판단 기준	어플리케이션 자체의 보안성을 위협하는 위험수준	단말기 간의 어플리케이션 및 데이터의 보안성을 위협하는 위험수준	네트워크 또는 통합시스템 단위의 보안성을 위협하는 위험수준
위험사례	해당 어플리케이션이 취급하는 정보의 유출, 조작 위험 초래 등	단말기에 담긴 정보의 유출, 조작 위험 초래, 단말기 사용성 방해 등	네트워크 통신 정보의 유출, 조작 위험 , 통합시스템 작동불능 초래, 사회적 문제 야기 등

위험평가 평가요소의 세 가지 평가요소 중에 발생가능성에 대한 판단기준은 다음 표와 같다.〔11〕

〔표 5〕 발생가능성에 대한 판단기준

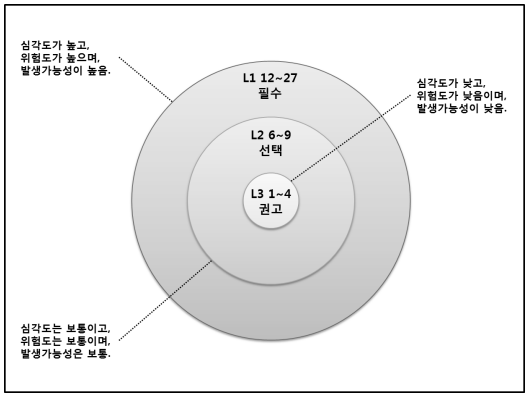
구분	1점	2점	3점
위험수준	낮음	보통	높음
판단기준	어플리케이션의 생명주기 동안 거의 발생하지 않음	어플리케이션의생명주기 동안 수차례 발생함	어플리케이션의생명주기 동안 매우 자주 발생함.
위험사례	명명규칙 및 표현의 부적절한 사용 등으로 인한 오류로 발생빈도가 적음	데이터 조작 및 연산 규칙의 부적절한 사용 등으로 인한 오류로 발생빈도가 보통	입력 검증 및 보안 특성의 부적절한 사용 등으로 인한 오류로 발생빈도가 높음

위험평가는 심각도, 영향도, 발생가능성의 세 가지 요소를 이용한다. 평가요소는 FMECA(Failure Mode, Effects, and Criticality Analysis)에 기초한 위험평가 방법을 준용한다.〔12〕 위험평가에 따른 등급산정 방법은 세 가지 평가요소의 값을 곱하여 결정하며, 시큐어코딩의 등급은 필수항목, 선택항목, 권고항목으로 분류한다. 평가요소 값의 범위는 1~27이며, 평가요소의 값이 1~4의 값이면 권고항목(Level3), 6~9는 선택항목(Level2), 12~27이면 필수항목(Level1)으로 등급화 한다.

〔표 6〕 위험평가 등급 산정기준

위험평가	내 용
필수 (12~27) 12 〈= (S*I*P) 〉= 27	• 시큐어코딩 필수 적용 - 어플리케이션에 보안 취약점이 다수 존재하며, 심각도 및 위험의 정도가 매우 크며, 발생가능성이 높은 수준
선택 (6~9) 6 〈= (S*I*P) 〉= 9	• 시큐어코딩 선택 적용 - 어플리케이션에 보안 취약점이 다수 존재할 수 있으나, 심각도 및 위험의 정도가 보통이며 발생 가능성도 보통 수준
권고 (1~4) 1 〈= (S*I*P) 〉= 4	• 시큐어코딩 적용 권고 - 어플리케이션에 보안 취약점이 다수 존재할 수 있으나, 심각도 및 위험의 정도가 수용 가능하며 발생가능성이 낮은 수준

위험을 평가하기 위한 산정기준과 등급을 도식화하면 다음과 같다.



〔그림 3〕 위험평가 평가도

M-SCSA 위험평가 기준은 시큐어코딩 기준표의 위험을 평가하기 위한 기준 자료로 활용된다. 위험평가를 통해 시큐어코딩 기준표를 등급화하여 필수, 선택, 권고 항목으로 도출한다. M-SCSA 위험평가 예는 〔표7〕과 같다.

