https://doi.org/10.5392/JKCA.2018.18.08.459

안전한 무기체계 소프트웨어를 위한 취약점 분석 기법에 관한 연구

A Study on Vulnerability Analysis Techniques for Secure Weapon System Software

김종복, 조인준
배재대학교 사이버보안학과

Jong-Bok Kim(chongpok@naver.com), In-June Jo(injune@pcu.ac.kr)

요약

무기체계 관련 어플리케이션과 국방 관련 기관에서 활용하는 정보시스템이 사이버 공격을 받을 경우 국가의 안보가 위험해지는 결과를 초래한다. 이러한 위험을 줄이기 위해 개발 단계에서부터 시큐어 코딩을 적용하거나, 발견된 취약점들을 체계적으로 관리하기 위한 노력이 지속적으로 행해지고 있다. 또한 다양한 분석 도구를 이용하여 취약점을 분석, 탐지하고 개발 단계에서 취약점을 제거하거나, 개발된 어플리케이션에서 취약점을 제거하기 위해 노력하고 있다. 그러나 취약점 분석 도구들은 미탐지, 오탐지, 과탐지를 발생시켜 정확한 취약점 탐지를 어렵게 한다. 본 논문에서는 이러한 문제점 해결방안으로 분석 대상이 되는 어플리케이션의 위험도를 평가하고 이를 기반으로 안전한 어플리케이션을 개발 및 관리할 수 있는 취약점 탐지기법을 새롭게 제안하였다.

■ 중심어 : | 취약점 분석 | 동적분석 | 정적분석 | 시큐어 코딩 |

Abstract

Cyberattacks on information systems used by applications related to weapon system and organizations associated with national defense put national security at risk. To reduce these threats, continuous efforts such as applying secure coding from the development stage or managing detected vulnerabilities systematically are being made. It also analyzes and detects vulnerabilities by using various analysis tools, eliminates at the development stage, and removes from developed applications. However, vulnerability analysis tools cause problems such as undetected, false positives, and overdetected, making accurate vulnerability detection difficult. In this paper, we propose a new vulnerability detection method to solve these problems, which can assess the risk of certain applications and create and manage secured application with this data.

■ keyword : | Vulnerability Analysis | Static Analysis | Dynamic Analysis | Secure Coding |

I. 서 론

소프트웨어의 품질을 높이는 방안의 일환으로 소프트웨어 보안성 제고 문제가 제기되고 있다. 이를 위해 정보시스템 운영단계에서 발현 가능성이 높은 보안 취약점을 소프트웨어 개발 단계에서 부터 사전 제거할 필요가 있다. 이러한 활동을 지원하기 위해 행정안전부와 한국인터넷진흥원은 전자정부 SW 개발·운영자를 위한

* 본 논문은 2018학년도 배재대학교 교내학술연구비 지원에 의하여 수행된 것임
접수일자 : 2018년 07월 16일
심사완료일 : 2018년 08월 16일
수정일자 : 2018년 08월 16일
교신저자 : 조인준, e-mail : injune@pcu.ac.kr

소프트웨어 개발 보안 가이드를 제작·배포하여 지원하고 있다[1].

국방 분야에서는 무기체계 소프트웨어 결함 및 취약점을 사전에 식별하여 제거하기 위해 소프트웨어 통합 및 시험 단계에서 소프트웨어 형상항목(CSCI) 단위로 신뢰성 및 보안성 시험을 실시하고 있다[2].

신뢰성시험 중 취약점 점검과 보안성 시험은 사전에 취약점 제거를 위한 시험이다. 여기에서 취약점 점검은 소프트웨어 소스코드에 CWE 658/659/660[3]에 정의된 약점을 포함하는지 여부를 점검하는 것이다. 그리고 보안성 시험은 행자부 소프트웨어 개발보안 가이드를 기준으로 소프트웨어의 보안 약점을 제거하기 위한 것이다. 상기에서 언급된 CWE 658/659/660[3]은 C, C++, Java로 개발된 소프트웨어에서 발견될 수 있는 결함을 유발할 수 있는 약점 목록으로 정의될 수 있다. 따라서 보안 관련성 및 무기체계 소프트웨어 관련성이 고려되지 않은 목록이다. 결론적으로 행정안전부의 소프트웨어 개발보안 가이드는 전자정부 소프트웨어 즉 웹 어플리케이션을 대상으로 도출한 약점 목록으로써 무기체계 소프트웨어에 대한 관련성은 반영되지 않은 목록이다. 따라서 이를 현재 국방 분야에서 보안 취약점 사전 제거 및 무기체계 소프트웨어 분야에 적용하는 것은 무리가 따른다.

무기체계는 지휘통제통신, 감시정찰, 기동, 함정, 항공, 화력, 방호, 기타 무기체계를 포함한다. 각 무기체계 소프트웨어는 공통 특성을 가지기도 하지만 수행 임무에 따라 서로 다른 운용 방식과 체계 소프트웨어 개발 특성을 가진다. 따라서 무기체계 별로 중요 보안 취약점이 달라질 수 있기 때문에 정확한 보안 취약점 진단을 위해서는 무기체계 소프트웨어별로 중요 보안 취약점 목록을 사전에 선정하고 탐지하는 것이 바람직하다.

