소스코드 보안성 검사 도구에 관한 연구

방기석*,공진산**, 강인혜***
*한림대학교 기초교육대학
**한림대학교 컴퓨터공학과
***서울시립대학교 기계정보공학과
e-mail:mysaver@hallym.ac.kr

A Study on Security Checker for Source Code

Ki-Seok Bang*,Jin-San Kong**, In-Hye Kang***
*College of General Education, Hallym University
**Dept of Computer Engineering, Hallym University
*** Dept. of Mechanical and Information Engineering, Univ. of SEOUL

요 약

정보시스템의 취약성을 사전에 예방하고 보안을 강화하기 위해 여러 가지 방안이 제시되고 있다. 본 연구에서는 소프트웨어의 소스코드 수준에서의 보안성을 확보하기 위해 소스코드 자체의 보안 위험성을 검사하는 도구를 제안한다. 개발자에 의해 무심코 사용되는 위험 함수들을 검출하고 그 대안을 제시하여 소프트웨어 내부에 잠재되는 보안 위협을 예방하는 도구를 개발하고자 한다.

1. 서론

정보시스템에 오류로 인한 취약성이 악용되면 사회적으로 심각한 문제가 발생하기 때문에 정보시스템의 취약점으로 인한 문제를 방지하기 위해 정보시스템의 보안은 강화되어야한다. 소프트웨어의 보안 취약성은 제품 사용중에 발생하거나 악의적인 사용자에 의해 강제로 발생하는 경우도 있지만, 소프트웨어 자체에 위험성이 내포되어 있는 경우가 많다. 이 경우 개발이 완료되기 이전 혹은 개발과정에서 보다 정확하고 보안을 고려한 개발 및 검사가 필수적이다. 그러나 보통 소프트웨어 개발 후 진행되는 테스팅을 제외하고는 정보시스템(시스템, 소프트웨어, 소스코드) 개발·구축 후 검사과정에서 기능 동작 여부만을 점검하고 취약성에 대한 검사가 미비하다. 따라서 정보시스템 프로그램내의 취약성(오류·결점·결함·보안취약점 등)을 효과적으로 예방하기가 어렵다. 소프트웨어 개발 단계에서 소스코드가 내포할 수 있는 보안 취약성을 검출하여 미리 제거한다면 보다 안전한 소프트웨어가 생산된다는 것은 분명하다.

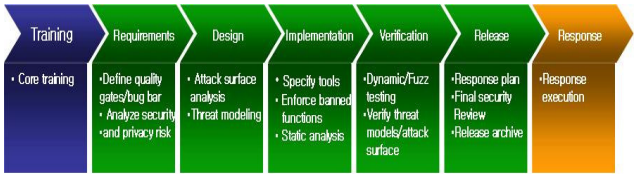
본 연구에서는 완성된 소스코드 뿐 아니라 개발 중인 소스코드의 보안 취약성 검사를 통해 위험한 함수의 사용을 사전에 방지하고 안전한 코딩을 가능하도록 도와주는 검사 도구를 개발한다.

논문은 다음과 같이 구성되어 있다. 2절에서는 안전한 소프트웨어의 개발을 위해 적용되는 기법을 소개하고 3절에서 본 연구의 목표인 소스코드 검사 도구에 대한 소개와 간단한 예제를 보인다. 그리고 결론과 향후 개선방향에 대하 논한다.

2. 관련연구

2.1 Microsoft Secure Development Lifecycle(MS-SDL)

MS에서는 보안을 소프트웨어 개발 프로세스의 주요 요소로 간주하고 있다. 2002년 1월 “Trustworthy Computing(TwC)”이라는 구호 아래 MS 내의 소프트웨어 개발 그룹에서는 코드 내의 보안성 개선을 위한 보안 강화를 강조해 왔다. 보안에 대한 이러한 움직임은 지속적으로 유지되어 보안 개발 생명주기(SDL : Security Development Lifecycle)를 도입하였다[1]. SDL은 개발 생명주기 내에 단계별 주체의 보안 활동을 통하여 소프트웨어 보안을 강화한다. 예를 들면 소프트웨어 설계 단계에서는 위협 모델의 개발, 구현 단계에서는 코드 스캐닝을 위한 정적 분석 도구의 사용, 검증 단계에서는 코드 검증 및 보안성 테스팅 작업이 포함되어 있다. SDL에 따라 개발된 소프트웨어는 출시하기 전에 개발팀과는 별도의 팀에 의하여 최종 보안 검증(Final Security Review)을 거치도록 한다[1,2].

