정 보 보 호 학 회 지
제26권 제1호, 2016. 2

안전한 소프트웨어 개발을 위한 시큐어 SDLC 동향

박 난 경*, 임 종 인**

요 약

최근 사이버 공격은 분야와 대상을 막론하지 않고 곳곳에서 발생하고 있으며 소프트웨어의 보안 취약점을 이용한 지능적인 수법으로 지속적인 공격을 수행하는 APT 공격 또한 확산하고 있다. 이와 같은 공격을 예방하기 위해서는 공격에 직접 이용되는 소프트웨어 보안 취약점을 사전에 제거해야 한다. 소프트웨어 보안 취약점(vulnerability)의 원천 원인은 소프트웨어 허점, 결점, 오류와 같은 보안 약점(weakness)이다. 그러므로 소프트웨어에서 보안 약점은 개발 단계에서 완전히 제거하는 것이 가장 좋다. 이를 위해 소프트웨어 개발 생명주기(SDLC:Software Development Life Cycle) 전반에 걸쳐 보안성을 강화하는 활동을 수행한다. 이는 소프트웨어 배포 이후에 발생할 수 있는 보안 취약점에 대한 보안 업데이트 및 패치에 대한 비용을 효과적으로 감소시키는 방안이기도 한다. 본 논문에서는 소프트웨어 개발 단계 보안을 강화한 소프트웨어 개발 생명주기로서 시큐어 SDLC에 대한 주요 사례를 소개한다.

Ⅰ. 서 론

본격적인 스마트 시대의 진입을 앞두고 스마트 기기를 구성하고 있는 소프트웨어에 대한 보안성이 중요하게 되었다. 특히 자동차나 의료기기 등과 같이 안전 중요 시스템에서 보안 취약점으로 인해 사이버 공격이 현실화됨에 따라 대상 소프트웨어의 보안성은 안전성, 신뢰성과 함께 더욱 중요하게 되었다. [7] 소프트웨어의 보안성을 강화하기 위해서는 해킹 등에 이용되는 소프트웨어 보안 취약점을 사전에 제거해야 한다. 소프트웨어 보안 취약점(vulnerability)의 근본원인은 소프트웨어 허점, 결점, 오류와 같은 보안 약점(weakness)이다. 그러므로 소프트웨어 보안성 확보를 위해서는 소프트웨어 개발 초기부터 보안 약점을 완전히 제거하기 위한 노력이 필요하다. 시큐어 SDLC는 SDLC 전반에 걸쳐 보안 기법을 적용함으로써 보안 약점을 제거하려는 노력의 일환이다. 동시에 소프트웨어 배포 이후에 발생할 수 있는 보안 취약점에 대한 보안 업데이트 및 패치에 대한 비용을 효과적으로 감소시키는 방안이기도 한다. [8]

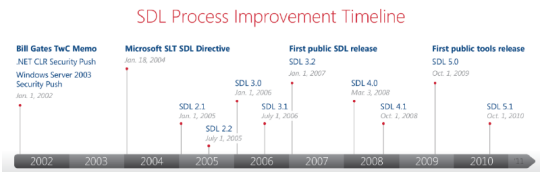
시큐어 SDLC의 대표적인 사례로 마이크로소프트사의 SDL (Security Development Lifecycle) [2,3,4,6] 과 미 국토안보부를 중심으로 한 Seven Touchpoint가 있다 [1]. 국내에는 대기업인 A사의 정보시스템 구축을 위해 적용되었던 P-SDLC 가 있다.[5]

본문의 구성은 다음과 같다. 제2장과 3장에서 국외 대표적인 사례인 MS-SDL과 Seven Touchpoint를 소개한다. 제4장에서는 국내 기업에 실제 적용한 바 있는 시큐어 SDLC 사례로서 P-SDLC를 소개하고 5장에서 시큐어 SDLC에 대한 적용 효과를 소개하고 결론을 맺는다.

Ⅱ. MS Security Development Lifecycle

마이크로소프트사는 2002 년 1월에 MS의 최고 경영자에 의한 Trustworthy Computing(TwC) 메모를 기점으로 소프트웨어 개발 시 보안을 최우선으로 고려하였다. 많은 소프트웨어 개발 그룹은 운영체제의 보안을 향상하기 위해 소프트웨어 배포 전에 위협 모델 갱신, 코드 검토, 테스트에 초점을 맞춘 보안 푸쉬를 적용하여 괄목할 만한 성과를 냈다. 2004년 이후 본격적으로 MS 내의 모든 소프트웨어 개발 프로세스에 걸쳐 보안을 필수적으로 적용하도록 제도화하였다. 2004년에 만들어진 MS SDL2.0은 최초의 공식적인 버전으로 위협 모델링, 정적분석, 최종 보안 검토 등의 활동으로 이루어졌