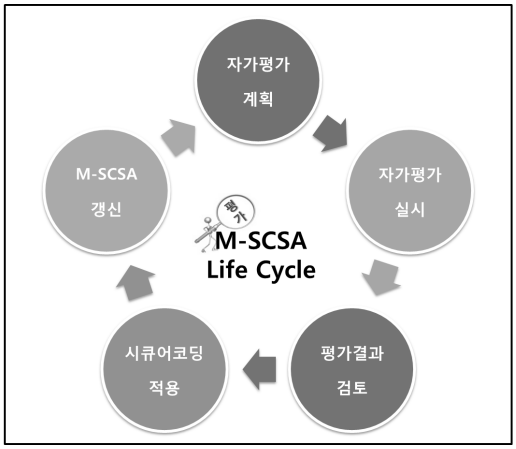
〔표 7〕 입력데이터 검증 및 표현의 위험평가

No	시큐어코딩 식별명
	심각도(S) * 영향(I) * 발생가능성(P)
001	상대 디렉터리 정보 조작
	(2) * (3) * (2) = 12
002	절대 디렉터리 정보 조작
	(2) * (3) * (2) = 12
003	메서드 파라메타를 적절히 검사한다.
	(2) * (3) * (2) = 12
004	생성자에서 재 정의될 수 있는 메서드를 호출하지 않는다.
	(1) * (2) * (2) = 4
005	반복문 루프카운터 변수로 실수 값을 사용하지 않는다.
	(1) * (2) * (2) = 4
006	wait()(또는 await())는 루프(loop)문에서 호출한다.
	(1) * (2) * (2) = 4
007	올바르게 정수를 표현한다.
	(1) * (1) * (1) = 1

III. 모바일 시큐어코딩 자가평가(M-SCSA) 모델

모바일 시큐어코딩 자가평가(M-SCSA) 모델은 정보보호관리체계(Information Security Manage-

ment System)의 계획(Plan)-시행(Do)-점검(Check)-개선(Action) 모델을 인용하였다. M-SCSA는 "계획-실시-검토-적용-갱신"의 생명주기로 적용한다. 〔그림4〕는 M-SCSA의 생명주기 모델이다.



〔그림 4〕 M-SCSA 생명주기

3.1 M-SCSA Life Cycle

M-SCSA 생명주기는 시큐어코딩 자가평가 방법의 지속적 개선을 그 목적으로 하고 있다. 본 논문에서 제시하는 "안드로이드 앱 시큐어코딩 기준표"는 시큐어코딩의 극히 일부분에 지나지 않는다. 시큐어코딩 항목은 개발하고자 하는 응용프로그램의 성격, 용도, 규모 등에 따라 다르며, 개인이나 단체, 조직에 따라 다를 수 있다. M-SCSA 생명주기는 이러한 다양성을 최소화하기 위하여 설계하였다.

(1) 자가평가 계획

• 자가평가 계획

- 개인・단체・조직에 따라 적합한 시큐어코딩 자가평가 계획 수립
- 시큐어코딩 자가평가 프로세스 및 절차 수립

(2) 자가평가 실시

• 자가평가 실시

- 계획에 따른 위험평가 실행
- 위험평가 결과를 반영한 시큐어코딩 자가평가 실시

(3) 평가결과 검토

• 평가결과 검토

- 시큐어코딩 평가 측정결과의 검토
- 검토결과에 따른 시정조치

(4) 시큐어코딩 적용

• 시큐어코딩 적용

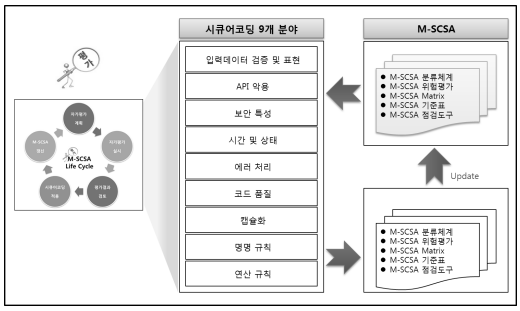
- 개인・단체・조직에 따라 최적화된 시큐어코딩 적용
- 시큐어코딩 적용에 따른 성과 보고

