정 보 보 호 학 회 지
제25권 제1호, 2015. 2

# 시큐어 SDLC 시각의 시코어코딩 활용과 평가

서 동 수*

요 약

악의적인 공격에 대해 안전한 소프트웨어를 개발하고자 하는 보안강화 활동은 소프트웨어개발 생명주기(SDLC)의 모든 단계에서 수행되어야 한다. 시큐어코딩은 개발 단계에서 적용될 수 있는 안전한 코딩 기법으로 실행코드가 지닐 수 있는 취약성의 근본 원인을 소스코드 수준에서 제거하고자 하는 시도이다. 그럼에도 불구하고 시큐어코딩을 구현활동의 일부로만 국한시켜 보는 시각은 기법이 갖는 장점을 충분히 살리지 못할 수 있다. 외국에서는 이미 시큐어코딩의 적용과 평가를 SDLC 수준에서 시행하고 있으며 시큐어 SDLC로 분류되는 BSIMM과 SAMM, MS SDL은 이러한 시도의 대표적인 사례라 할 수 있다. 본 고에서는 이들 보안 프레임워크를 대상으로 시큐어코딩이 어떻게 정의되고, 수행되며, 평가되는지 비교를 통해 효과적인 시큐어코딩 활동의 이해를 돕고자 한다.

## Ⅰ. 서 론

최근 국내에서는 대형 포털사, 금융 및 카드사, 통신사, 의료 서비스 영역 등 광범위한 분야에서 보안 사고가 발생한 바가 있다. 특히, 2014년 1분기에만 총 1억 5,800만 건의 개인정보가 유출되었다고 보고된 바 있다[1]. 이 사건으로 인해 우리나라는 데이터 유출 침해부문 세계 1위라는 불명예도 떠안게 되었다. 이들 중 254건의 보안 사고는 데이터 암호화, 입력데이터 검증 등의 필수적인 보안 활동을 누락한 것이 직접적인 원인으로 지적되었다고 한다. 개발과정이 엄밀하고 체계적이었다면 피해갈 수도 있었을 법한 대목이다.

서구 여러 나라에서는 소프트웨어 개발 과정을 강화시킴으로써 소프트웨어의 보안성을 향상시키려는 시도를 오래전부터 해오고 있다. 이러한 시도의 대표적인 예로 미국 국토안보부의 보안강화 구상 (Building Security-In Initiatives)을 통해 소개된 보안강화 성숙도 모형 BSIMM[2], 비영리 기관인 OWASP에서 제안된 보안평가 성숙도모형 SAMM[3], 그리고 마이크로소프트사의 SDL[4] 등이 있다. 목적은 틀리지만 우리나라의 공공기관 정보시스템 구축단계에 적용되는 행정자치부의 소프트웨어 개발보안제도[5] 역시 많은 주목을 받고 있다.

본 고는 시큐어코딩의 개념과 활동의 중요성을 소프트웨어 개발 생명주기 시각에서 고찰한다. 특히 BSIMM, SAMM, MS SDL과 같은 보안강화 프레임워크에서는 시큐어코딩 활동이 어떻게 규정되며, 관리되는지를 살펴본다. 이를 통해 시큐어코딩은 단순히 코딩 단계에서 준수되어야 할 코딩 규칙 정도로 이해하기 보다는 소프트웨어개발 생명주기의 선후 활동과 깊은 관련이 있다는 사실을 강조하고자 한다.

## Ⅱ. 코딩표준과 시큐어코딩, 정적분석

코딩 표준은 코딩 과정에서 정해진 규적을 준수하도록 강제함으로써 코드의 품질을 높이는 활동을 말한다. 널리 알려진 코딩 표준으로는 자동차 산업계의 사실표준인 MISRA C 코딩 표준[6], 우주 항공 분야의 NASA JPL의 Java 코딩 표준[7], 국내 방위산업 분야의 무기체계 소프트웨어 코딩 표준[8] 등이 있다.