보안취약점을 탐지하는 기법으로는 소스코드를 기반으로 하는 정적분석 방법과 실행 시스템을 기반으로 하는 동적분석 방법이 있다. 하지만, 이러한 방법들이 지니는 현재의 한계는 오탐지(誤探知), 미탐지(未探知) 등이 빈번이 발생되는 문제점이다.

본 논문은 기존의 정적분석 기법과 동적분석 기법의 단점들을 보완하는 방안에 관한 내용이다. 이를 위해 소프트웨어 개발자에 의해 수행되는 취약점 분석의 오탐지를 효율적으로 줄일 수 있도록 자동화하기 위한 분석 기법을 새롭게 제안하였다.

II. 관련연구

현재 국방 분야에서 정의하고 있는 무기체계 소프트웨어는 무기체계 및 비무기체계 분야 소프트웨어로 구분하고 있다.

무기체계 분야 소프트웨어로써는 항공무기체계, 감시정찰무기체계, 유도무기체계, 화력·방호무기체계, 기동무기체계, 함정무기체계 분야의 내장형 소프트웨어 분야와 지휘통제체계, 전술훈련모의장비, M&S체계, 통신체계/장비 분야의 정보시스템으로 구분되어 있다.

비무기체계 분야 소프트웨어로써는 자원관리체계, 교육훈련용 장비/물자, 일반군수품(장비 등), 기타일반시설, 기반체계 및 M&S체계(무기제외) 분야의 정보시스템으로 구분되어 있다.

무기체계 소프트웨어와 관련된 국방부 및 방위사업청 규정 및 지침은 [표 1]과 같이 구성되어 있다.

표 1. 무기체계 소프트웨어 관련 제도

구분	규정 / 훈령 / 지침서
국방부	국방정보화업무훈령(국방부 훈령 제1801호, '15.6.11) 국방전력발전업무훈령(국방부 훈령 제 1975호, '17.6.5)
	국방아키텍처 프레임워크 Ver. 1.2 (국방부 지침서, '08.5) – 상호 운용성 확보 및 정보시스템의 효율적 구축, 관리 등을 위하여 운용관점/체계관점/기술관점 에서 정보시스템 아키텍처를 종합적으로 설계한 청사진
	국방CBD 방법론 Ver. 2.0 (국방부 지침서, '10.5) – 재사용성 제고 등을 위해 정보시스템 개발시 독립적으로 실행/배포 가능한 소프트웨어 단위인 컴포넌트 개발 방법론을 제시
방위사업청	방위사업관리규정(방위사업청 훈령 제361호, '17.6)
	무기체계 소프트웨어 개발 및 관리 매뉴얼 (방위사업청 예규 제161호, '16.7)
	무기체계 소프트웨어 개발지원에 관한 규정 (방위사업청 훈령 제327호, '16.10)

무기체계 소프트웨어의 취약점을 제거하기 위해서는 개발 단계에서부터 시큐어 코딩규칙의 적용이 필요하다[4]. 그러나 국내 현실은 무기체계 소프트웨어에 대해서 시큐어 코딩규칙에 대한 적용 의무화가 본격적으로

논의되지 않고 있는 실정이다. 따라서, 무기체계 소프트웨어의 보안성을 보완하고 발전시킬 수 있는 시큐어 코딩규칙을 개발하고, 평가하고, 적용시키는 방안 등이 충분히 논의되지 못하고 있다. 학술적 측면에서 방대한 무기체계 소프트웨어와 국방소프트웨어를 분류하여 각각의 분야별로 개발단계에서 보안을 적용하고자 하는 연구[4]정도가 시도되고 있을 뿐이다. 결론적으로 국내 국방 분야와 무기체계 분야에서의 시큐어 코딩규칙과 개발단계에서 보안을 적용하는 방안에 대한 전반적인 학술적 연구와 실무적 논의는 아직 미흡한 실정이다.

최근에 방위사업청에서는 무기체계 소프트웨어의 신뢰성/보안성 시험 절차를 규정하였다[3]. 이 규정에 따르면, 무기체계 소프트웨어 개발 후 시험단계에서 신뢰성 시험과 보안성 시험을 실시하도록 규정하고 있다. 신뢰성 시험은 정적시험과 동적시험으로 구분하여, 정적시험에서 코딩규칙 검증, 취약점점검, 소스코드 매트릭 등을 확인하도록 하고 있다. 동적시험에서는 코드실행율(statement 실행율, branch 실행율, MC/DC실행률)을 결정하도록 규정한다. 무기체계 소프트웨어 분야에서 주로 사용하는 정적시험 도구로써는 CodeInspector CodeSonar, LDRA Testbed 등이 있으며 동적시험 도구로써는 LDRA TBRun, Controller Tester, DT10 등이 사용되고 있다.

보안성 시험에 대한 권고사항은 국방 분야의 무기체계 소프트웨어 중 전장관리정보체계 분야에 한해서 행정안전부의 소프트웨어 개발 보안가이드를 준수하도록 하고 있다. 이와 같은 신뢰성/보안성 시험에 대한 기본계획(적용대상 CSCI, 평가항목, 사용도구 및 기타사업별 필요사항)은 체계개발실행계획서에서 미리 반영하고, 상세설계단계에서 소프트웨어 검토회의 등을 통해 확정하고 있는 추세이다.