(5) M-SCSA 갱신

• M-SCSA 갱신

- 개인・단체・조직에 따라 적합한 시큐어코딩 규칙 축적
- 일회성이 아닌 지속적 관리를 위한 M-SCSA 갱신

위와 같이 (1)시큐어코딩 계획을 통해 개인・단체・조직에 따라 적합한 시큐어코딩 자가평가 계획을 수립하고, 시큐어코딩 자가평가 프로세스 및 절차를 수립한다. 다음으로 (2)자가평가 실시에서 계획에 따른 효과적・효율적 위험평가를 실행하여 위험평가 결과를 반영한 시큐어코딩 자가평가를 실시한다. (3)평가결과 검토를 통해 시큐어코딩 평가 측정결과를 검토하여 시정 조치한다. 이후 (4)시큐어코딩 적용으로 개인・단체・조직에 최적화된 시큐어코딩을 적용하고 성과를 보고한다. 마지막으로 (5)M-SCSA 갱신은 개인・단체・조직에 따라 적합한 시큐어코딩을 축적함으로써, 일회성이 아닌 지속적 관리를 통해 M-SCSA를 개선한다.



〔그림 5〕 M-SCSA 평가 모델

3.2 M-SCSA 평가 방법

M-SCSA는 시큐어코딩의 적용여부에 대한 안전성을 스스로 평가하기 위해 "심각도", "영향도", "발생가능성"에 따라 시큐어코딩의 적용을 적절히 강구하고 있는지를 평가하는 것이다. 이를 통해 개인・단체・조직이 스스로 모바일 응용프로그램 개발 시 정보보안수준을 진단할 수 있는 능력을 배양하고, 보안대책이 미흡한 부분이나 정보보안의 사각지대를 해소함으로써 보다 안전한 모바일 응용프로그램을 개발할 수 있을 것이다. 또한 개발단계에서 발생할 수 있는 소프트웨어 허점 및 보안취약성을 사전에 제거하고, 코드의 모호한 부분을 제거하여 보안성이 강화된 코드를 개발하는 데 도움이 될 것이다.

M-SCSA 평가 방법으로 시큐어코딩 위험평가에 따라 평가요소별 중요도를 산정하여 우선순위를 부여한다. 〔표 8〕은 M-SCSA 평가기준에 따른 항목별 중요도 기준표이다.

〔표 8〕 M-SCSA 평가 중요도 기준표

평가요소(E)	중요도(I)		
	높음	보통	낮음
입력데이터 검증 및 표현	81~100%	51~80%	1~50%
API 악용	81~100%	51~80%	1~50%
보안 특성	81~100%	51~80%	1~50%
시간 및 상태	81~100%	51~80%	1~50%
에러 처리	81~100%	51~80%	1~50%
코드 품질	81~100%	51~80%	1~50%
캡슐화	81~100%	51~80%	1~50%
명명 규칙	81~100%	51~80%	1~50%
연산 규칙	81~100%	51~80%	1~50%

위에서 제시한 M-SCSA 평가 중요도 기준표에 따라 평가요소별 중요도를(I), 각 평가요소(E_i), 위험평가 최대점수 27을(R_{MAX}), 위험평가 항목 개수를(n)이라고 하면 총점에 따른 산정식은 다음과 같다.

$$I = \frac{\sum_{i=1}^{n} Ei}{R_{MAX} \times n} \times 100$$

중요도 산정을 위하여 평가요소(E)별 평균값을 계산한다. 산정식에 따라 산정된 결과 중요도(I)가 높을수록 평가요소에 포함된 시큐어코딩 "필수" 항목이 많은 것을 의미한다. 위와 같이 평가요소별로 산정하여 우선순위를 자체적으로 결정하고, 우선순위별 가중치 계수를 참조하여 시큐어코딩 평가에 반영한다.