* 고려대학교 정보보호대학원(ranpark@korea.ac.kr)
** 고려대학교 정보보호대학원 교수 (jilim@korea.ac.kr)



〔그림 1〕 MS SDL의 주요 이정표와 타임라인

다. SDL은 ICT 변화를 수용하여 계속 발전하였고 2007년 이후 공개 배포되어 시큐어 SDLC의 주요 사례가 되었다.[4]

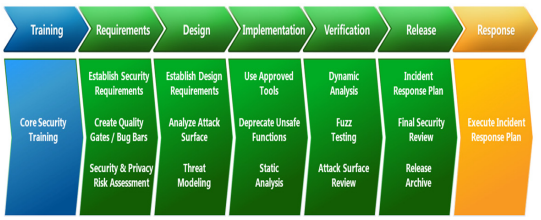
2.1. MS SDL의 기본 원리

MS가 도입한 소프트웨어 개발보안을 위한 기본 원리는 secure by design, secure by default, secure by deployment, communication (SD3+C) 이다. [6]

- Secure by Design: 보안 아키텍처, 설계 및 구조에서 보안 이슈를 고려하고, 위협 모델링 및 위협 완화 방안을 제시, 분석 및 테스트 도구를 사용하여 보안 취약점을 제거, 보안 수준이 낮은 기존 프로토콜과 코드의 사용을 중지
- Secure by Default: 최소권한으로 실행, 심층 방어를 위한 복수의 위협 완화 방안 고려, 디폴트 설정을 최소화, 위험한 디폴트 설정에 대한 변경 회피, 80% 이하 사용자에게 제공되는 서비스에 대한 디폴트 설정을 해제
- Secure by Deployment: 배포가이드 생성, 분석 및 관리 도구를 활용하여 최적의 보안 수준 설정, 패치 배포 도구 제공
- Communications: 보안 취약점 보고서 및 보안 업데이트 신속 대응, 커뮤니티에서 보안 관련 사용자 질문 대응에 적극적으로 참여

MS SDL은 기본 원리인 SD3+C를 SDLC와 결합한 프로세스이다. 각 개발단계에 SD3+C의 보안 활동을 대응시킨 결과 그림 2와 같은 형태로 발전하였다.[6]

MS SDL은 Pre-SDL (교육) 단계, 요구사항 단계, 설계 단계, 구현 단계, 검증 단계, 배포 단계, Post-SDL (대응) 단계로 구성되어 있다. MS SDL에서는 단계별 보안활동의 적용 수준을 필수 보안 활동과 선택 보안 활동으로 구분하고 있다. 필수 보안 활동의 경우는 반드시 수행해야 하는 활동이며 그림 2와 같이 16 개로 구성되어 있다. 선택 보안 활동은 상황에 따라 적용할 수



〔그림 2〕 단순화한 MS SDL

있는 활동이다. [3]

2.2. 개발 단계별 필수 보안 활동

단계별로 수행되어야 할 보안 활동을 필수 보안 활동을 중심으로 설명한다.

2.2.1. Pre-SDL 단계 : 보안 교육

개발 부서의 모든 구성원은 연 1회 이상 보안 및 개인정보보호의 기본 개념과 최신 동향에 대한 교육을 이수하여야 한다. 필수적으로 알아야 할 보안 기본 개념은 보안 설계, 위협 모델링, 시큐어 코딩, 보안 테스팅, 개인정보보호 등이다. 필요에 따라 심화 개념인 보안설계와 아키텍처, 사용자 인터페이스 설계, 보안 취약점, 위협 완화 방안 구현 등을 포함한다.

2.2.2. 요구사항 단계

보안 요구사항을 수립, 품질 게이트와 버그 바를 정의, 보안 및 개인정보보호에 대한 위험 평가를 수행한다.

- 보안 요구사항 수립: 보안 책임자를 할당하고 소프트웨어에 대한 최소 보안요건을 정의한다. 또한, 버그 유형과 작업 추적 도구를 명세화한다. 보안 책임자는 보안 추적관리, 보안 이슈에 대한 조정 및 의사소통 임무를 수행하며 규모에 따라 팀이나 개인으로 할당한다. 보안 버그와 작업 항목 추적시스템을 명세화하여 배포한다.
- 품질 게이트와 버그 바 정의: 보안 위험을 이해하고 보안 버그를 식별하고 수정하기 위해 품질 게이트/버그 바, 보안을 위한 최소 허용 수준을 보안 책임자의 승인 아래 정의한다. 이는 추후 최종 보안 검토를 위한 기준이 된다. 예외 승인 프로세스를 정의하며 보안 부서는 예외에 따른 잠재적 위험을 이해하고 배포 후에 대한 위험 완화 계획을 수립해야 한다.