다양한 분야와 종류의 코딩 표준들이 있음에도 불구하고 이들이 공통적으로 추구하는 코딩 표준의 목적을 살펴보면 다음으로 요약이 가능하다.

- 코딩 오류 유입의 최소화: 방어적 프로그래밍을 통해 발생 가능한 오류에 대해 선제적 대응을 한다. 예를 들어 널 포인터 역참조의 가능성이 있는 변수는 사용전에 반드시 if문을 통해 널 값의 존재를 확인한다.

* 성신여자대학교 IT학부 (dseo@sungshin.ac.kr)

- 표준의 수용: 코딩표준을 통해 전사적 표준 뿐 아니라 국제 표준의 준수 여부를 확인한다. 예를 들어 자동차 전자제어 분야는 MISRA 코딩 규칙의 준수 여부를 확인할 있는 증거를 요구하는 경우가 이에 해당한다.
- 성능 및 호환성 향상: 코딩 표준의 준수를 통해 성능, 호환성에 영향을 주는 부분을 개선할 수 있다. 예를 들어 MISRA에서는 16바트와 32비트 자료간의 타입 변환에 관한 명시적인 규칙을 제시함으로써 이식성 문제에 대응한다.

시큐어코딩은 소프트웨어의 보안성을 향상시킬 목적으로 행해지는 프로그램에 대한 코딩 규약을 말하는 것으로 코딩표준의 보안 확장이다. 대표적인 시큐어코딩 표준으로는 CERT C[9]와 정보시스템 개발보안가이드[10]를 들 수 있다.

시큐어코딩은 인터넷 상에서의 보안 문제를 유발하는 여러 원인에 대한 선제적 대응을 목적으로 개발되었다. 이러한 점은 일반적으로 프로그램 오류를 강조하는 MISRA와 같은 코딩 표준과 차이를 보이는 부분이다. 예를 들어 대표적인 시큐어코딩 표준인 정보시스템 개발보안가이드는 보안 약점의 발생 원인을 입력데이터 검증 및 표현, 보안기능, 시간 및 상태, 에러처리, 코드오류, 캡슐화, API오용 등과 같은 7개의 카테고리로 구분하여 총 47개의 보안 약점을 설명하고 있다.

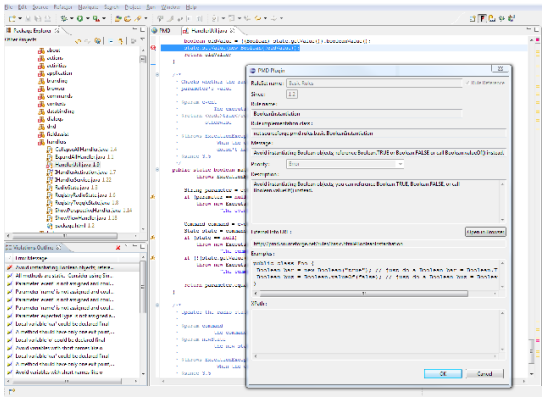
시큐어코딩은 종종 테스팅과 대비되어 언급된다. 구체적으로 여기서의 테스팅은 퍼징, 침투테스팅과 같은 실행을 기반으로 하는 테스팅을 말한다. 실행기반 테스팅은 실행환경에서 보안 오류인 보안 취약점(vulnerability)의 발견을 목적으로 테스트케이스를 자동 혹은 수동의 방법으로 프로그램에 주입시켜 확인하는 방법이다. 반면, 시큐어코딩은 보안취약점의 원인이 될 수 있는 소프트웨어의 결함, 실수, 혹은 버그로 지칭되는 보안약점(weakness)의 제거를 목적한다. 예를 들면, 보안약점 중 하나인 SQL 주입(injection) (CWE-89) 보안약점은 ZOHO 관리 엔진의 Create RepoprtTable.jsp에서 보안취약점(CVE2015-1479)[11]을 발생시키는 것으로 보고되었다.