〔표 9〕 M-SCSA 평가요소별 우선순위

평가요소	우선순위(예)
입력데이터 검증 및 표현	7
API 악용	2
보안 특성	1
시간 및 상태	3
에러 처리	4
코드 품질	5
캡슐화	6
명명 규칙	8
연산 규칙	9

시큐어코딩 중요도에 따른 가중치 계수는 다음 표와 같다. 가중치 계수는 1~9 까지 주관적인 기준에 따라 계수를 정할 수 있으며, 본 연구에서는 다음 표에 따라 가중치를 적용한다.

〔표 10〕 우선순위별 가중치 계수

우선순위	가중치
1	1.4
2	1.3
3	1.2
4	1.1
5	1.0
6	0.9
7	0.8
8	0.7
9	0.6

시큐어코딩 항목별 중요도에 의해 결정된 우선순위에 따라 시큐어코딩을 적용하고, 적용여부에 의하여 시큐어코딩 평가요소별 총 평가점수를 산정한다. 물론 시큐어코딩의 적용 여부는 코드레벨에서의 검토가 필요하나, 이는 현실적으로 어렵다. 다만, 본 논문에서 제시하는 자가평가와 같이 스스로 적용여부를 체크하여 평가해 볼 수 있다.

시큐어코딩의 적용 여부에 따른 시큐어코딩의 준수성을 평가한다. 시큐어코딩의 준수성은 M-SCSA의 종합평가를 산정하기 위한 기초자료로서 활용된다. 시큐어코딩 적용에 따른 준수성 여부의 평가점수를(T), 각 평가요소별 시큐어코딩 적용여부(Y) 개수를 (E_i), 시큐어코딩 전체항목 개수를(E_{MAX}), 시큐어코딩 제

외항목(N/A)의 개수를 (E_n)이라고 하면, 산정식은 다음과 같다.

$$T = \frac{Ei}{E_{MAX} - E_n} \times 100$$

마지막으로 중요도에 따른 우선순위별 가중치 계수와 시큐어코딩 적용에 따른 준수성 평가점수를 이용한 시큐어코딩의 종합평가는 다음과 같다. 준수성 평가점수(T_i), 우선순위별 가중치 계수(λi)를 합산하여 종합평가 점수(Z)를 산출한다. (Λ는 가중치 총합)

$$Z = \sum_{i=1}^{n} \frac{T_i \times \lambda_i}{\Lambda}, \quad \Lambda = \sum_{i=1}^{n} \lambda_i$$

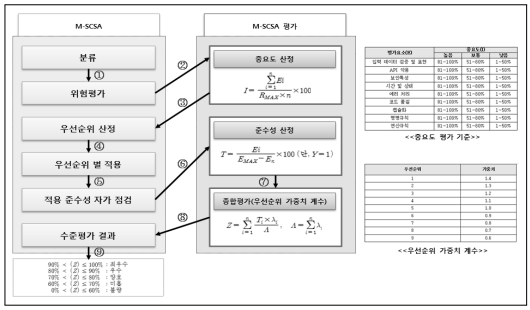
본 논문에서 제시하는 M-SCSA 종합평가는 시큐어코딩 항목별 위험평가 결과에 따라 중요도를 산정하여 우선순위를 산정한 후, 우선순위에 따라 시큐어코딩을 적용한다. 이후 시큐어코딩의 적용여부인 준수성을 스스로 검토하여 준수성을 산정한다. 다음으로 중요도에 따른 우선순위 가중치 계수(λ_i)와 준수성 평가점수(T)를 합산하여 종합평가(Z)를 산정한다. 종합평가(Z)에 따라 5개의 등급으로 M-SCSA 수준을 평가한다.

