논문 2018-13-35

무기체계 소프트웨어의 자료경합을 탐지하기 위한 프레임워크

(A Framework for Detecting Data Races in Weapon Software)

오 진 우, 최 으 뜸, 전 용 기*
(Jin-Woo Oh, Eu-Teum Choi, Yong-Kee Jun)

Abstract : Software has been used to develop many functions of the modern weapon systems which has a high mission criticality. Weapon system software must consider multi-threaded processing to satisfy growing performance requirement. However, developing multi-threaded programs are difficult because of concurrency faults, such as unintended data races. Especially, it is important to prepare analysis for debugging the data races, because the weapon system software may cause personal injury. In this paper, we present an efficient framework of analysis, called ConDeWS, which is designed to determine the scope of dynamic analysis through using the result of static analysis and fault analysis. As a result of applying the implemented framework to the target software, we have detected unintended data races that were not detected in the static analysis.

Keywords : Multi-thread, Weapon system software, Concurrency faults, Data races, Dynamic analysis

Ⅰ. 서 론

무기체계는 전장에서 전투력을 발휘하기 위한 무기와 이를 운영하는데 필요한 장비, 부품, 소프트웨어 등 제반요소를 통합한 것이다. 무기체계 소프트웨어는 무기체계의 임무와 관련된 기능들을 어떠한 조건과 환경 속에서도 반드시 주어진 시간 내에 정확히 수행해야 한다. 미래 전장 환경이 네트워크 중심으로 진화됨에 따라 무기체계에서 소프트웨어를 이용한 기능구현은 점점 증가하고 있다.

무기체계 소프트웨어에서 결함의 존재는 장비와 비용 손실뿐만 아니라 인명 피해 등의 치명적인 결과를 초래할 수 있다. 따라서 무기체계 소프트웨어의 개발 과정 중에 검증하는 단계는 소프트웨어 코드가 일으킬 수 있는 결함을 사전에 발견하고 이를 제거하여 신뢰성을 높일 수 있기 때문에 중요하다 [1].

*Corresponding Author (jun@gnu.ac.kr)
Received: Apr. 30 2018, Revised: June 7 2018, Accepted: Sep. 6 2018.
J.W. Oh: Gyeongsang National University, DTaQ
E.T. Choi, Y.K. Jun: Gyeongsang National University
※ 본 논문은 2015년 경상대학교의 발전기금재단재원으로 수행되었음.

ⓒ IEMEK J. Embed. Sys. Appl. 2018 Dec. 13(6) 305-312
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2018.13.6.305

미래 전장의 첨단 복합화로 인하여 무기체계의 연산처리 능력 강화가 지속적으로 요구되고 있다. 이에 따라 멀티스레드 프로그래밍을 사용하여 처리 능력을 향상시키고자 노력하고 있지만 필연적으로 자료경합 (data races)과 같은 동시성 결함 (concurrency faults)에 노출된다. 자료경합은 의도하지 않은 비결정적인 결과를 초래하기 때문에 시험 및 디버깅을 위해서 반드시 탐지되어야 한다. 하지만 자료경합을 탐지하고 제거하는 것은 프로그램의 모든 인터리빙을 고려해야 하므로 매우 어렵다 [2].

자료경합의 탐지는 정적시험과 동적시험으로 나뉜다. 정적시험은 소스코드를 분석하기 때문에 잘못된 알람이 포함되어 결과의 신뢰성이 낮다. 동적시험은 수행 중에 발생한 자료경합만 보고하기 때문에 정적시험보다 정확도가 높지만 오버헤드가 크다. 기존의 국내 연구결과 및 방위사업청의 매뉴얼은 정적시험으로만 자료경합을 탐지하기 때문에 운용중인 프로그램에 자료경합이 발생될 수도 있다 [3].

본 논문에서는 무기체계 소프트웨어의 자료 경합의 탐지율을 높이기 위하여 동적시험도구를 적용한 프레임워크인 ConDeWS (concurrency detection weapon system software)를 제안한다. ConDeWS 프레임워크는 기존의 정적시험 수행 결

과를 FAR (fault analysis report)로 분석하고, 그 분석 결과를 바탕으로 선정한 범위에 대한 동적시험을 수행하도록 설계되었다. 본 논문에서는 실제 대상 소프트웨어에 적용함으로써 제시한 ConDeWS 프레임워크가 자료경합의 탐지율이 향상시킴을 보인다.