- 보안 및 개인정보보호 위험평가: 심층 검토가 필요한 영역을 식별하고 다음과 같은 정보를 파악한다.
  - 위협 모델, 보안 설계 검토가 필요한 부분
  - 외부 인력에 의한 침투테스트가 필요한 부분
  - 퍼즈 테스트에 대한 세부 범위
  - 개인정보보호 영향 등급 (상, 중, 하)

2.2.3. 설계 단계

보안 설계 요구사항을 수립, 공격 표면 분석, 위협 모델을 완성한다.

- 보안 및 개인정보보호를 위한 설계를 검토하고 암호학적 설계 요구사항을 수립한다.
- 코드 접근 보안, 방화벽 예외 관리 등 공격 표면을 분석한다.
- 비용분석 단계에서 위협모델을 완료하며 개발자, 검사자, 프로그램 관리자가 승인한다. 공격표면에 노출된 모든 코드와 제삼자 코드까지 포함하며 최소 품질 조건을 만족한다.

2.2.4. 구현 단계

승인받은 도구를 사용하고, 안전하지 않은 기능은 사용하지 않아야 하며, 코드에 대한 정적분석을 수행하여 시큐어 코딩을 준수한다

- 개발부서는 작업 도구에 대한 명세와 컴파일러/링커 옵션과 경고 등의 보안 점검사항을 정의하고 보안 담당자의 승인 아래 게시한다.
- 안전하지 않은 함수를 식별하고 사용을 금지하기 위해 사용 금지 목록을 정의한 후, 이를 점검할 수 있는 헤더 파일을 사용하거나 코드 점검용 스캐닝 도구를 사용한다.
- 소스코드에 대한 정적 분석을 주기적으로 수행하며 정적 분석 도구는 보안 코드 검토를 위한 확장성을 제공하고 시큐어 코딩 정책의 적용을 보장해야 한다. 일반적으로 다른 도구를 보완하여 사용하거나 인적 검토를 병행한다.

2.2.5. 검증 단계

요구사항과 설계 단계에서 수립된 보안과 개인정보보호 목적에 따라 테스트를 수행한다. 실행프로그램에 대한 동적 코드 분석과 퍼즈 테스팅, 공격표면 확인을 포함한다.

- 동적 분석: 런타임 검증을 통해 프로그램이 설계대로 작동함을 보장한다. 메모리 변조, 사용자 권한, 기타 중대한 보안 문제에 대한 애플리케이션의 행위를 감시할 수 있는 도구를 사용한다.
- 퍼즈 테스팅: 애플리케이션에 잘못된 데이터나 무작위 데이터를 적용하여 프로그램 실패를 일으키는 동적 분석의 형태이다. 애플리케이션에 대한 기능 및 설계 명세를 의도적으로 사용하며 필요에 따라 범위를 넓혀서 테스트한다.
- 공격표면 확인: 검증 대상은 통상 요구사항과 설계단계에서 제작한 기능 및 설계 사양과 차이가 난다. 코드를 완료하면 설계나 구현의 변화로 인해 발생하는 새로운 공격 벡터를 반영할 수 있도록 위협 모델과 공격표면을 재검토해야 한다. 이미 정의하였던 버그 바와 품질 기준을 토대로 식별한 보안 버그를 모두 검토한다.

2.2.6. 배포 단계

제품 배포를 준비하는 동안 사고 대응 계획을 수립하고 최종 보안 검토를 수행하고 소프트웨어 서비스를 위한 모든 데이터를 보관하여야 한다.

- 사고 대응 계획: 배포 당시에 알려진 취약점이 없는 프로그램조차도 새로운 위협에 대해 종속될 수 있다, 최초 접점, 365일 24시간 통화 가능한 전화, 모든 코드에 대한 보안 서비스 계획을 포함한다.
- 최종 보안 검토: 소프트웨어 배포에 앞서 수행하는 모든 보안 활동에 대한 검사이다. 보안 책임자가 개발부서와 보안 및 개인정보보호 부서와 협력하여 수행한다. 최종 보안 검토는 빠진 보안 활동을 수행하는 기회가 아니다. 대개 위협모델, 예외 요청, 도구 출력과 정해진 품질 게이트 및 버그 바에 대한 수행을 검사한다. 최종 보안 검토 결과는 통과, 예외 통과, 미통과 등과 같이 구분한다.