$90\% < (Z) \leq 100\%$: 최우수
$80\% < (Z) \leq 90\%$: 우수
$70\% < (Z) \leq 80\%$: 양호
$60\% < (Z) \leq 70\%$: 미흡
$0\% < (Z) \leq 60\%$: 불량

3.3 M-SCSA 평가 방법 예

M-SCSA의 평가 절차는 〔그림6〕과 같이 크게 9단계로 나누어진다.

M-SCSA 평가 절차 9단계는, **〔1단계〕** 시큐어코딩의 분류 ⇨ **〔2단계〕** 시큐어코딩 항목별 위험평가 ⇨ **〔3단계〕** 위험평가 결과에 따른 중요도 산정 ⇨ **〔4단계〕** 중요도에 따른 우선순위 산정 ⇨ **〔5단계〕** 우선순위별 시큐어코딩 적용 ⇨ **〔6단계〕** 시큐어코딩 적용에 따른 준수성 자가 점검 ⇨ **〔7단계〕** 준수성 산정 ⇨ **〔8단계〕** 준수성 산정 결과와 우선순위 가중치를 합산한 종합평가 산정 ⇨ **〔9단계〕** M-SCSA 수준평가 결과로 완료된다.



〔그림 6〕 M-SCSA 평가 절차 9단계

3.3.1 M-SCSA 평가요소별 중요도 산정 예

평가요소별 위험평가 결과를 중요도 산정식에 대입하여 시큐어코딩 항목별 중요도를 산정한다.〔표 7 참조〕

$$T = \frac{\sum_{i=1}^{n} Ei}{R_{MAX} \times n} \times 100$$

T = {(E001) + (E002) + (E003) + (E004) + (E005) + (E006)+ (E007) / (27) X 7} X 100
= {(12+12+12+4+4+4+1) / 189} X 100
= {49 / 189} X 100
= 25.92%

이므로, 입력데이터 검증 및 표현의 중요도는 낮음으로 평가된다.

3.3.2 M-SCSA 평가요소별 준수성 산정 예

시큐어코딩 항목별 중요도에 의해 결정된 우선순위에 따라 시큐어코딩을 적용하고, 적용여부를 평가하여 준수성 산정식에 대입한다.

〔표 11〕 입력데이터 검증 및 표현의 준수성 자가 점검 결과

No	시큐어코딩 식별명	적용여부
001	상대 디렉터리 정보 조작	Y
002	절대 디렉터리 정보 조작	Y
003	메서드 파라메타를 적절히 검사한다.	Y
004	생성자에서 재정의될 수 있는 메서드를 호출하지 않는다.	N
005	반복문 루프카운터 변수로 실수 값을 사용하지 않는다.	Y
006	wait()(또는 await())는 루프(loop)문에서 호출한다.	N/A
007	올바르게 정수를 표현한다.	Y

$$T = \frac{Ei}{E_{MAX} - E_n} \times 100$$

위의 산정식에 따라, 시큐어코딩 항목을 적용한 경우 Y의 개수는 5, N/A 개수는 1이므로,

T = {(E001) + (E002) + (E003) + (E005) + (E007) / (7 - 1} X 100
= {5 / 6} X 100
= 83.33%

으로, 입력데이터 검증 및 표현의 준수성을 산정할 수 있다.

3.3.3 M-SCSA 종합평가 산정 예

마지막으로 중요도에 따른 우선순위별 가중치 계수와 시큐어코딩 적용에 따른 준수성 평가점수를 종합평가 산정식에 대입하여 시큐어코딩의 종합평가를 산정한다.