전장관리정보체계 분야 소프트웨어에 대한 보안성 시험은 시험대상 항목(소프트웨어형상항목(CSCI:Computer Software Configuration Item), 소프트웨어구성품(CSC:Computer Software Component), 단위소프트웨어(CSU:Computer Software Unit))을 먼저 확립한 후 해당 시험대상에 대한 시험기준을 설정한 후 보안성 시험을 실시한다. 소프트웨어 보안성 시험 충족 여부는 국가정보원 CC 인증된 도구를 사용하여 행정안전부의 소프트웨어 개발 보안가이드를 준수하는지 여부를 판단하게 된다.

최근 공공기관에서 개발단계에서 보안 적용 의무화 시행 이후, 공공기관 정보시스템 구축 및 금융 전산보안 분야의 경우, 시큐어 코딩과 소프트웨어 개발단계의 보안 쟁점이 활발하게 이루어지고 있다. 또한 소프트웨어 개발과 관련한 코드 리뷰 기준과 기법에 대해 주로 실무적인 접근을 중심으로 급격하게 발전하고 있다.

그러나 이 분야 역시 현재까지 학술적으로 많은 연구들이 이루어지고 있지는 않은 실정이다. 현재 국내에서 사용되고 있는 시큐어 코딩규칙은 외국의 공개 취약점 분석 목록과 시큐어 코딩규칙을 참조하여 시큐어 코딩 목록을 만들었다. 그리고 공개된 취약점 목록의 상위영역의 취약점들과 시큐어 코딩규칙을 매핑하는 형태로 개발되어 사용되고 있다.

국외에서 이루어지는 시큐어 코딩규칙 개발의 경우, 정부기관이 소프트웨어 관련 연구기관에 연구용역을 주어 개발하였다. 용역을 받은 연구기관은 언어별 특성에 따라 발생할 수 있는 취약점을 특성별로 분석하여 평가하고 목록화한다. 그리고 이들 언어별 취약점들을 기존의 취약점 분석 공개 목록과 비교 분석하여 개발언어별 프로그래밍 취약점 목록을 도출하는 방식으로 개발되었다[8]. 이러한, 방식은 기존 개발 언어 표준 연구와 취약점 분석이 주(主)가 된다.

2012년에 행정안전부가 개발단계에 보안 적용 의무화 제도를 시행하면서 제시한 코딩규칙 또한 여러 단계를 거쳐 만들어진 것이다. 즉, 전자정부프레임워크 개발보안에 대한 시큐어 코딩규칙을 기존 개발 언어 표준 취약점 분석과 매핑 개발하는 과정에 연구용역으로 수개월에서 수년이 소요되었다. 그러나 이러한 방식으로 만들어진 시큐어 코딩규칙들은 적용될 시스템에 최적화되어 있지 않고, 개발 언어의 특성을 중심으로 구성된 일반적인 목록의 성격을 지니고 있다. 많은 시큐어 코딩규칙을 가지고 있을수록 보다 효과적이고 안정적인 대응이 가능하겠지만, 소프트웨어 개발 제한성(시간과 인력)이 높아지게 된다. 또한, 소프트웨어 테스트 비용도 함께 증가하게 되므로, 개발 제한성과 테스트 비

용간의 적정선을 찾는 것이 필요하다. 이를 위해 실질적인 적용 우선 순위를 고려한 시큐어 코딩규칙의 선정과 중복된 유사 규칙들의 식별 과정이 필요하게 되었다.

현재 제공되고 있는 정부의 지침의 경우에도, 유사한 규칙의 중복성이 상당수 발견되고 있는 실정이다.

시큐어 코딩규칙의 적용 여부는 정적 검사 도구에 의한 자동화 검사를 통한 코드 리뷰를 통해 검증되는 것이 일반적인 절차이다. 많은 규칙이 중복되어 적용되는 경우 오탐이 증가하여 정적검사 검토 결과를 신뢰하기 어려운 경우가 흔히 발생하고 있다. 또한, 이에 따라 개발된 코드의 수정 과정이 힘들어져 소스 수정 과정에서 발생할 수 있는 추가적인 오류 사항에 대한 재귀시험(Recursive Testing) 단계를 거쳐야 하는 어려움이 산재되어 있다. 그러므로 적용 대상 무기체계에 대한 시큐어 코딩규칙 적용 우선순위가 선별되어 최적화된 보안 취약점을 탐지해 내는 과정이 필요하다.

시큐어 코딩은 무기체계 소프트웨어 획득 과정에서 이루어지는 소프트웨어 통합 및 시험 단계에서, 소프트웨어 개발 언어의 사용에서 정의되지 못한 논리적 오류 및 설계 구현 과정에서 소프트웨어 개발자의 실수로 인해 발생할 수 있는 보안 취약점들을 줄이기 위한 일련의 보안성 향상 활동을 의미한다[3][4].

소프트웨어 보안 약점은 소프트웨어 결함, 오류 등으로 인하여 해킹 등 사이버공격을 유발할 가능성이 있는 잠재적인 보안 취약점을 의미한다[4].

시큐어 코딩규칙은 형태적으로 기존에 소프트웨어 품질이나 신뢰성 향상을 위해 사용되던 코딩규칙 및 코드 리뷰(Code Review)와 유사한 형태를 가지고 있다. 또한, 소프트웨어 품질이나 신뢰성 분야에서 사용되는 권장 코딩규칙은 상당 부분 보안을 위한 코딩규칙에도 포함되고 있다.