시큐어코딩의 준수 여부를 확인하는 방법은 정적 분석을 사용하는 코드검토에 의존한다. 정적분석은 일반적으로 소스코드의 특성을 찾아내기 위해 비실행을 전재로 코드를 분석하는 방법이다. 정적분석은 초기에는 컴파일러에서 코드 최적화 방법으로 널리 사용된 기법이다. 그러나 최근에는 테스팅 비용을 절감하려는 목적으로, 혹은 소스코드를 작성하는 시점에서 코드 품질을 향상시키려는 목적으로 많이 사용되고 있다.

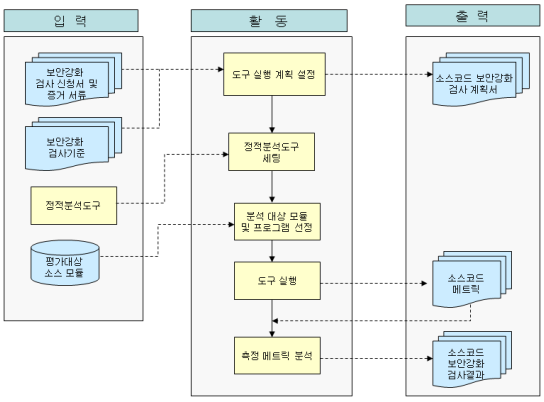
개발자들은 종종 정적분석을 디버깅 활동과 결합하여 즉흥적으로 실행한다. 예를 들어 오픈소스 정적분석 도구인 PMD는 Java개발환경인 이클립스에 플러그인 되어 사용할 수 있도록 진화되었다[12]. 그림 1과 같이 이 환경에서 개발자는 컴파일과 디버깅, 정적분석을 번갈아 가며 수행할 수 있다.

게리 맥그로우가 세븐 터치포인트[13]를 통해 주장했듯이 올바른 정적분석을 수행하기 위해서는 계획, 수행, 평가라는 세 단계의 활동이 필요하다.

먼저, 계획 단계에서는 보안 담당자는 소스코드 안전성 보증 평가를 수행하기 위한 계획과 절차를 수립하고, 독립적인 평가를 준비한다. 두 번째, 개발 단계에서는 개발 과정에서 소스코드 위험성이 적절한 수준에서 계획되고 관리 되었다는 증거를 제공한다. 이 때 개발자가 수행하는 코드 검사의 목표는 소스코드 품질 목표와도 일관성을 유지해야 한다. 마지막으로 평가 단계에서 평가자는 소스코드를 대상으로 정적분석 도구를 실행한다. 만일 인증 기준을 만족시키지 못한 코드에 대해서는 탈락 근거와 함께 검사 결과를 개발자에게 통보하여 개선을 요구한다. 그림 2는 평가의 시작으로부터 종료에 이르는 활동 과정을 역할별로 구분한 것으로 정적분석은 제 3자 평가를 통한 코드의 품질 개선이 중요함으로 보여준다.



〔그림 1〕 이클립스 플러그인된 PMD




〔그림 2〕 정적분석의 적용과정

## Ⅲ. 시큐어 SDLC와 시큐어코딩

시큐어코딩은 코딩 단계에서 일어나는 코드품질 확보에 관한 노력이다. 그러나 소프트웨어 개발 생명주기 관점에서 보면 코딩에는 요구분석과 설계에서 확정된 제약사항을 코드에 반영해야 하는 종속성이 존재한다. 만일, 소스코드에서 보안약점이 발견된다면 이는 프로그래머가 설계를 잘 못 이해하였거나, 혹은 설계의 오류가 코딩단계로 유입된 것이라 판단할 수 있다.

이러한 단계의 종속성은 설계와 코딩에서만 나타나는 것이 아니라 요구분석과 설계에도 나타날 수 있다. 따라서 코드 품질에 관한 고려는 이미 요구분석과 설계 때부터 고려되어야 하며, 보안강화 소프트웨어 개발 생명주기에서는 이러한 점을 강조한다.

본 절에서는 소프트웨어의 보안강화를 위한 목적으로 제안된 시큐어 소프트웨어 생명주기 모형(Software Development Lifecycle, SDLC) 중 대표적인 사례를 살펴본다.