〔표 12〕 M-SCSA 우선순위, 가중치, 준수성 산정결과 예

평가요소	우선순위(예)	가중치	준수성 산정 결과(예)
보안 특성	1	1.4	66.66%
API 악용	2	1.3	100%
시간 및 상태	3	1.2	60%
에러 처리	4	1.1	33.33%
코드 품질	5	1.0	66.66%
캡슐화	6	0.9	100%
입력데이터 검증 및 표현	7	0.8	83.33%
명명 규칙	8	0.7	66.66%
연산 규칙	9	0.6	72.72%

$$Z = \sum_{i=1}^{n} \frac{T_i \times \lambda_i}{\Lambda}, \quad \Lambda = \sum_{i=1}^{n} \lambda_i$$

$$Z = \frac{T_1 \times \lambda_1}{9} + \frac{T_2 \times \lambda_2}{9} + \frac{T_3 \times \lambda_3}{9} + \frac{T_4 \times \lambda_4}{9} + \frac{T_5 \times \lambda_5}{9} + \frac{T_6 \times \lambda_6}{9} + \frac{T_7 \times \lambda_7}{9} + \frac{T_8 \times \lambda_8}{9} + \frac{T_9 \times \lambda_9}{9}$$

이므로,

Z = (66.66% × 1.4 / 9) + (100% × 1.3 / 9) + (60% × 1.2 / 9) + (33.33% × 1.1 / 9) + (66.66% × 1.0 / 9) + (100% × 0.9 / 9) + (83.33% × 0.8 / 9) + (66.66% × 0.7 / 9) + (72.72% × 0.6 / 9)
= 10.36 + 14.44 + 8 + 4.07 + 7.40 + 10 + 7.40 + 5.18 + 4.84
= 71.69%

이므로, 종합평가 결과 "양호"로 평가된다.

IV. 결론

본 논문에서는 모바일 어플리케이션의 시큐어코딩 적용에 따른 자가평가 방법(M-SCSA)을 연구하여 제안하였다. 개인・단체・조직에 따라 다양한 환경과 개발 과정에서 공통적으로 적용되는 시큐어코딩 항목과 그렇지 않은 시큐어코딩 항목이 존재한다. 이러한 시큐어코딩 항목들은 본 논문에서 제안하는 M-SCSA 평가 모델을 활용하여 지속적으로 관리・축적하여 개인・단체・조직의 환경에 최적화된 시큐어코딩을 적용할 수 있을 것이다. 이를 통해 체계적이고 효율적인 자가평가 기준을 마련한다면 보안성이 확보된 모바일 어플리케이션 개발에 활용이 가능할 것이다.

참고문헌

〔1〕 행정안전부, 정보시스템 SW개발・운영자를 위한 소프트웨어 개발보안 가이드, http://www.mopas.go.kr/gpms/view/jsp/download/userBulletinDownload.jsp?userBtBean.bbsSeq=1012390&userBtBean.ctxCd=1002&userBtBean.orderNo=5, 2011년 6월

〔2〕 한국인터넷진흥원, 스마트폰 어플리케이션 마켓중심의 정보보호 대응 방안 연구, http://www.kisa.or.kr/jsp/common/libraryDown.jsp?folder=017163, 2010년 9월

〔3〕 행정안전부, SW 개발단계부터 보안약점 제거(시큐어코딩) 의무화, http://www.mopas.go.kr/gpms/view/jsp/download/userBulletinDownload.jsp?userBtBean.bbsSeq=1022211&userBtBean.ctxCd=1012&userBtBean.orderNo=1, 2012년 5월

〔4〕 오준석, "안드로이드 앱 시큐어 코딩 표준," 석사학위논문, 고려대학교 대학원, 2010년 12월

〔5〕 CWE, 2010 CWE/SANS Top 25 Most Dangerous Software Errors, http://cwe.