시큐어 코딩규칙은 소프트웨어 보안성 향상을 목표로 하고 있으므로 개발 언어별 특성에 따라 발생하는 소프트웨어 취약점 분석을 통해 지속적으로 식별되고 있다. 취약점 판단과 분석 근거로는 OWASP(The Open Web Application Security Project)의 웹 보안 위협 상위 TOP 10 목록, CWE(Common Weaknes Exposure), CVE(Common Vulnerability Exposure), SANS Top 25 Software Errors 등의 보안 취약점 공개 목록들이 주로 활용되고 있다[4]. 그런데, 이런 취약점 분석과정은 그 분석 대상을 일반 비즈니스 업무에서 많이 사용하는 정보체계를 중심으로 진행되고 있는 실정이다.

결론적으로 무기체계 소프트웨어 분야의 경우에는 일반 비즈니스 정보체계 분야와는 달리 정부 개발 보안 가이드 등의 지원이나 공식적인 기준 또는 표준 대비, 무기체계 소프트웨어 개발 및 관리 매뉴얼[3]에 필요한 만큼 만 규정되어 있다. 앞서 살펴본 바와 같이, 일반적인 시큐어 코딩이나 보안 취약점 탐지 방법에 비해, 무기체계 소프트웨어 분야는 관련된 학술연구와 실무적 논의가 현재 그 필요성에 비해 미비한 편이라고 판단된다.

III. 제안 방안

본 논문에서 제안하는 보안 취약점 분석 기법은 기존의 정적분석 기법과 동적분석 기법의 단점들을 보완하고 소프트웨어 개발자들에 의해 수행되던 오탐지 재조정 노력을 자동화하기 위한 취약점 분석 기법이다. 또한 제안 기법은 행정자치부와 한국인터넷진흥원에서 배포하는 소프트웨어 개발 보안가이드 중 SANS 25 Software Error의 상위 10개 항목과 일치하는 항목을 대상으로 한다.

본 논문에서는 위에서 언급한 10개 대상 항목에 대하여 보안 취약점을 분석할 수 있는 방안을 제안한다. [그림 1]은 제안 기법의 개념도이다.

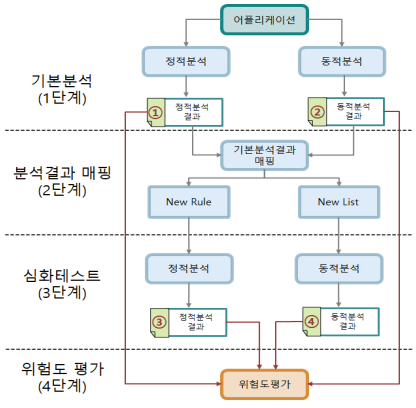

그림 1.제안기법의 개념도

1. 취약점 분석 절차

본 절에서는 제안 기법의 취약점 분석 절차에 대해 설명한다. [그림 2]에서 보는 바와 같이 취약점 분석 절차는 다음과 같다.

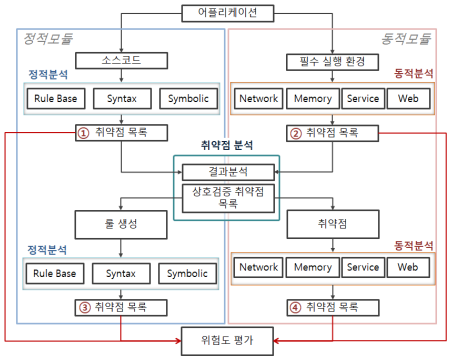

그림 2. 제안 기법 분석 절차

1.1 기본분석 단계

기본분석 단계는 제안하는 기법의 첫 번째 단계로 기존의 정적분석 도구와 동적분석 도구를 이용하여 어플리케이션의 취약점을 분석하는 단계이다. 기본 취약점 분석 단계는 [그림 3]과 같이 정적 모듈과 동적 모듈로 구성되어 있다.

어플리케이션의 소스코드는 정적 모듈의 소스 코드 컬렉션부(Ⓐ:Source Code Collection)로 보내진다. 소스코드 컬렉션부(Ⓐ)는 소스코드에 대한 사전 정보들을 동적 모듈의 어플리케이션 유형 분석부(Ⓓ:Application Type Analysis)로 전달한 후 분석하려고 하는 소스 코드가 어떤 유형의 어플리케이션인지 판별한다.

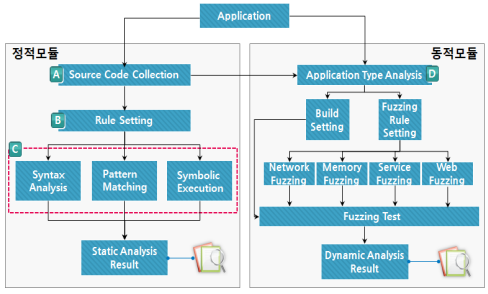

그림 3. 기본 분석 단계

어플리케이션 유형분석부(Ⓓ)에서는 소스 코드 컬렉션부(Ⓐ)로부터 전달받은 정보를 이용하여 해당 어플리케이션의 유형(애플릿, 웹 어플리케이션, 단독 시스템 등)의 유형을 분석한다.

1.2 분석결과 매핑 단계

분석결과 매핑 단계에서는 기본분석 단계의 결과를 분석하여 심화테스트 단계를 위한 관련 정보를 생성한다. 분석결과 매핑에 대한 일련의 과정은 [그림 4]에서 보여주고 있다.