### 3.1. BSIMM

BSIMM(Building Security In Maturity Model)은 미국 국토안보국의 지원을 받아 수행된 소프트웨어 보증(Software Assurance, SwA)프로젝트의 결과물 중 하나이다. BSIMM은 보안활동의 성숙도 수준을 영역별로 측정함으로써 소프트웨어 개발에 필요한 보안 능력 향상을 목표로 하는 개발 프레임워크이다. BSIMM은 2015년 현재 BSIMM4까지 발표되어 지속적인 개선이 이루어지고 있다.

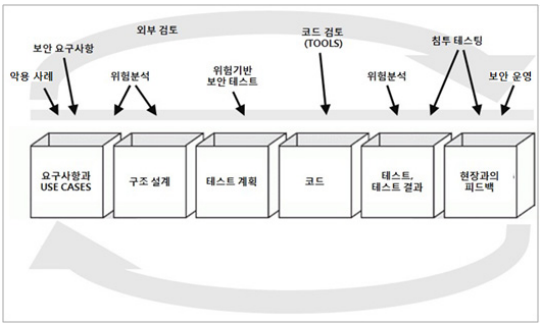
BSIMM은 표 1과 같이 12개의 보안 강화 활동들을 통해 정보시스템의 보안성을 향상시킬 수 있도록 설계되었다. 이들 12개 활동은 실무활동이라 불리며, 거버넌스, 인텔리전스, SSDL 터치포인트, 배치 등의 4개 도메인으로 분류되어 설명된다.

- 거버넌스: 보안그룹을 조직, 관리하고 보안활동을 측정하는 실무로 구성된다.
- 인텔리전스: 조직 전체의 SW 보안 활동의 수행에 사용된 지식자산의 축적에 관한 실무를 정의한다.
- 터치포인트: 그림 3에서와 같이 특정의 SW 개발 결과물에 대한 분석과 평가 그리고, 프로세스에 관련된 실무를 정의한다.
- 배치: 전통적인 네트워크 보안과 소프트웨어 유지보수 조직을 연결한 실무를 설명한다.

BSIMM에서 시큐어코딩은 터치포인트 카테고리 내의 코드리뷰 활동으로 정의된다. 터치포인트의 제안자인 맥그로우는 시큐어코딩이 BSIMM의 핵심 활동임을 파악하였으며, 나아가 위험분석, 오남용 케이스, 보안설계 등의 개념을 요구분석과 설계 활동에 추가하여 시큐어코딩과의 연계성을 강화했다.

〔표 1〕 BSIMM의 보안 프레임워크구성

| 거버넌스 | 인텔리전스 | 터치포인트 | 배치 |
|---|---|---|---|
| 전략과 매트릭 | 공격 모델 | 아키텍처 분석 | 침투 테스팅 |
| 정책과 순응성 | 보안 속성 설계 | 코드 리뷰 | SW 환경 |
| 트레이닝 | 표준과 요구사항 | 보안 테스팅 | 취약성 관리 |




〔그림 3〕 터치포인트 요소

### 3.2. Open SAMM

Open SAMM(Software Assurance Maturity Model)은 OWAPSP에서 개발한 개방형 보안 프레임워크이다. BSIMM이 국토안보국의 지원으로 진행된 공공부문의 보안 프레임워크라면 SAMM의 경우는 개방을 원칙으로 소규모, 중규모, 대규모로 점진적인 확대가 가능한 융통성있는 프레임워크를 표방한다. SAMM은 현재 버전 1.0이 발표되었다.

SAMM 프레임워크는 선행 개발된 BSIMM의 영향으로 표 2와 같이 4개의 영역, 12개 세부 항목의 보안 실무로 구성된다. 각 실무는 독립적인 소프트웨어 보안보증의 대상이 된다. SAMM은 각 실무 내에 달성 수준에 따라 초보수준에서 숙련수준까지의 3 단계 성숙도 수준을 제공함으로써 개발자의 보안 활동이 현재 어느 수준에 있는지를 체크리스트를 통해 확인할 수 있게 한다.