mitre.org/top25/index.html
〔6〕 Gary McGraw, Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors, IEEE Security and Privacy Magazine, Vol.3. No.6, pp.81-84, Nov. 2005.
〔7〕 행정안전부, Android-JAVA 시큐어 코딩 가이드, http://www.mopas.go.kr/gpms/view/jsp/download/userBulletinDownload.jsp?userBtBean.bbsSeq=1012390&userBtBean.ctxCd=1002&userBtBean.orderNo=9, 2011년 6월
〔8〕 Steve McConnell, CODE COMPLETE 소프트웨어 구현에 대한 실무서 2판, 정보문화사, 2005년 4월.
〔9〕 Robert C. Seacord, 버그 없는 안전한 소프트웨어를 위한 CERT C 프로그래밍, 에이콘, 2010년 2월
〔10〕 한명묵, 이철수, 정보보호개론, 서울:정익사, 2008년 3월
〔11〕 SK텔레콤, 안드로이드 개발 보안 지침서, 인포섹(주), 2010년 12월
〔12〕 FMECA, Failure Modes and Effects Analysis (FMEA) and Failure Modes, Effects and Criticality Analysis (FMECA), http://www.weibull.com/basics/fmea.htm, MIL - P - 1629
〔13〕 Charlie Lai, "Java Insecurity: Accounting for Subtleties That Can Compromise Code" Software, IEEE, pp.13-19, Feb. 2008
〔14〕 Gary McGraw, Software Security, Addison-Wesley, Feb. 2006
〔15〕 John Viega and Gary McGraw, Building Secure Software, Addison-Wesley, Seb. 2001
〔16〕 Lynn Futcher and Rossouw von Solms, "Guidelines for Secure Software Development," ACM Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology, pp.56-65, 2008.

부 록

M-SCSA 시큐어코딩 식별명 및 분류표

1. 시큐어코딩 분류번호표

No	분류항목	분류번호	설 명
1	입력데이터 검증 및 표현	IVP	Input data Validation and Presentation
2	API 악용	API	API Exploit
3	보안 특성	SEC	SECurity property
4	시간 및 상태	TMS	TiMe and Status
5	에러 처리	EOR	ErrOR handling
6	코드 품질	COD	CODe quality
7	캡슐화	CAP	enCAPsulation
8	명명 규칙	NAM	NAMing Rules
9	연산 규칙	OPR	OPRration Rules

2. 입력데이터 검증 및 표현 분류표

분류항목	분류번호	No	시큐어코딩 식별명
입력데이터 검증 및 표현	IVP	001	상대 디렉터리 정보 조작
		002	절대 디렉터리 정보 조작
		003	메서드 파라메타를 적절히 검사한다.
		004	생성자에서 재정의될 수 있는 메서드를 호출하지 않는다.
		005	반복문 루프카운터 변수로 실수값을 사용하지 않는다
		006	wait()(또는 await())는 루프(loop)문에서 호출한다.
		007	올바르게 정수를 표현한다.

3. API 악용 분류표

분류항목	분류번호	No	시큐어코딩 식별명
API 악용	API	001	null 매개변수 미검사
		002	equals()와 hashCode() 하나만 정의
		003	3.4.1 액티비티(Activity) 클래스
		004	3.4.2 인텐트(Intent) 클래스
		005	3.4.3 서비스(Service) 클래스
		006	3.4.4 대화상자(AlertDialog) 클래스
		007	3.4.5 토스트(Toast) 박스 클래스
		008	3.4.6 미디어플레이어(MediaPlayer) 클래스
		009	3.4.7 이벤트 처리
		010	3.4.8 캔버스(Canvas) 클래스

4. 보안 특성 분류표

분류항목	분류번호	No	시큐어코딩 식별명
보안특성	SEC	001	기밀 정보의 단순한 텍스트 전송
		002	취약한 암호화 알고리즘의 사용
		003	적절하지 않은 난수값의 사용
		004	전역적으로 접근가능한 파일
		005	외부에서 접근하여 활성화 가능한 컴포넌트
		006	공유 아이디에 의한 접근제어 통과

5. 시간 및 상태 분류표

분류항목	분류번호	No	시큐어코딩 식별명
시간 및 상태	TMS	001	경쟁 조건: 검사시점과 사용시점
		002	제대로 제어되지 않은 재귀
		003	Thread.stop()로 스레드를 중지시키지 않는다.
		004	"thread-safe"메서드를 "thread-unsafe"메서드로 재정의하지 않는다.
		005	Thread.run()으로 스레드를 실행하지 않는다.
		006	멀티 스레드 프로그램에서 적절한 동기화를 구현한다.
		007	"Double-Checked Locking" 기법을 사용하지 않는다.