2.3. 선택 보안 활동

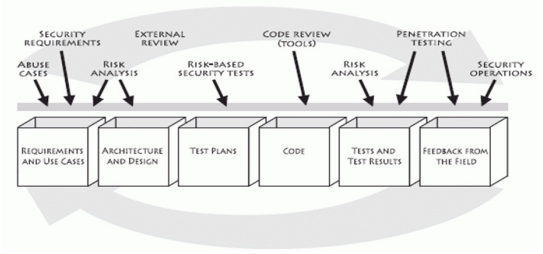
소프트웨어를 매우 심각한 상황에서 사용하는 경우 구성요소에 대한 심층 보안 분석이 필요하다. 주로 수동

코드 검토, 침투테스트, 유사 애플리케이션에 대한 취약점 분석을 수행한다.[3]

- 수동 코드 검토: 자동화된 분석 도구가 완벽하지 않기 때문에 이를 보완하기 위해서 수동 검토를 수행한다. 애플리케이션 보안부서나 보안 책임자 중에서 고도로 숙련된 인력을 활용한다. 개인식별정보 등의 민감정보를 포함하거나 암호학적 기능을 구현하는 구성 요소 등에 대해 주로 수행한다.
- 침투테스트: 침투테스트는 해커와 같은 보안 전문가가 수행하는 화이트박스 테스트이다. 코딩 에러, 시스템 구성 오류, 등 약점을 찾아내고 자동화 도구 및 수동 코드 검토를 병행하여 수행할 수 있다.
- 유사 애플리케이션에 대한 취약성 분석: 유사한 애플리케이션에서 발생하는 설계나 구현 이슈를 통해 취약점 분석을 수행한다.

Ⅲ. SEVEN Touchpoint

Seven Touchpoint는 실무적으로 검증된 소프트웨어 개발 보안 프로세스로서 보안 강화 기법 (build security in)을 지원하는 중요한 기법의 하나이다. [1] 이 프로세스는 SDLC의 각 단계에서 보안 기능과 관련한 모든 활동을 별도로 구분하여 관리한다면 소프트웨어의 보안성을 강화할 수 있다는 통찰에서 출발하였다. 중점적으로 관리해야 할 보안강화 활동으로 7개의 터치포인트를 그림 3과 같이 정의하였다 .



〔그림 3〕 Seven Touchpoint

3.1. 터치포인트 활동

일곱 가지의 터치포인트는 기업 환경에서 가장 효과적으로 적용할 수 있도록 경량화된 형태이다. 실제 적용 시 효과성과 중요도에 따라 터치포인트의 보안활동에 순서를 부여하였다. 순서는 적용 사례에 따라 달라질 수 있다. 그중에서 코드 검토와 아키텍처 위험 분석 활동은 가장 중요한 활동으로 간주하고 있다.

① 코드 검토: 코드 레벨에서 수행하며 소프트웨어 버그 발견에 초점이 있다. 자동화된 정적 분석 도구를 사용하여 보안 취약점을 발견하기 위함이다. 코드 검토를 통해 소프트웨어 버그를 식별하고 수정할 수 있으나 설계 결함을 찾아내기는 어렵다.

② 아키텍처 위험분석: 설계 및 아키텍처에서 일관되고 통합된 보안성을 유지하기 위해서 데이터 보호 오류, 인증이나 접근통제 오류 등과 같은 위험을 찾아내는 활동이다. 가정에 대한 명확한 문서화를 기반으로 공격 가능성을 식별하고 아키텍처 결함을 찾아 우선순위별로 위험 완화를 한다. 아키텍처 위험 분석을 소홀히 하는 것은 소프트웨어 생명주기 내 비용 증가를 초래한다. 위험에 대해서는 지속해서 추적 관찰하고 위험관리를 한다.

③ 침투테스트: 아키텍처 위험 분석에 따른 테스트 활동은 특히 중요하며 실제 시스템 환경에서 대상 소프트웨어에 대한 이해를 돕는 활동이다. 그러나 소프트웨어 아키텍처를 고려하지 않고 수행하는 테스트를 통해 모든 위험을 발견하기는 어렵다. 침투테스트는 화이트 햇 방식과 블랙 햇 방식을 혼용하되 공격자 관점에서 보안 기능과 사양에 대한 결함을 철저히 찾아야 한다. 그러나 시간과 자원의 제약으로 인해 표면적인 보안 문제만을 찾아내고 개선함으로써 보안을 증거 하는 요식행위로 전락할 우려가 있다. 침투테스트의 주요 성공 인자는 해커의 솜씨, 지식, 경험이다. 침투테스트는 최종 운영 환경에서 수행하는 것이 가장 효과적이다.