분석결과 매핑 단계에서는 기본분석 단계를 거쳐 도출된 결과를 토대로 소스코드상의 동일한 위치에서 동일한 취약점이 탐지되었는지 여부를 분석한다. 취약점의 특성상 정적분석 도구와 동적분석 도구가 탐지할 수 없는 취약점과 탐지할 수 있는 취약점이 존재한다. 결과 구분부(Ⓐ:Result Categorization)에서는 전자정부 소프트웨어 개발 보안 가이드와 SANS 25의 상위 10개 항목 간 매칭 항목을 대상으로 공통 탐지 취약점 목록과 별건 탐지 취약점 목록을 작성한다. 정적모듈과 동적 모듈에서 공통 탐지 가능 취약점 목록은 매핑가능(Mapping Available) 모듈을 거치게 되고, 공통으로 탐지될 수 없는 목록(별건 탐지 취약점 목록)은 매핑 불가 모듈을 거치게 된다.

매핑 불가 모듈로 분류된 취약점 항목들은 기본분석 단계의 어느 모듈 결과인지를 확인한 후 관련 정보와 함께 심화테스트 단계를 거치게 된다.

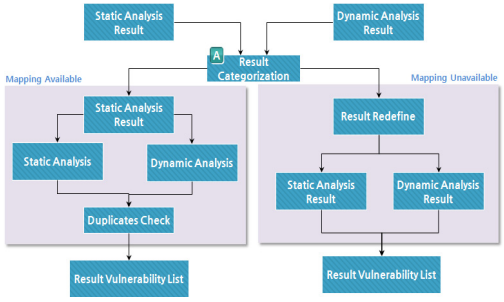

그림 4. 분석결과 매핑 단계

매핑 가능 모듈로 분류된 취약점 항목들은 정적분석

및 동적분석 과정에서 동일한 원인으로 발생하게 된 취약점인지를 식별되어진다. 동일한 원인으로 탐지된 취약점 항목들은 심화테스트 단계를 거치지 않고 위험도 평가 단계로 전달한다. 해당 취약점 항목들이 동일한 원인으로 탐지되지 않은 취약점인 경우, 해당 취약점 항목들은 오탐지의 확률이 높다. 이 취약점 항목들은 취약점 탐지의 정확도를 높이기 심화테스트 단계를 거치게 된다.

1.3 심화테스트 단계

심화테스트 단계에서는 분석결과 매핑 단계에서 전달된 동적분석 모듈 결과의 정확도를 높이기 위해 정적분석 모듈로 심화테스트를 수행한다. [그림 5]는 이 과정을 도식화한 그림이다.

심화테스트 단계의 정적분석 단계로 전달되는 동적 취약점 리스트(Ⓐ:Dynamic Analysis Result List)는 기본분석 단계(그림[3])의 동적분석 과정을 마친 취약점 항목들이다. 이 항목들은 기본분석 단계의 정적분석 과정을 거친 취약점들과 동일한 원인으로 판단되지 않는 취약점 항목들이다.

규칙 타입 분석기(Ⓑ:Rule Type Analysis))는 동적 취약점 리스트(Ⓐ)를 분석하여 해당하는 규칙 타입을 분류한다.

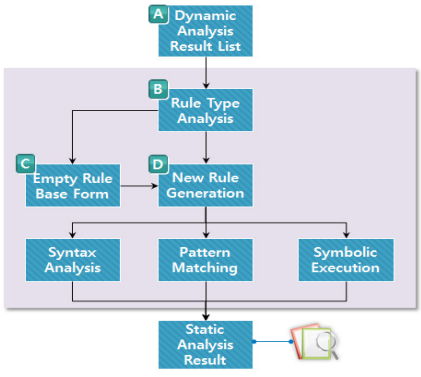

그림 5. 정적분석 모듈 심화테스트 절차

신규 규칙 생성기(Ⓓ:New Rule Generation)는, 본 논문에서 취급하는 10개 항목과는 별개의 빈(空) 규칙 형식(Ⓒ:Empty Rule Base Form)을 신규로 구성한 후, 전달되어 온 취약점 항목들을 재확인할 수 있는 맞춤형 규칙들을 생성한다.

신규 규칙 생성기(Ⓓ:New Rule Generation)에서 생성된 규칙들은 [그림 5]와 같이 정적분석 모듈 심화테스트 과정을 거친다.

정적분석 모듈 심화테스트 과정은 동적분석 도구에서 탐지한 취약점이 소스 코드에 실제로 존재하는지 확인하는 과정이다. 이 과정을 통하여 소스 코드 상에서 패턴과 구문 매칭 등으로 탐지하지 못했던 부분을 재확인하고, 오탐지 및 과탐지를 제거하여 취약점 탐지 정확도를 높일 수 있다.

반면에 분석결과 매핑 단계에서 전달된 정적분석 모듈 결과에 대한 정확도를 높이기 위해 동적분석 모듈 심화테스트를 수행한다.