6. 에러 처리 분류표

분류항목	분류번호	No	시큐어코딩 식별명
에러 처리	EOR	001	오류 메시지 통한 정보 노출
		002	오류 상황에 대한 처리 부재
		003	클래스 멤버변수를 외부에 노출하지 않는다.

7. 코드 품질 분류표

분류항목	분류번호	No	시큐어코딩 식별명
코드 품질	COD	001	널포인터 역참조
		002	"Little endian"을 사용하는 시스템과 통신시, 데이터를 "Little endian"으로 변환한다.
		003	public static키워드를 사용하지 않는다.
		004	finally블록에서 종료행위를 하지 않는다.

8. 캡슐화 분류표

분류항목	분류번호	No	시큐어코딩 식별명
캡슐화	CAP	001	공용 메소드로부터 리턴된 private 배열-유형 필드
		002	private 배열-유형 필드에 공용 데이터 할당
		003	시스템 데이터 정보 누출
		004	중첩클래스에서 외부클래스의 private 필드를 노출하지 않는다.

9. 명명 규칙 분류표

분류항목	분류번호	No	시큐어코딩 식별명
명명규칙	NAM	001	유사한 형태를 가진 문자열을 동시에 사용하지 않는다.
		002	동일한 타입을 갖는 변수들을 한 라인에 선언하지 않는다.
		003	복수의 명령문들을 한 라인에 작성하지 않는다.
		004	프로그램에서 상수를 사용한다.
		005	Long타입을 사용할 경우, l을 사용하지 않는다.
		006	중복된 이름을 사용하지 않는다.
		007	직관적인 명칭이름을 사용한다.
		008	복합어는 구분되도록 사용한다.
		009	명명규칙을 준수한다.
		010	final 객체를 의도한 경우, 내부 멤버들도 final로 선언한다.

10. 연산 규칙 분류표

분류항목	분류번호	No	시큐어코딩 식별명
연산규칙	OPR	001	나머지연산자(%)의 결과 값에 주의한다.
		002	올바른 방법으로 두 배열객체들을 비교한다.
		003	char 로 read()하지 않는다.
		004	시프트 연산자를 남용하지 않는다.
		005	할당연산자의 동작순서를 인지한다.
		006	'NaN' 판별시, 'NaN'을 비교연산자에 사용하지 않는다.
		007	어떤 객체를 실수형으로 변환시, 범위검사를 한다.
		008	배열첨자는 0 부터 사용한다.
		009	연산 후, 연산동작이 올바르게 수행됨을 보장한다.
		010	묵시적 캐스팅 시, 변수표현 범위 내에서 연산을 수행한다.
		011	괄호로 묶여진 명령문들은 연산자 우선순위에 의존하지 않는다.
		012	명시적 캐스팅시, 변수표현 범위 내에서 연산을 수행한다.
		013	올바른 방법으로 문자열을 비교한다.

〈著者紹介〉

김 동 원 (Dong-Won Kim) 정회원
2009년 2월: 서울과학기술대학교 컴퓨터공학과 졸업
2012년 2월: 건국대학교 정보통신대학원 정보보호학과 석사
2012년 3월~현재: 고려대학교 정보보호대학원 정보보호학과 박사과정
2012년 6월~현재: 현대오토에버 정보보안기술팀 과장
〈관심분야〉 시큐어코딩, 정보보호, 모바일 보안, 지능형 차량 보안, 정형기법

한 근 희 (Keun-Hee Han) 종신회원
서울과학기술대학교 컴퓨터공학과 졸업
한양대학교 공과대학원 컴퓨터공학 석사(정보보호 전공)
고려대학교 대학원 이학박사(정보보호전공)
2002년 9월~현재: 건국대학교 정보통신대학원 겸임교수
2012년 3월~현재: 드림시큐리티 부사장
〈관심분야〉 인터넷 보안, 통합보안관리, 모바일 보안, 차세대 인터넷 등