④ 위험 기반 보안테스트: 개발을 완료한 후 운영환경에서 외부에서 내부로 수행하는 침투테스트와 달리 위험 기반 보안테스트는 시스템 통합 전 구성요소 또는 단위 레벨에서 수행한다. 보안테스트는 시스템 아키텍처에 대한 이해를 바탕으로 공격자 관점에서 위험 기반 접근방식으로 수행한다. 오남용 사례나 악용사례를 기반으로 보안 요구사항 명세에 따라 공격자 관점에서 아키텍처 기반 위험 분석을 수행하고 식별한 위험에 따라 공격이 가능한 영역의 코드에 대해 집중적으로 테스트를 수행한다. 이러한 접근방식은 고전적인 블랙박스 테스트보다 더 높은 수준의

정보보증을 제공한다.

⑤ 악용사례: 요구사항과 유즈케이스를 통해 악용사례를 생성하며 소프트웨어에 대한 규범적 사고를 벗어나 공격자의 사고를 빠르게 흉내 낼 수 있도록 한다. 가장 간단한 방법은 브레인스토밍 기법이며 이때 참여자들의 경험과 역량이 중요하다. 다른 방법은 반요구사항(anti-requirement)과 공격모델을 활용하는 것이다. 반 요구사항이란 소프트웨어에서 절대로 수행하기를 바라지 않는 요구사항이며 실제 요구사항과 밀접한 연관이 있다. 통상 보안 분석가가 요구사항 분석가와 같이 요구사항의 실패를 일으키는 공격을 식별하고 명세화하기 위해 요구사항과 유즈케이스를 분석하는 과정을 통해 생성한다. 반 요구사항은 어떻게 시스템을 악용할 것인지에 대한 통찰을 제공한다. 공격모델이란 기존에 알려진 공격 유형을 구체적으로 적용하는 것이다. 요구사항과 위협을 명세하여 각각의 공격방법을 적용 가능할지 확인해본다. MS의 STRIDE 모델이나 공격 패턴이 유용하다. 시스템과 관련한 공격패턴을 선택하여 이를 포함한 악용사례에 누가 시스템에 접근할 수 있는지를 확인해야 한다.

⑥ 보안 요구사항: 보안 요구사항은 암호의 사용과 같은 명확한 기능적 보안 요구사항과 악용사례나 공격패턴과 같이 부가적인 특징을 포함한다. 보안 요구사항을 식별하고 유지하기 위해 다양한 노력을 수반한다.

⑦ 보안 운영: 보안 운영 시에 네트워크 보안 전문가의 참여를 독려하여 개발부서에서 간과하기 쉬운 네트워크 보안 운영의 경험을 보완하는 것이 좋다. 설계와 구현이 완벽해도 공격은 늘 발생할 수 있다. 현장에서 얻는 지식과 경험을 피드백하여 소프트웨어 개발에 반영하도록 한다.

3.2. 개발단계별 보안 활동

Seven Touchpoint는 개발 과정을 6 단계로 구분하며 단계별 보안 활동은 아래와 같이 구성한다.

- 요구사항과 유즈케이스 단계: 악용사례, 보안 요구사항 정의 및 명세, 위험분석을 수행한다. 즉, 악용사례와 위험분석을 통해 보안 요구사항에 대한 정의와 명세를 하고 악용사례에 대한 정의 및 유즈케이스 예를 작성한다.
- 아키텍처와 설계 단계: 다음과 같이 위험요소를 분석한다.
  - 공격저항 분석: 공격 패턴, 취약성, 이미 알려진 공격정보를 사용하여 아키텍처의 위험요소를 설계한다.
  - 모호성 분석: 새로운 위험을 발견하기 위한 프로세스이며, 최소한 두 번 이상 분석을 수행한다. 부서 내 구성원 각자가 분석활동을 수행한 후 개별적인 이해를 통합하는 형태로 분석한다.
  - 약점 분석: 외부 소프트웨어 종속성에 미치는 영향을 이해하기 위한 프로세스다. 소프트웨어는 일반적으로 복잡한 미들웨어 구조를 활용하며 외부 라이브러리, 공통 라이브러리 등을 포함한다.
- 테스트 계획 단계: 공격패턴, 위험분석 결과, 악용사례를 통해 위험기반 보안테스트를 수행한다.
- 코드 단계: 구현 오류, 특히 소스코드 취약성을 발견하기 위해 정적분석을 통한 코드 검토를 수행한다.
- 테스트와 테스트 결과 단계: 위험분석과 침투테스트를 수행한다.
- 현장 피드백 단계: 침투테스트와 보안운영을 통해 얻은 공격에 대한 지식과 경험을 개발자에게 피드백한다