기본분석 단계에서 정적분석 모듈로 탐지한 취약점 항목들 중 동적분석 모듈의 탐지 결과와 매핑되지 않는 취약점 항목이 발생할 수 있다. 또한 매핑은 되었지만 정적분석 및 동적분석 간 동일한 원인으로 탐지가 되지 않는 취약점 항목도 발생 할 수 있다. 이러한 취약점 항목들은 정적분석 결과 취약점이 탐지되었지만, 오탐지로 분류될 가능성이 있기 때문에 동적분석 모듈로 재확인하여 탐지 정확도를 높이는 과정이 필요하다. [그림 6]은 이러한 절차를 도식화한 것이다.

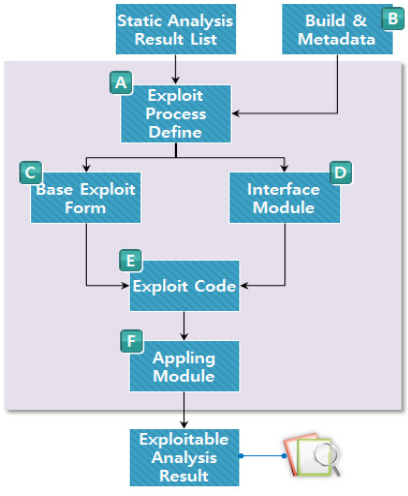

그림 6. 동적 분석 모듈 심화테스트 절차

[그림 4]의 분석 결과 매핑단계에서 전달된 취약점 항목들을 분석하여 각 취약점 항목마다 Exploit 코드를 생성하기 위한 프로세스를 정의(Ⓐ:Exploit Process Define)한다.

동적분석 모듈은 어플리케이션이 실행되고 있는 상태에서 취약점 분석이 이루어지기 때문에 취약점이 발견된 코드가 실행될 수 있는 환경을 만드는 과정이 필요하다. 이러한 이유로 [그림 6]의 Exploit 프로세스 정의(Ⓐ:Exploit Process Define) 과정을 거친다. 즉, 어플리케이션에 대한 빌드(Build) 정보(Ⓑ:Build&Metadata)와 어플리케이션을 실행한 후 취약점 코드 부분까지 도달할 수 있는 정보를 Exploit 프로세스 정의(Ⓐ) 과정에서 제공한다.

취약점 항목이 존재하는 소스가 실행되는 동안 삽입될 정보를 기본 Exploit 양식(Ⓒ:Base Exploit Form) 에 삽입하여 Exploit 양식을 완성한다. 동시에 인터페이스 모듈(Ⓓ:Interface Module) 에서 Exploit 양식이 실제 실행될 수 있는 스크립트 등을 생성하여, Exploit 코드(Ⓔ:Exploit Code)를 생성한다. 응용모듈(Ⓕ:Appling Module) 에서는 작성된 Exploit 코드를 해당 어플리케이션에 적용하여 취약점 탐지가 정확히 수행되었는지 확인한다.

1.4 위험도 평가 단계

위험도 평가 단계는 기본분석 단계, 분석결과 매핑단계, 심화테스트 단계에서 도출된 결과를 이용하여 해당 어플리케이션의 최종 위험도를 평가하는 단계이다.

최종 위험도는 각 단계에서 전달된 취약점 탐지 리스트들 자체 위험도, 해당 어플리케이션에서 탐지된 취약점 항목들의 정확도, 그리고 탐지된 취약점 항목들이 해당 어플리케이션에서 탐지되는 빈도를 이용하여 어플리케이션 위험도를 최종 평가한다.

IV. 고찰 및 검증

본 논문에서 제안하는 취약점 분석 기법으로 기존의 정적분석도구와 동적분석 도구의 취약점 탐지 정확도를 비교 분석하였다. 비교 대상 도구로써 동적분석 모듈은 SPIKE, 정적분석 모듈은 PMD를 활용하였다.

PMD는 JAVA 프로그래밍 언어로 구현된 소스코드를 대상으로 SW를 실행하지 않고 해당 소스코드를 분석하여 소스코드에 존재하는 미사용 변수, try-catch 구분에서 비어 있는 catch 블록 등 SW 결함을 탐지해 주는 정적분석 기반의 공개용 진단도구이다. PMD는 JAVA 및 XPath를 기반으로 진단규칙을 구현할 수 있으며, 270여개 정도의 진단규칙을 보유하고 있다. 현재 전자정부표준프레임워크에서 SW 구현 시 소스코드 분석을 위해 JAVA 클래스 파일 대상 보안약점 진단 도구로 PMD 사용을 권고하고 있다[5][6].

SPIKE는 소스코드를 분석하여 테스트케이스를 생성하거나 취약성관련 부분의 프로그램을 실행시켜 그 행위를 보며 취약성을 분석하는 동적분석 도구이다. SPIKE는 네크워크 프로토콜, 버퍼 오버플로우, 정수 오버플로우, 메모리 할당 에러와 같은 결함 등을 효율적으로 찾을 수 있도록 해 준다.

보안 취약점 탐지 대상 목록은 SANS 25의 상위 10개 항목과 매칭되는 전자정부 소프트웨어 개발보안 가이드의 취약점 목록을 반영하였다.