SAMM역시 검증이라는 개념에서 시큐어코딩을 주요하게 파악하고 있다. 특히, BSIMM과는 달리 설계리뷰와 코드리뷰, 그리고 보안테스팅을 3개의 중요한 검증 활동으로 정의함으로써 이들 활동 간의 연계성이 중요함으로 역설한다.

### 3.3. MS SDL

MS SDL(Security Development Lifecycle)은 마이크로소프트사가 2004년 이후 자사의 소프트웨어 개발에 의무적으로 적용하도록 고안한 보안강화 프레임워크이다. SDL은 개발 중인 제품이 보안 위협에 대해 얼마나 강인한가를 측정하기 위해 동일한 제품에 대해 “pre-SDL"과 “post-SDL"의 두 개의 버전으로 테스트를 한다. 이 결과를 통해 SDL의 적용이 제품의 보안성 향상에 얼마나 기여했는지를 입증할 수 있다.

- pre-SDL : SW 개발팀 모두가 최신 트랜드와 보안 개념을 학습하고 준비하는 단계이다.
- 1단계 (요구) : 버그분류, 프로세스 재정의, 보안 요구사항 수립한다.
- 2단계 (설계): 보안설계검토, 위협모델링, 위협모델 품질보증, 보안설계서 검토 및 승인을 수행한다.
- 3단계 (구현): 정적분석을 통해 시큐어코딩이 준수되도록 한다.
- 4단계 (검증): Fuzzing 테스트 실행 및 SW 동적 테스트를 수행한다.
- 5단계 (릴리즈): FSR(Final Security Review)를 받는 단계이다
- post-SDL : 보안 패치 등을 통해 발견된 오류 수정 및 보안 업그레이드 실행한다.

〔표 2〕 SAMM의 보안 프레임워크 구성

| 거버넌스 | 구축 | 검증 | 배치 |
|---|---|---|---|
| 전략과 매트릭 | 공격 모델 | 설계 리뷰 | 취약성 관리 |
| 정책과 순응성 | 보안 요구 | 코드 리뷰 | 환경 강화 |
| 교육과 가이드 | 보안 아키텍처 | 보안 테스팅 | 운영 개시 |

앞서 살펴본 다른 프레임워크과도 공통되는 사항이지만 MS SDL은 보안과 관련한 교육을 강조한다. 교육은 보안에 SDL에 대한 전문 지식과 보안성을 높이기 위한 코딩 기술이며 교육으로 모든 개발자가 의무적으로 이수하도록 강제한다.

시큐어코딩과 관련하여 SDL은 요구 분석 단계에서 버그분류 및 보안요구사항을 수립하여 소스코드를 평가를 위한 기준으로 활용하도록 한다. 이 관계가 입증이 된 후 개발된 제품은 보안상 오류를 최소화한 제품으로 판정가능하다. 또한, 위협 모델링을 통해 개발자들에게 밝혀진 취약점을 약화시킬 수 있는 소스코드가 개발되도록 하며, 개발자 역시 이러한 요구를 반영하는 소스코드 작성에 집중해야 한다.

구현 시에는 SDL은 각종 자동화된 도구를 활용하여 코드 내부에 취약성이 존재하는지 검사를 항시 수행하도록 요구한다. Visual Studio 환경에서 개발하는 것을 전제로 했을 때 기본적으로 개발환경 자체에서 제공하는 Code Analyzer를 활용하여 C/C++ 코드의 정적분석과 보안성 분석을 수행할 수 있다..

### 3.4. 개발보안가이드

2012년 6월 고시된 행정자치부의 정보시스템 구축·운영지침은 소프트웨어 개발보안 가이드를 통해 공공정보시스템에 생길 수 있는 보안약점을 사전에 제거하고 보안성을 높이도록 규정화하고 있다. 이 지침에서 언급된 ‘개발보안’의 핵심은 시큐어코딩을 기반으로 하는

보안약점의 제거이다.