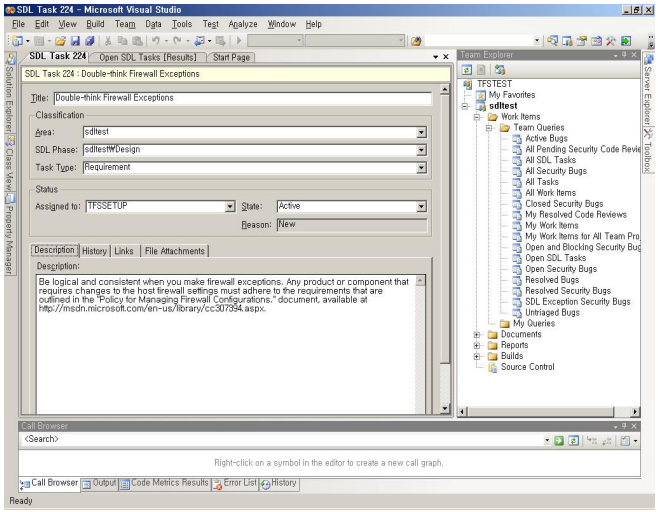


(그림 1) MS Security Development Lifecycle

SDL의 각 단계마다 수행해야 할 보안 활동이 정의되어 있고 각 활동을 엄격하게 수행하도록 강조하고 있다. 특히 설계/구현/검증 단계에서는 자동화된 도구를 제공하여 보다 편리하고 정확한 검사를 수행하도록 한다.

SDL 활동은 Visual Studio Team System(VSTS)[3] 환

경 속에서 통합 관리된다. VSTS는 일반적인 컴파일러의 환경뿐 아니라 SDL Template를 통해 팀 구성원들간의 업무를 관리하고 각 SDL 단계의 활동을 확인할 수 있다. 팀 구성원들은 VSTS를 통해 Team Fountation Server(TFS)에 접속하여 SDL 활동을 수행한다. SDL 각 단계에서 필요한 보안 활동은 자동화 도구를 통해 수행된다. 보안위협 모델링은 STRIDE 기반의 Threat Modeling Tool[4,5]을 이용하여 수행한다. 소스코드의 보안 위험성 검사는 VSTS에 내장되어 있는 CAT(Code Analysis Tool)[6,7]과 FxCop[8]과 같은 전용 검사 도구를 활용한다. 특히 CAT는 VSTS에 내장되어 있는 보안 규칙을 코드와 비교하여 소스코드 내부에 규칙을 위반하는 명령이 사용되는지를 보여준다.



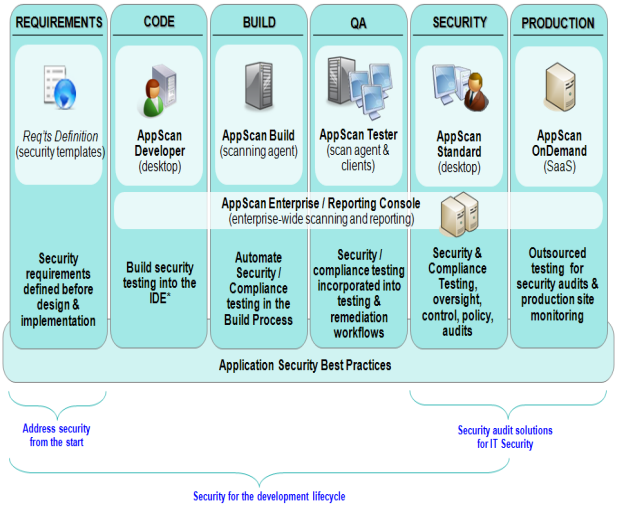
(그림 2) SDL 템플릿을 적용한 Visual Studio 작업화면

VSTS 통합환경을 통해 구성원 각자는 자신이 수행해야 하는 작업에만 집중할 수 있으며 본인이 담당하는 코드에 대한 안전성만을 신경쓰면 된다. 최종적으로 모든 부분은 다시 VSTS에서 통합되어 전체 프로젝트를 완성하게 된다.