표 2. 제안 기법의 취약점 목록 매칭

순위	SANS 25의 상위 10개 항목	소프트웨어 개발 보안 가이드	
		구분	항목
1	SQL삽입	입력 데이터 검증 및 표현	1.SQL삽입
2	운영체제 명령어 삽입		4.운영체제 명령어 삽입
3	버퍼 오버플로우		15.메모리 버퍼 오버플로우
4	크로스사이트 스크립트		3.크로스사이트 스크립트
5	적절한 인증 없는 중요기능 허용	보안기능	1.적절한 인증 없는 중요 기능 허용
6	부적절한 인가		2.부적절한 인가
7	하드코드된 중요정보		6.하드코드된 비밀번호
8	중요정보 평문 전송 및 저장		5.중요정보 평문저장
9	파일 업로드	입력데이터 검증 및 표현	5.위험한 형식 파일 업로드
10	보안기능 결정에 사용되는 부적절한 입력 값		14.보안기능 결정에 사용되는 부적절한 입력값

본 논문의 제안 기법과 기존 도구와의 비교 분석 대상 무기체계 소프트웨어는 A기관에서 사용 중인 '무기체계 형상관리 시스템'이며, 이 '무기체계 형상관리 시스템'은 총 16개 주요 모듈과 약 600여개 화면으로 구성되어 있다. 이 '무기체계 형상관리 시스템'은 전자정부 표준프레임워크 Version 3.6 기반으로 개발되었으며 소스 라인 수는 약 320만 라인에 이른다. 본 논문에서는 전자정부 표준프레임워크 자체 소스 및 화면 구성 소스는 비교분석 대상에서 제외하였다.

먼저, 정적분석도구인 PMD와 제안 기법간의 취약점 탐지 개수 비교 결과이다[표 3].

표 3. PMD와 제안 기법간의 취약점 탐지 결과

순위	소프트웨어 개발 보안 가이드		PMD			제안기법		
	구분	항목	전체	정탐	오탐	전체	정탐	오탐
1	입력 데이터 검증 및 표현	1.SQL삽입	4	4	0	18	16	2
2		4.운영체제 명령어 삽입	14	14	0	36	31	5
3		15.메모리 버퍼 오버플로우	0	0	0	0	0	0
4		3.크로스사이트 스크립트	1547	1256	291	3985	3724	261
5	보안 기능	1.적절한 인증 없는 중요 기능 허용	0	0	0	0	0	0
6		2.부적절한 인가	2	2	0	18	18	0
7		6.하드코드된 비밀번호	12	8	4	10	8	2
8		5.중요정보 평문저장	7	7	0	54	48	6
9	입력 데이터 검증 및 표현	5.위험한 형식 파일 업로드	5	5	0	18	17	1
10		14.보안기능 결정에 사용되는 부적절한 입력값	59	48	11	74	69	5

취약점 오탐지에 대한 판단은 각 취약점 항목의 전체 탐지 결과를 전수 조사하여 판단하였다. [표 3]에서 알 수 있듯이 본 논문의 제안 기법은 모든 항목에서 정탐의 개수가 많이 도출되어 기존 도구인 PMD보다 취약점 탐지 능력이 우수하다고 판단된다.

다음은 동적분석 도구인 SPIKE와 제안 기법간의 취약점 탐지 개수 비교 결과이다.

표 4. SPIKE와 제안 기법간의 취약점 탐지 결과

순위	소프트웨어 개발 보안 가이드		SPIKE			제안기법		
	구분	항목	전체	정탐	오탐	전체	정탐	오탐
1	입력 데이터 검증 및 표현	1.SQL삽입	2	2	0	18	16	2
2		4.운영체제 명령어 삽입	16	15	1	36	31	5
3		15.메모리 버퍼 오버플로우	0	0	0	0	0	0
4		3.크로스사이트 스크립트	431	412	19	3985	3724	261
5	보안기능	1.적절한 인증 없는 중요 기능 허용	0	0	0	0	0	0
6		2.부적절한 인가	8	8	0	18	18	0
7		6.하드코드된 비밀번호	0	0	0	10	8	2
8		5.중요정보 평문저장	28	21	7	54	48	6
9	입력 데이터 검증 및 표현	5.위험한 형식 파일 업로드	8	7	1	18	17	1
10		14.보안기능 결정에 사용되는 부적절한 입력값	22	14	8	74	69	5

[표 4]는 기존 동적분석 도구인 SPIKE와 제안 기법간의 보안 취약점 탐지 개수를 나타내고 있다. 탐지되지 않은 두 개의 항목을 제외하고 모든 취약점 항목에서 본 제안 기법이 정탐한 취약점 항목의 개수가 SPIKE 보다 많은 것으로 나타났다. 따라서 기존 동적분석 도구의 취약점 탐지 능력보다 본 논문의 제안 기법의 취약점 탐지 능력이 우수한 것으로 판단된다.

실제로 A 기관에서는 초기 사업 진행 시 발견하지 못한 취약점 제거를 위하여 고도화 사업을 통한 지속적인 취약점 제거 작업을 추진 중에 있다.

V. 결 론

본 논문에서는 어플리케이션에 내재되어 있는 취약점의 탐지 정확도를 높이기 위해, 기존의 분석 도구를 통해 얻은 결과를 재검증하여 정탐 정확도를 높이는 취약점 분석 기법을 제안하였다. 무기체계 소프트웨어 분야 중 무기체계 형상관리 시스템에 대한 실험 결과, 본 논문에서 제안하는 기법의 취약점 탐지 능력은 기존 정

적, 동적분석 도구의 취약점 탐지 능력보다 전반적으로 높게 나타났다. 따라서 본 연구에서 제안하는 기법이 기존 분석 도구(공개 소프트웨어:Open Source Software)보다는 탐지 능력과 정확도가 우수한 것으로 나타났다.