개발보안 가이드는 각 단계의 활동을 통해 전자정부 시스템의 개발자로 하여금 47개 보안약점의 제거를 의무화한다는 점에서 앞서 살펴본 다른 보안 프레임워크와 차별성을 갖는다. 또한, 가이드는 시큐어코딩의 수행 여부를 확인하기 위한 보안전문가인 '진단원'의 활용 감리단계에서 요구한다.

개발보안가이드의 적용대상은 신규개발 소스코드와 유지보수 단계에 작성된 코드이다. 이는 개발보안 활동이 코딩 뿐 아니라 유지보수 단계까지도 대상으로 포함한다는 점에서 기존 시큐어코딩의 개념을 확장시킨 것은 분명하다. 그럼에도 불구하고 설계와 분석 단계에 대해서는 구체적인 활동의 정의를 담고 있지 않은 점에 대해서는 활용이 제한적일 수 있는 단점이 있다.

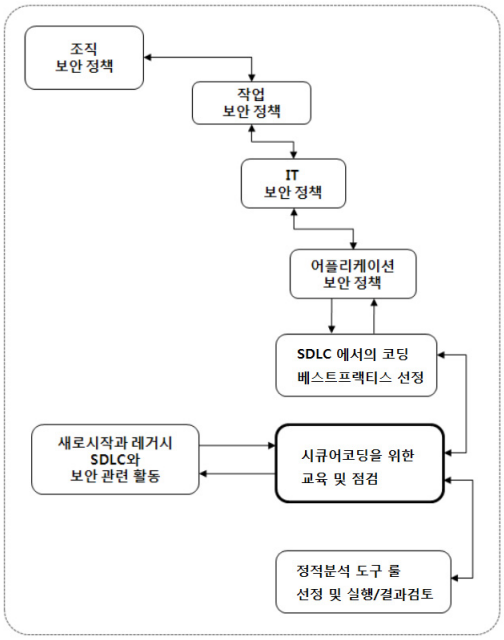
## Ⅳ. 성공적인 시큐어코딩을 위한 고려사항

시큐어코딩은 코딩 단계에만 적용되는 활동이라 생각하기 쉽다. 또한 시큐어코딩의 준수성은 정적분석도구를 사용하면 쉽게 파악될 수 있다고 믿기도 한다. 하지만, 시큐어코딩은 코딩 단계에만 적용되는 활동이 아니며, 정적분석도구만으로는 이를 해결해줄 수도 없다는 점을 인식하는 것은 중요하다.

본 고에서 살펴본 대부분의 보안강화 개발프레임워크에서는 보안계획단계에서부터 시큐어코딩의 목표를 정하도록 요구한다. IT보안 정책은 소프트웨어의 개발과정에서부터 운영까지의 전체 단계에 대한 포괄적인 보안 정책을 규정한다. 따라서 시스템 개발 초기서부터 계획이 주어진다면 시큐어코딩의 수준과 코딩 룰의 설정, 도구의 활용 방법 등에 관한 사전 계획이 도출될 수 있다. 그림 4는 보안정책이 시큐어코딩에 반영되는 과정의 단계를 보여준다.

정적분석도구의 실행은 그림 5에서와 같이 여러 가지 제반 활동을 수반한다. 먼저 도구 실행 전에 정적분석활동의 목표가 설정되어야 한다. 오탐 및 미탐의 수준을 정의하고, 어떤 수준의 탐지 룰을 사용할 것인가, 누가 담당할 것인가에 관한 정책 역시 전제가 되어야 하는 활동이다.

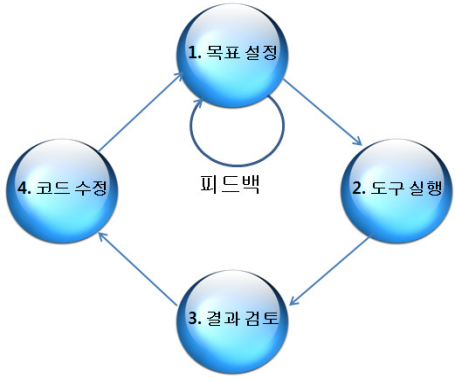
도구의 실행은 시큐어코딩의 준수성을 검증하는 중요한 활동이다. 도구 실행을 통해 얻어진 결과를 토대로 결과검토를 통해 개발자는 항상 설정된 목표를 만족시




〔그림 4〕 보안정책과 시큐어코딩의 관계

키는 수준의 확인을 했는지를 염두에 두어야 한다.