이러한 SDL 활동을 통해 MS에서는 자사의 제품에 대해 많은 보안 위협을 감소시켰으며 개발 기간 및 비용의 단축을 성취했다고 발표하고 있다.

2.2 AppScan – IBM

IBM에서는 안전한 소프트웨어를 개발하기 위하여 AppScan[9]을 사용하여 소프트웨어 개발 생명주기(SDLC) 각 단계별로 보안 테스팅을 수행한다. AppScan은 보안 소프트웨어 개발 주기 안에 보안적인 요소를 통합하여 응용 프로그램 개발 과정에서 보안 테스트, 품질 확인 절차의 한 부분으로 보안/인수 테스트, 배포 이전에 보안 감사로서 테스트, 배포된 응용 프로그램의 감시 또는 재감사를 위해 사용된다.



(그림 3) IBM 웹 애플리케이션 보안과 AppScan

AppScan은 Cross-Site Scripting, Buffer Overflow와 같은 웹 애플리케이션의 취약성을 진단해 주는 도구이다. 이 도구는 Web 2.0을 지원하기 때문에 JavaScript, Adobe Flash AJAX 기반의 각종 애플리케이션의 취약성도 진단이 가능하다. 이 도구는 AppScan eXtension Framework를 통해 사용자 커뮤니티에서 개방형 소스의 애드온 기능을 구축하며 자료를 공유할 수 있다. 검사 결과 탐지되지 않은 이슈들을 해결하는데 필요한 수정 권고사항을 제시하는 Results Expert 마법사를 제공하여 개발자 및 보안 담당자가 애플리케이션의 보안 취약성에 대비할 수 있도록 한다.

3. 코드 보안성 검사 도구 개발

3.1 보안성 검사 도구 개요

프로그램 개발자들에게 있어서 코드의 보안성을 확보한는 것은 매우 중요하다. 완성된 코드에 대한 기능성 검사는 다양한 방법을 통해 검사가 가능하다. 또한 여러 가지 보안 기법을 이용해 보안 코드를 작성하는 방법도 사용하고 있다. 하지만 이러한 보안성을 내포하는 코드는 보안에 대해 숙달된 프로그래머들이 아니라면 쉽게 생각하기 어렵다. 특히 눈으로 쉽게 발견하기 어려운 코드 자체의 보안 위협 요소를 생각하여 구현하는 것은 더욱 어려운 작업이다. 개발자는 보안 요소보다는 기능적인 구현을 우선시하기 때문에 코드의 잠재 위협을 생각하기 어렵다. 이렇게 개발되는 코드에는 많은 위협을 내포하는 함수를 사용하기 쉬우며 보안상 문제가 발생하기 전에는 위협에 대처하기 어렵다. 따라서 보안 위협을 내포하는 함수를 사용한 코드에 대해 사전에 경고하고 검출하는 과정이 필수적이다.

3.2 보안성 검사 도구 특징

현재 개발중인 보안 검사 도구는 C/C++ 언어 소스 코드

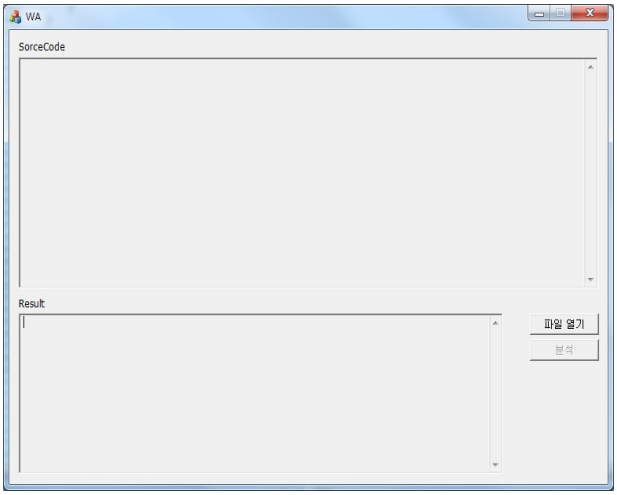
에서 발생할 수 있는 경고(Warning)에 대한 정보를 개발자에게 제공한다. 이 도구를 통해 보고되는 일반적인 코드 경고에는 버퍼 오버런, 초기화되지 않은 메모리, null 포인터 역참조, 메모리 및 누수 등이 있을 수 있다.