Ⅳ. A사 소프트웨어 개발 보안 생명주기

국내 대기업에서 적용한 개발 보안 활동 사례로서 기업 내 정보시스템을 구축하는 프로젝트에서 적용하였던 개발단계 보안 강화 프로세스 (P-SDLC) 를 소개한다. [5]

4.1. P-SDLC

A사는 기업 정보시스템의 개편을 위해 효과적인 보안 관리가 필요하게 되어 기존 프로세스에 내재화하여 적용하고 있던 시큐어 SDLC에 대한 평가를 수행하였다. 기존 프로세스에서는 사전 보안 검토, 시큐어 코딩, 모의해킹을 중점적으로 수행하였다. 사전 보안 검토 활동에는 프로젝트 계획 수립 시점에 보안 아키텍처가 보안 아키텍처 검토 수준에서 참여하였다. 이후 구현단계에서 개발부서가 자체적으로 시큐어 코딩을 수행하면 테스트 단계에서 테스트 전문가가 시큐어 코딩 결과를

확인하였으며, 모의해킹은 통합 테스트 단계에서 화이트 해커가 수행하였다. 개발 단계별로 나름대로 보안 활동을 수행하고 있으나 역할이 다른 참여자가 상호 연관이 없이 독립적으로 참여하는 형태로서 보안 영역에 대한 변경이 발생했을 때 변경을 추적하고 관리하기가 쉽지 않았다. 또한, 프로젝트 완료 시점에 수행한 모의해킹을 통해 중대한 설계상의 결함이 발견될 수가 있다. 이 경우에는 비용, 일정 등의 제약으로 인해 발견한 결함에 대해 즉각적인 개선이 어려우므로 보안 약점을 내포한 소프트웨어를 우선 배포한 후 운영 상태에서 보안 패치를 수행하게 된다. 이로 인해 프로젝트 종료 후까지 개발 인력을 유지해야 하거나 보안 위험을 지속해서 수용해야 하는 상황을 초래하게 된다. 이에 개선된 프로세스에서는 모든 개발 단계에서 보안 영역의 변화를 추적 관리할 수 있도록 사전 기술 검토에 이어 설계단계 보안 검토를 중점적으로 추진하도록 하고 각 단계 보안 강화 활동의 결과를 종합하여 최종 보안 검토를 수행하도록 하였다. (그림 4)

단계	계획수립/분석	설계	구현	테스트
보안 활동	-보안검토 -보안교육 -보안 요구사항, 위협요인 식별	- 보안통제 수립 -설계보안 자가점검 -보안부서 확인점검	- 시큐어코딩 -소스코드 보안약점 진단(도구) -진단결과 3 자 확인	- 실행코드 보안취약점 진단 (모의해킹) -최종 보안검토

〔그림 4〕 P-SDLC

4.2. 개발 단계별 보안 활동

4.2.1. 계획 수립 / 분석 단계

계획수립 단계에서 애플리케이션 개발 범위에 대한 사전 보안 검토를 수행하고 개발 부서에 대해 보안 교육을 수행한다.

- 사전 보안 검토: 프로젝트 추진계획과 기업 정보시스템 특징을 반영한 아키텍처 점검사항에 대한 개발팀의 답변을 토대로 사전 보안 검토를 수행한다. 또한, 보안 요구사항 및 위협요인을 식별하여 개인정보보호 등 보안규제 준수, 보안 타당성을 검토한다. 보안 규제를 준수하기 위해 비즈니스 요건을 확인하고 기업 내 아키텍처 구조를 수용할 수 있도록 프로젝트 개발에 필요한 비용을 분석한다. 이 과정에서 비즈니스 요건에 따라 아키텍처를 포함하는 다양한 보안 검토를 수행할 수 있다.
- 보안 교육: 개발 부서 대상 보안부서가 작성한 가이드를 중심으로 교육한다. 주요 교육내용은 개발 보안 프로세스, 보안 설계, 시큐어 코딩, 개인정보보호, 모의해킹을 통한 취약점 발생사례 등이다.

분석 단계에서는 사전 보안 검토 결과를 반영하여 보안 요구사항과 위협 요인을 식별한다.

- 보안 요구사항 수립: 중요 정보, 사용자 유형, 개인정보보호, 인터페이스 등의 위협 요인 인자와 보안 위험 요소를 식별하고 보안 요구사항을 수립한다.