많은 경우 정적분석 도구는 진단과 더불어 소스코드에 대한 메트릭을 제공한다. 유용한 코드 메트릭으로는 사이클로메틱 복잡도 같은 내부 복잡도, 정보흐름 복잡도와 같은 모듈 간 외부 복잡도, 클래스나 함수의 응집도, 결합도와 같은 모듈 복잡도 메트릭은 코드이 유지보수와 이해도 판단에 도움이 되는 정보이다.




〔그림 5〕 시큐어코딩 준수를 점검하는 단계

## Ⅴ. 결 론

시큐어코딩은 악의이거나 의도하지 않았던 오류로부터 시스템을 안전하게 보호하기 위한 선제적 대응 이다. 기존의 침투 테스팅이나 보안패치가 수동적이며 사후대응 성격이라면 시큐어코딩을 통한 안전한 소스코드이 확보는 적극적이며, 장기적으로는 비용 감소 효과를 가져올 수 있는 좋은 실무활동이다.

최근 국내에서는 개발보안과 같은 제도, 혹은 인증과 같은 필요성에 의해 시큐어코딩이 확산되고 있는 추세이다. 그러나 BSIMM, SAMM, 혹은 MS SDL의 경우를 통해 살펴보았듯이 경쟁력있는 외국의 개발자들은 이미 10년 전부터 소프트웨어 생명주기에 근거하여 시큐어코딩을 활용하고 있다.

시큐어코딩의 효과를 극대화하기 위해서는 성능 좋은 도구의 확보도 중요한 일이다. 하지만 그것보다도 교육을 통해 개발자, 혹은 보안담당자에게 시큐어코딩의 이해를 충분히 하도록 해야한다. 개발 생명주기에 근거한 적절한 보안 계획의 수립 역시 중요한 활동이다.. 이를 기반으로 하는 시큐어코딩의 목표설정과 실행, 그리고 평가라는 순환 사이클을 통해 접근한다면 보안 문제에 대해 충분히 강화된 소스코드 확보가 가능할 것이다.

## 참 고 문 헌


[1] 데이터넷, http://www.datanet.co.kr/news/ article-View.html?idxno=72320

[2] Building Security In Maturity Model 2, http://bsimm2.com/index.php, 2010

[3] OWASP, Software Assurance Maturity Model, version 1,0, 2009. 3

[4] Security Development Lifecycle, http://www.microsoft.com/security/sdl/default.aspx 마이크로소프트, 2015

[5] 행정기관 및 공공기관 정보시스템 구축·운영 지침, 행정자치부, 2013. 8

[6] Motor Industry Software Reliability Association (MISRA), MISRA-C: 2012

[7] JPL Institutional Coding Standard for the C Programming Language, 2009

[8] 무기체계 소프트웨어 개발 및 관리지침서, 부록 6, 무기체계 소프트웨어 코딩 규칙, 방위사업청

[9] The CERT® C Coding Standard, Second Edition: 2014

[10] 안전행정부, 소프트웨어 개발보안 가이드, 2013

[11] CVE-2015-1479, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1479

[12] PMD, http://pmd.sourceforge.net/eclipse/

[13] Gary Mcgraw, Software Security: Building Security In, Addison-Wesley, 2006


## 〈저 자 소 개〉


**서 동 수 (Dongsu Seo)**
정회원

1987년 : 중앙대학교 컴퓨터공학과 (이학사)
1990년 : 맨체스터대학교, Depart. of Computation(공학석사)
1994년 : 맨체스터대학교, Depart. of Computation(공학박사)
1998~현재 : 성신여자대학교 IT학부 교수