Ⅱ. 연구배경

무기체계에서 소프트웨어를 이용한 기능구현이 점점 증가하고 있다. 이에 따라 무기체계 소프트웨어의 성능을 만족시키기 위하여 다중 스레드로 연산능력을 향상시키고 있으나 필연적으로 자료경합에 대한 위험성에 노출된다. 본 장에서는 무기체계 소프트웨어, 자료경합, 그리고 자료경합 탐지에 대해 설명한다.

1. 무기체계 소프트웨어

무기체계는 유도무기, 항공기, 함정 등 전장에서 전투력을 발휘하기 위한 무기와 이를 운영하는데 필요한 장비, 부품, 소프트웨어 등 제반요소를 통합한 것으로서 지휘·통제통신무기체계, 감시정찰무기체계, 기동무기체계, 함정무기체계, 항공무기체계, 화력무기체계, 모의분석훈련 소프트웨어 및 장비 등 그 밖의 무기체계로 분류된다.

무기체계에서 소프트웨어를 이용한 기능구현이 점점 증가하고 있으며, 전투기의 경우 소프트웨어로 구현되는 기능이 1960년대 생산된 F-4가 8%였지만 2007년에 생산된 F-35의 경우 90%가 소프트웨어에 의해 기능이 구현되었다. 이러한 무기체계 개발에서 주요 기능을 소프트웨어로 구현하는 비중이 폭발적으로 증가함에 따라 무기체계 소프트웨어 개발과 검증의 중요성이 증가하고 있다 [1].

무기체계 소프트웨어의 높은 성능요구조건을 만족시키기 위하여 설계 시 멀티스레드를 고려하여 연산능력을 향상시키기 위한 노력이 진행되고 있으나 필연적으로 자료경합 (data races)과 같은 동시성 결함 (concurrency faults)에 노출된다.

2. 자료경합

자료경합은 병행하게 수행되는 둘 이상의 스레드들이 적절한 동기화 없이 공유 자원에 적어도 하나 이상의 쓰기 접근을 포함할 때 발생한다. 자료경합에 의한 실제 사고사례는 1985년 Therac-25 방사능에 의한 인명피해 [4], 2003년 약 60억 달러의 경제적 손실이 발생되었던 미국 북동부 대정전 [5], 그리고 2012년 페이스북 (Facebook) 기업공개 주식거래 지연 [6]이 있다. 또한 2017년 까지 지난 10년간 오픈소스 소프트웨어 프로젝트의 버그 리포트에서 11,860개의 리포트 중 351개의 리포트에서 자료경합이 발생되었음을 알 수 있다 [7].

사고 사례처럼 자료경합은 발생 시 인적 및 물적 피해를 줄 수 있으므로 소프트웨어 개발 중에 시험 및 탐지를 통해 제거되어야 한다. 특히 임무 긴요도가 높은 무기체계 소프트웨어는 신뢰성, 생존성, 그리고 실시간성을 우선적으로 고려해야하기 때문에 [8] 자료경합에 대한 강력한 대비가 필요하다. 하지만 다중 스레드의 다양한 인터리빙을 모두 고려해야 하므로 정확하게 자료경합을 탐지하고 제거하는 것은 매우 어렵다 [2].

3. 자료경합 탐지

자료경합의 탐지는 프로그램의 수행 없이 소스코드로만 모든 스레드 인터리빙을 고려하여 탐지하는 정적시험 (static testing)과 프로그램의 수행 중에 스레드의 발생과 동기화 정보 및 메모리 접근 등을 감시 및 분석하여 탐지하는 동적시험 (dynamic testing)으로 나뉜다. 정적시험은 실제 프로그램의 수행에서 발생할 수 없는 인터리빙까지 고려하여 다수의 잘못된 알람 (false alarms)이 포함되어 결과의 신뢰성이 낮다. 동적시험은 프로그램의 추적, 재수행, 감시를 통해 수행 중에 발생한 실제 자료경합만 보고하기 때문에 정적시험보다 탐지 정확도가 높다. 하지만 실행 중에 동시성 정보를 유지시켜야 하므로 시간 및 공간 오버헤드가 높다.