개발자가 작성한 코드를 입력받고 도구 내에 정의되어 있는 금칙 명령문의 존재 여부를 판단한다. 판단을 위해 코드 전체에 대해 파싱을 수행하여 금칙 명령문들과의 동일성을 검사하여 금칙 명령문으로 정의된 사항을 발견할 시 이를 개발자에게 알려준다. 또한 각 명령문에 의해 발생할 수 있는 보안 위협사항을 개발자에게 알려줌으로써 코드에 잠재할 수 있는 보안 위협을 사전에 예방할 수 있다.

검사 도구는 완료 되지 않은 코드의 입력이 가능하다. 이는 소스코드에 대한 수행을 위해 컴파일 하는 작업이 아니기 때문이다. 따라서 규모가 큰 코드를 검사하는 시간을 줄일 수 있고, 팀에 의해 개발중인 부분 코드에 대해서도 검사가 가능하다. 또한 C/C++ 컴파일러가 설치되어 있지 않은 환경에서도 동작하기 때문에 프로젝트 매니저나 클라이언트들도 편리하게 사용이 가능하다.

이 검사도구는 보안 규칙의 설정이 용이하다. 사전에 정의된 금지 명령어를 개발자에 의해 추가/삭제할 수 있다. 보안 위협은 매우 다양하며 요소에 따라 위험도가 다르다. 현재 알려진 보안 위협을 내포하는 함수들 뿐 아니라, 새로 발견되는 보안 위협에 노출되는 함수에 대해서도 대처가 필요하다. 반대로 발생 빈도가 매우 낮거나 그 영향이 거의 없는 함수에 대해서는 금지 항목에서 삭제할 수 있다. 이러한 기능을 통해 보안 강도를 조절하도록 한다.

보안상 위험한 함수가 발견되면 개발자에게 차선책을 알려줘야 한다. 일반적으로 같은 기능을 하는 함수가 여럿 존재하지만 쉽게 생각하지 못하는 경우가 많다. 이러한 보고 기능을 통해 보다 안전한 함수의 사용을 가능하도록 한다.