4.2.2. 설계 단계

설계 단계에서는 보안 아키텍처 내에서 보안 요구사항과 위협 요인에 대한 보안 통제를 수립하고 설계 단계 보안 점검을 수행한다.

- 보안 통제 수립: 기업 보안 아키텍처에 기초하여 보안 요구사항에 대한 보안 기능을 설계한다. 또한, 가능한 위협에 대하여 보안 통제 방안을 수립한다. 설계 단계 보안 점검을 수행하고 결과를 반영하는 활동을 반복함으로써 기업 내 아키텍처 범위에 적합한 보안 통제를 수립할 수 있게 된다.
- 설계 보안 점검: 개발부서는 일차적으로 보안부서에서 배포한 설계 보안 점검 항목을 기준으로 자가 점검을 수행하고 보안 부서에 추가 점검을 요청한다. 보안부서는 자가 점검결과를 토대로 보안 통제가 적절한 것인지에 대한 확인 점검을 수행한다. 설계 점검을 통해 보안 설계를 수용하거나 위험 완화를 위한 대안을 제시한다.

4.2.3. 구현 단계

소스코드 구현 단계에는 시큐어 코딩을 적용해야 한다. 시스템의 구성 요소 또는 각각의 단위를 완성하는 단계에서 소스코드 보안 약점을 주기적으로 진단하여 구현을 완성한다.

- 시큐어 코딩: 보안 기능을 포함한 모든 프로그램에 대한 시큐어 코딩을 수행한다. 이미 제공한 시큐어 코딩 가이드를 참조하도록 하며 구현 오류를 지속해서 수

정한다. 특히 보안 기능 구현 시에는 보안 약점을 예방하기 위해 라이브러리로 제공되는 보안 컴포넌트를 적극적으로 활용한다.

- 소스코드 보안 약점 진단: 개발 부서는 자동화된 정적 분석 도구를 사용하여 소스코드 보안 약점을 주기적으로 진단하고 개선하는 활동을 수행하여야 한다. 자동화된 도구에 의해 진단한 보안 약점은 원칙적으로 모두 제거해야 한다. 개발 부서는 구현 완료 단계에서 보안 약점에 대한 최종 진단 결과 보고서를 작성하여 확인부서에 제공해야 한다. 보안 약점 진단 결과에 관한 확인 부서는 최종 진단 결과 보고서를 통해 보안 약점을 제거하였음을 확인한다. 보안 약점을 완전히 제거하지 않았으면 개발부서, 확인부서, 보안부서 간에 보안 약점을 포함하는 것이 대한 합의가 필요하며 이에 따른 위험을 수용하거나 완화할 방안을 제시하여야 한다.

4.2.4. 테스트 단계

테스트 단계는 실행코드에 대해 보안 취약점 진단을 위한 모의해킹을 수행하고 개발 단계별 보안 활동을 종합하여 최종 보안 검토를 수행한다.

- 실행코드 보안 취약점 진단: 보안부서는 코드를 실행한 환경에서 보안 취약점을 진단하여 진단결과를 개발부서에 피드백을 수행한다. 통합 및 운영 테스트 환경에 수행하는 침투테스트는 해당 애플리케이션을 포함한 주변 연동 시스템까지를 포함하여야 하며 주변 연동시스템의 보안 취약점까지 진단하는 효과가 있다.
- 최종 보안 검토: 설계 보안점검을 수행하는 시점에서는 이미 수립한 보안 통제를 적용하였는지를 실제 시험 환경에서 점검한다. 또한, 보안 요구사항의 변경을 추적하고 변경에 따른 추가위험에 대한 분석을 수행하였는지, 분석의 완화 방안은 적절한 것인지 추가로 검토해야 한다. 구현 단계와 테스트 단계에서의 보안 활동의 결과를 종합하여 모든 보안 문제는 모두 개선되었는지를 점검하는 방식으로 최종 보안 검토를 수행한다. 최종 보안 검토 결과는 대상 애플리케이션을 운영 가동을 할 것인지 최종 심사하는 가동 심사에 대한 중요한 기준이 된다.

Ⅴ. 결 론

이상으로 시큐어 SDLC의 대표 사례인 MS SDL,과 Seven Touchpoint를 소개하였고 국내 기업에서 적용하였던 소프트웨어 개발보안 활동 (P-SDLC)을 소개하였다.