국내에서 무기체계 소프트웨어를 개발 할 경우 방위사업청의 “무기체계 소프트웨어 개발 및 관리 매뉴얼”을 따른다. 하지만 이 매뉴얼에서는 자료경합 탐지에 대한 집약적인 고려가 포함되어 있지 않고 미국 MITRE의 CWE 규칙의 일부를 참조하여 정적시험의 한 부분인 취약점 점검만 고려한다 [1, 3, 8-13]. 그리고 자료경합 탐지에 대한 국내 연구는 무기체계 소프트웨어에 정적시험을 수행하여 특정 유형의 자료경합이 탐지되었는지 제시하거나 [11], 정적시험을 통해 자료경합을 탐지해야 하는 필요성을 제시하는 [1] 사례만 있어 동적시험이 고려되고 있지 않다.

하지만 무기체계 소프트웨어를 개발 할 때 정적시험으로만 자료경합을 탐지하는 것은 신뢰도가 낮다 [2]. 또한 실제 프로그램 수행 중에 발생될 수 있는 자료경합을 미탐지 (false negative) 할 수 있다. 정적시험을 통해 발생하는 잘못된 탐지는 소프

표 1. NASA Handbook의 조사 시험도구
Table 1. Testing tools on NASA handbook

No	Tools Name	Supplier
1	Code Sonar	Grammatech
2	Astree	AbsInt
3	KlocWork	Rogue Wave Software
4	PolySpace	MathWorks
5	CodeHawk	Kestrel Technology

트웨어의 테스팅 과정에서 많은 시간 및 인력 소비를 발생 시킨다. 그리고 미탐지된 자료경합은 무기체계 소프트웨어의 실 운용 중에 발생하게 되면 신뢰성, 생존성, 그리고 실시간성을 보장하지 못하게 될 수도 있다. 따라서 자료경합은 임무의 실패뿐만 아니라 인적 또는 물적 손실을 야기할 수도 있기 때문에 반드시 제거되어야 한다.

Ⅲ. 탐지 프레임워크

본 논문에서는 무기체계 소프트웨어의 자료 경합의 탐지율을 높이기 위하여 동적탐지도구를 적용한 프레임워크인 ConDeWS를 제안한다. 본 장에서는 자료경합 탐지 도구를 분석하고 ConDeWS 프레임워크를 소개한다.

1. 자료경합 탐지 시험도구

본 논문에서 제안하는 ConDeWS 프레임워크를 구현하기 위하여 자료경합 탐지 기능을 제공하는 국내·외 시험도구들을 조사하였다. 조사 대상은 미국 NASA에서 2013년도에 발행한 Engineering Handbook에 소개된 시험도구들이다. 이 핸드북은 NASA가 실제 사용하고 있는 다양한 상용 COTS와 오픈소스를 소개한다 [14].

NASA의 Engineering Handbook에서 등록된 소프트웨어 관련 도구는 총 101개이다. 그 중 자료경합 탐지 기능을 제공하는 도구는 5종이다. 무기체계 소프트웨어는 특성상 대부분 C/C++ 언어로 개발되기 때문에 JAVA와 같은 다른 언어를 대상으로 하는 도구들은 제외하였다. 조사된 도구들은 아래 표 1과 같다. 이 도구들은 모두 C/C++ 언어 분석이 가능하며 상용 COTS로 판매되는 제품이고 구문분석을 수행하는 정적시험도구이다.

다음으로 상용 COTS로 판매되거나 오픈소스로 공개되고 있는 국·내외 시험도구들에 대해서도 조사하였다. 상용 COTS는 국내의 소프트웨어 시험도구를 제작 및 판촉하는 업체를 대상으로 조사하였으며, 오픈소스는 공개 사이트와 기존 연구논문에 기재된 시험도구들을 조사하였다. 그 결과, 총 7종의 시험도구에 대해서 자료경합이 탐지 가능함을 확인할 수 있었다. 조사한 시험도구들은 모두 C/C++언어를 제공하는 동적시험 도구로써 DT10이라는 1개의 상용 COTS와 TSAN, Maple, RaceChaser 등 6개의 오픈소스 소프트웨어였다 [15-18].

2. ConDeWS 프레임워크

ConDeWS 프레임워크는 입력으로 들어온 컴퓨터 파일을 총 4개의 Phase를 통해 최종적으로 정확성 높은 자료경합 탐지 결과를 도출해 낸다. 그리고 동적 시험을 효율적으로 수행하기 위하여 정적 시험의 결과를 토대로 동적 시험의 범위를 지정하는 FAR (fault analysis report)를 사용한다.

그림 1은 ConDeWS 프레임워크의 전체 구조를 보여주며 각 단계 별 역할은 다음과 같다.