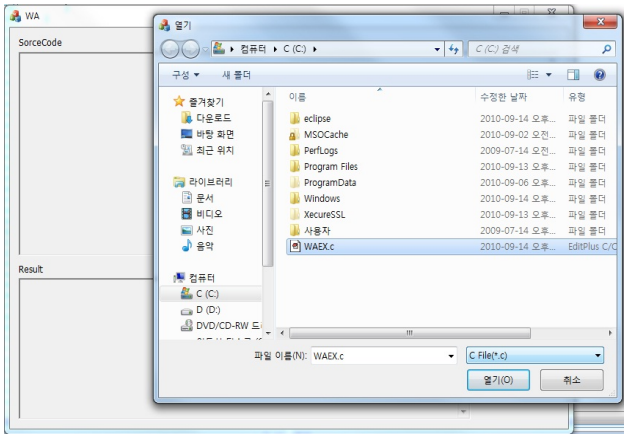


(그림 4) 개발 중인 코드 보안 규칙 검사 도구

3.3 보안 검사 예

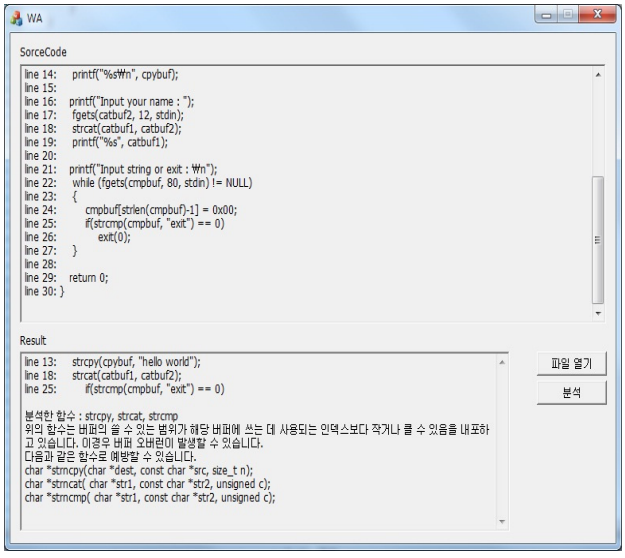
(그림 3)은 개발중인 도구의 초기 모델이다. 현재 소스코드를 입력하고 분석하면 위협을 내포한 함수와 가능한 위협에 대한 안내, 그리고 대체할 수 있는 함수들을 출력해 준다.

개발중인 소스코드 파일을 선택하면 화면에 각 라인별로 코드가 표시된다. 그리고 분석 버튼을 누르면 현재 작성중인 코드에 사용된 위험한 함수와 보안 위협이 보고된다.



(그림 5) 개발 중인 코드 보안 규칙 검사 도구

예제 코드의 경우 scrcpy(), strcat(), strcmp() 함수를 사용했으며 이 함수들은 버퍼 오버런 공격에 취약한 함수로 알려져 있다. 이를 개선하기 위해 strncpy(), strncat(), strncmp() 함수를 사용할 것을 보고한다.



(그림 6) 보안 검사 결과

4. 결론 및 향후 연구

본 연구에서는 보안 위협을 내포하고 있는 소스코드를 검사하여 위험성을 제거할 수 있는 검사도구를 개발하는 것을 목표로 하고 있다. 많은 보안 위협은 외부로부터의 공격 뿐 아니라 소스코드 자체에 내포하고 있다. 기능적으

로는 정상적이지만 함수의 특성상 외부의 침입을 유발하거나 위험한 동작을 보일 가능성이 있기 때문에 보다 안전한 함수의 사용이 필수적이다.

이러한 보안 위협을 내포한 함수를 개발 단계에서 검출하고 개발자에게 알려주기 위한 도구를 개발하고 있다. 이 도구를 사용하면 완성되지 않은 코드에 대해서도 검사가 가능할 뿐 아니라 특정한 컴파일러가 없어도 코드의 보안성 검사가 가능하다.

또한 보안 코딩에 대한 교육에도 활용 가능할 것이다. 프로그래밍을 학습하고 있는 단계에서 본인의 코드에 내포하는 보안 취약성을 제거하는 연습을 한다면 더욱 고품질의 제품을 생산해 낼 수 있을 것이다. 이를 위해 보안 코딩 교육용으로 활용한다면 더욱 좋은 효과가 있을 것이다.

현재는 초기 형태로 다양한 보안 위협 및 위험한 함수에 대한 검사는 수행하지 않는다. 향후 발생 가능한 보안 위협에 대한 추가와 위험한 함수들에 대한 분류를 통해 보다 많은 취약성 검사가 가능하도록 할 계획이다. 또한 실제 코드를 직접 수정할 수 있도록 도구를 개선할 계획이다.

참고문헌

[1] www.microsoft.com/sdl

[2] Michael Howaard and Steve Lipner, The Security Development Lifecycle. . Microsoft Press, 2006

[3] Richard Hundhausen, VSTS Working with Visual Studio 2005 Team System, Microsoft Press

[4] Frank Swiderski, Threat Modeling. Microsoft Press, 2004

[5] MSDN Magazine, "Security Briefs: 위협 모델링 프로세스의 재발견", http://msdn.microsoft.com/ko-kr/magazine/cc700352.aspx

[6] CAT.NET, http://msdn.microsoft.com/en-us/security/sdl-tools-download.aspx

[7] MSDN Magazine, "코드분석을 사용한 C/C++ 코드 품질 분석", http://msdn.microsoft.com/ko-kr/library/ms182025.aspx

[8] http://msdn.microsoft.com/ko-kr/library/bb429476(en-us,VS.80).aspx

[9] "An Overview of the Rational Unified Process", IBM