MS에서는 MS-SDL 적용 이후 자사의 제품에서 보안 취약점이 현저히 감소하였으며 Forrester 리서치 사는, 기업에서 본문 사례와 같은 시큐어 SDLC를 채택하는 경우, 개발 초기부터 소프트웨어의 취약점을 수정할 수 있으므로 결과적으로 소프트웨어 개발 비용을 매우 감소시킨다고 하였다. 또한, Aberdeen 그룹에 의하면 소프트웨어 개발 보안은 연간 투자 비용보다 4배 이상의 이익을 거둔다고 보고하였다. [4]

현재 국외에서는 소프트웨어 개발 기업을 중심으로 시큐어 SDLC를 활발히 적용하는 추세에 있으며 국내에서는 기업의 적용 사례로서 P-SDLC를 제시하고 있다. [5] P-SDLC 의 적용 경험에 따르면, P-SDLC 적용을 통해 소프트웨어에 대한 보안통제를 효과적인 이행할 수 있으며 개인정보 보호조치 등의 규제 준수가 쉬워진다. 또한, 보안 운영 및 유지보수 비용을 절감시킬 수 있으므로 투자 대비 효과가 크다.

의료, 자동차 등을 포함하는 국내 제조 환경에서도 소프트웨어 비중이 갈수록 커짐에 따라 보안성 확보를 위한 시큐어 코딩 등에 관한 관심이 높아지고 있다. 그러나 보안 취약점에 관한 더욱 적극적이고 효과적인 접근은 구현 단계가 아니라 분석 설계 단계와 같은 개발 초기부터 시큐어 SDLC를 적용하는 것이다. 국내 기업의 경우에는 이미 국내 환경에 맞춰서 적용되고 있는 P-SDLC의 사례를 적극적으로 참조할 수 있다. 지금부터라도 국내 기업이나 공공 분야에서 P-SDLC 등과 같은 시큐어 SDLC를 적극적으로 도입한다면 보안 취약점의 발생을 크게 예방할 수 있게 되어 국내 소프트웨어 보안 수준을 한층 높일 수 있을 것이다. 더 나아가 정보보호 선진수준과의 격차를 빠르게 극복할 수 있을 것으로 생각된다.

참 고 문 헌

[1] G. McGraw, “Software Security: Building Security In”, *Addison Wesley*, 2006.

[2] Microsoft, “Security Development Lifecycle,“ https://www.microsoft.com/en-us/sdl/default.aspx, 2016

[3] Microsoft, "Simplified Implementation of the MS SDL", updated Nov. 4. 2010, http://www.microsoft.com/en-us/download/details.aspx?id=12379

[4] David Ladd, Frank Simorjay, Georgeo Pulikkathara, Jeff Jones, Matt Miller , Steve Lipner, Tim Rains, "Progress reducing software vulnerabilities and developing threat mitigations at Microsoft 2004 - 2010", http://www.microsoft.com/en-us/download/details.aspx?id=14107, 2011

[5] 박난경, 최진영,임종인, “국내 기업의 SW 개발 보안 적용 사례 분석”, *한국소프트웨어공학학술대회*, Jan. 2016

[6] Microsoft, https://msdn.microsoft.com/ en-us/library/windows/desktop/cc307406.aspx

[7] Cadie Thompson, “The 14 scariest hacks of 2015, Techinsider,” http://www.techinsider.io/14-of-the-scariest-hacks-in-2015-8, Dec. 13, 2015

[8] P. Mohan, A. Udaya Shankar, K. JayaSriDevi, “ Quality Flaws: Issues and Challenges in Software Development, ” Computer Engineering and Intelligent Systems, ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online), Vol 3, No.12, 2012, http://www.iiste.org/Journals/index.php/CEIS/article/viewFile/3533/3581

〈저 자 소 개〉

박 난 경 (Ran Kyoung Park)
종신회원

1990년 2월 : KAIST 수학과 학사
1992년 2월 : POSTECH 수학과 석사
2015년 8월 : 고려대학교 정보보호대학원 박사과정 수료
1992년 2월~2015년 3월 : POSCON, POSTECH 통신 및 보안 연구실, POSCO ICT
관심분야 : 정보보호, 융합보안, 기업보안, 개발보안 등

임 종 인 (Jong In Lim)
종신회원

1980년 2월 : 고려대학교 수학과 졸업
1982년 2월 : 고려대학교 수학과 석사
1986년 2월 : 고려대학교 수학과 박사
現 고려대학교 정보보호대학원 명예원장, 고려대학교 사이버국방학과 교수, 개인정보보호위원회 위원, 대검찰청 디지털수사자문위원회 위원장, 국방부 정보화책임관 자문위원, 한국저작권위원회 위원 등
관심분야 : 사이버국방, 정보법학, 디지털포렌식, 개인정보보호, 융합기술보안 등