본 논문에서 제안하는 기법을 활용하여, 국방 무기체계 소프트웨어 획득 프로세스 상의 소프트웨어 통합 및 시험 단계를 거치게 된다면, 보다 안전한 무기체계 소프트웨어를 구현하는데 일조(一助)할 수 있을 것으로 판단된다. 하지만 제안하는 기법은 오픈 소스 분석 도구들을 이용한다는 한계점을 가지고 있으며, 기존의 분석도구가 가지고 있지 않는 취약점 탐지 규칙의 취약점에 대해서는 탐지할 수 없다는 단점이 존재한다. 이러한 단점을 보완하기 위해서는 오픈 소스 기반 분석 도구들을 대상으로 새로운 취약점들이 발견될 때마다 해당 취약점 탐지를 위한 규칙들이 업데이트가 가능한 시스템이 필요하다. 이러한 시스템의 구성은 향후 과제로 남아 있으며, 이와 함께 분석 도구들의 취약점 탐지 정확도, 적중률, 정밀도, 민감도, F-지표 등과 같은 통계 지표들을 산출하기 위한 연구도 필요하다.

참 고 문 헌

[1] 행정안전부, 한국인터넷진흥원, *소프트웨어 개발 보안 가이드,* 2017.

[2] 박상현, 최준성, “전장정보체계를 위한 시큐어 소프트웨어 구현에 관한 연구,” 한국통신학회 학술대회논문집, pp.620-621, 2017.

[3] 방위사업청, *무기체계 소프트웨어 개발 및 관리 매뉴얼,* 2017.

[4] 최준성, *시스템 특성에 최적화된 시큐어 코딩규칙 선정 평가 모형 개발,* 서울과학기술대학교 대학원, 학위논문박사, 2013.

[5] 행정안전부, *행정기관 및 공공기관 정보시스템 구축•운영 지침 개정안,* 2017.

[6] 방지호, *소스코드 보안약점 진단도구 기반의 소프트웨어 보안성 보증 방안 연구,* 홍익대학교 대학원, 학위논문박사, 2014.

[7] 최경철, 문관주, *이태연, 쉽게 배우는 안드로이드 앱 취약점 진단 : 안드로이드 앱 취약점 분석 실무 기술,* secuBOOK, 2015.12.

[8] 최경철, 김태한, *웹 모의해킹 및 시큐어코딩 진단 가이드,* secuBOOK, 2014.4.

[9] 김영숙, *해킹 방어를 위한 JAVA 시큐어코딩,* OPENEG, 2014.8.

[10] 김형주, 강정호, 김경훈, 이재승, 전문석, “융복합 전자정부 서비스를 위한 전자정부 표준프레임워크 기반 시큐어 코딩 점검 시스템 설계 및 개발,” 디지털융복합연구논문지, 제13권, 제2호, pp.201-208, 2015.

[11] 권경용, 주준석, 김태식, 오진우, 백지현, “신뢰성 시험 프로세스 개선을 통한 무기체계 내장형 소스코드 품질확보에 관한 연구,” 정보과학회논문지, 제42권, 제7호, pp.860-867, 2015.

[12] 김석모, *동적분석을 통한 정적분석의 과탐 보완에 관한 연구: 소프트웨어 취약점 탐지 사례 중심,* 단국대학교 대학원, 학위논문석사, 2015.

[13] 김준호, *소스코드 보안약점 탐지를 위한 정적도구의 활용과 기호실행 엔진을 활용한 개선 방안 연구,* 숭실대학교 정보과학대학원, 학위논문석사, 2016.

[14] 현요한, *웹 소스코드 분석을 통한 취약점 점검 효율에 관한 연구,* 서강대학교 정보통신대학원, 학위논문박사, 2017.

[15] 류인현, *웹 취약점 분석을 위한 CVSS 기반의 스코어링 기법,* 충북대학교 대학원, 학위논문석사, 2017.

[16] 최종석, *고성능 정적 코드 분석을 위한 취약점 연관 변수 분류기,* 부산대학교 대학원, 학위논문박사, 2017.

[17] 이문구, “사이버 국방 보안에 대한 연구,” 한국콘텐츠학회지, 제11권, 제4호, pp.18-22, 2013.

[18] 김정숙, “소프트웨어 보안을 위한 시큐어 코딩,” 한국콘텐츠학회지, 제11권, 제4호, pp.56-60, 2013.

저 자 소 개

김 종 복(Jong-Bok Kim) 정회원

- 1997년 2월 : 명지대학교 산업공학과(공학사)
- 2000년 2월 : 명지대학교 산업공학과(공학석사)
- 2016년 3월 ~ 현재 : 배재대학교 사이버보안학과 박사과정 재학

<관심분야> : 소프트웨어공학, 정보보호, 시큐어코딩

조 인 준(In-June Jo) 정회원

- 1982년 2월 : 전남대학교 계산통계학과 졸업
- 1985년 2월 : 전남대학교 전자계산학과 석사
- 1999년 2월 : 아주대학교 컴퓨터공학과 박사
- 1983년 ~ 1993년 : 한국전자통신연구원 선임연구원
- 1994년 ~ 현재 : 배재대학교 사이버보안학과 교수

<관심분야> : 정보보호, 컴퓨터네트워크보안, 전산조직응용