- Phase 1: Compile 수행 단계
- Phase 2: 정적시험수행 및 FAR 분석단계
- Phase 3: 선정된 시험 범위로 동적시험수행
- Phase 4: 자료경합 결과 취합 및 분석

FAR는 Phase 2와 Phase 3간에 유기적인 연결관계를 형상하여 자료경합을 탐지하기 위한 정적 시험결과를 다각도로 분석하여 동적 시험을 효율적으로 수행하기 위한 방안을 제시한다.

정적 시험과 동적 시험의 Phase를 분할함에 따라 각각의 Phase 별로 영향성을 최소화함으로써 FAR의 수정을 통해 쉽게 도구 추가가 가능하다. 따라서 추가적으로 신규 시험도구를 추가 하는 데에 발생할 수 있는 제약이 없다. 또한 정적시험 수행 Phase의 결과를 FAR로 분석함으로써, 그 다음 단계인 동적시험 수행 Phase의 시험범위 설정에 도움을 줄 수 있도록 단계별로 구성하였다. 이는 무기체계 개발 시, 제한적인 개발시험평가 기간 동안 최대한의 효율성을 발휘하여 잠재된 자료경합을 탐지하기 위해 동적시험 범위를 설정함으로써, 시험에 소모되는 많은 시간을 감소시킬 수 있다.

하지만 ConDeWS 프레임워크는 자료경합을 탐지하기 위하여 기존의 정적 시험도구에 추가적으로 동적 시험도구를 포함시키므로, 동적시험 도구 구매에 따른 초기 프레임워크 구축비용이 발생된다. 또한 각각 Phase로 나누어 정적시험을 수행하고, 그 결과를 FAR로 분석한 후 동적시험을 수행하도록 설계되었기 때문에 기존 연구에서 다루던 정적시험만 수행하는 것보다 시험기간이 추가적으로 요구된

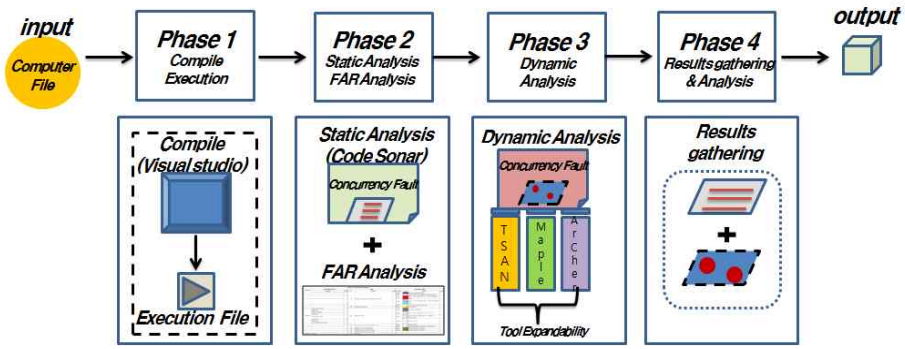

그림 1. ConDeWS 프레임워크 구조
Fig. 1 Structure of ConDeWS Framework

다는 제약사항이 있다. 과거 국방기술품질원에서 무기체계 소프트웨어에 대한 정적시험과 동적시험의 수행기간을 비교해 보면, 정적시험의 경우에는 평균 5.3일 (13건/17년도), 동적시험은 평균 21일 (시범사업 2건/15년도)가 소요되었다 [9]. 따라서 ConDeWS 프레임워크를 구축하게 될 경우, 산술적으로 정적시험만 수행하던 것 보다 시험기간이 약 4배 정도 더 소요될 것으로 예상된다.

Ⅳ. 구 현

1. 각 Phase별 소개

ConDeWS 프레임워크의 Phase 1은 대상 소프트웨어에 대한 Compile 정보와 실행파일을 생성하는 단계로 구성되어 있다. 대상 소프트웨어의 컴퓨터 파일이 입력으로 들어오게 되면 Phase 2의 정적 시험도구가 지원하는 Compiler를 활용하여 우선적으로 Compile를 수행하여 실행파일을 생성한다.

Phase 2에서는 생성한 Compile 정보와 실행파일로 정적시험을 수행하고 그 결과를 FAR로 분석한다. FAR는 정적시험 결과를 분석하여 잠재적인 자료경합이 존재하는 부분의 특정 위치를 지정해줄 수 있으며 다음 Phase에서 동적시험을 수행할 때 시험범위를 설정하는 데에 지표가 된다.

Phase 3에서는 앞서 Phase 2에서 FAR 분석한 정적시험 결과 중 자료경합 관련된 결함을 식별해내고 그 결함이 발생된 부분을 동적시험 범위로 선정한다. FAR의 Main Related Rule에 해당되는 결함이 검출되었을 경우, Phase 3의 동적시험을 통해서 정확한 자료경합 위치를 특정하여 탐지할 수 있다. 또한 Sub Related Rule에 부합될 경우에도, 동적시험으로만 탐지될 수 있는 자료경합도 탐지할 수 있다.

Phase 4는 결과취합 및 분석 단계이다. 앞선 Phase 2와 3에서 도출된 결과물들을 취합하고 정적 및 동적시험 결과를 분석함으로써 최종적으로 대상 소프트웨어에 잠재되어 있는 자료경합 탐지결과를 밝히는 과정이다. 세부내용은 그림 1과 같다.

2. FAR 세부내용

FAR는 다음과 같이 총 3개의 구성요소를 가지고 있다.

- Tool Provided Rule : 정적시험 도구 자체에서 제공하는 Rule중, 자료경합 관련 Rule 항목
- CWE Main Related Rule : CWE 규칙 중 자료경합과 직접적으로 관련된 규칙 총 28개
- CWE Sub Related Rule : 자료경합과는 직접적인 연관관계는 적지만, 앞서 정의한 Main Related Rule과는 간접적인 연관이 있는 규칙 총 94개

추가 정적시험 도구를 프레임워크에 추가하기 위해서는 FAR의 Tool Provided Rule 부분만 작성하면 되기 때문에 신규 도구 추가가 매우 용이하다.

CWE Main Related Rule과 Sub Related Rule은 CWE 규칙 1,006개 중 자료경합과의 관련성을 분석해 총 122개의 규칙을 도출한 것이다 [12].

실제 CWE 규칙에서는 각각의 규칙마다 연관되는 규칙들을 소개하고 있기 때문에, 자료경합과 직접적으로 연관관계는 적지만 Main Related Rule로 선정한 규칙과 관련 있다고 명시되어 있는 규칙들을 Sub Related Rule로 선정하였다.

FAR를 통해 정적시험 결과를 분석하여 자료경

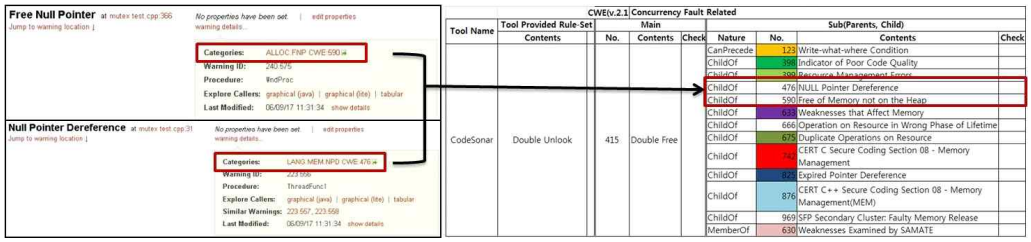

그림 2 Phase 2 수행결과 (정적시험 결과를 FAR로 비교 분석)
Fig. 2 Phase 2 execution results (Mapping of CWE rules and FAR analysis)

합의 발현가능성이 있는 동적시험 범위를 선정하여 적용할 수 있다. 예를 들어, 정적시험 후 그 결과를 FAR와 비교 분석해보니 실제 도구에서 제공하는 자료경합 탐지 관련 Rule-Set에 해당되거나 앞서 정의한 Main Related Rule에 해당할 경우, 동적시험 범위로 우선적으로 포함할 수 있다. 위의 두 경우에는 자료경합의 정확한 발현가능 지점을 선택할 수 있기 때문에 동적시험 범위를 명확하게 선정하여 탐지율을 높일 수 있다. 또한 정적시험 결과 중에 앞선 두 규칙에 해당되지 않다 하더라도, 만약 Sub Related Rule에 해당할 경우에도 동적시험 범위에 포함시켜서 동적시험으로만 탐지되어 잠재적으로 발생될 수 있는 자료경합을 탐지할 수 있도록 설계하였다.

Ⅴ. 실험

1. 실험 설계

본 장에서는 제안한 프레임워크의 평가를 위해 ConDeWS 프레임워크를 실제 무기체계 품목의 소프트웨어에 적용한다. 그리고 기존 연구 결과인 정적시험에서 도출되었던 결과와 ConDeWS 프레임워크 간의 결과를 비교한다.

ConDeWS 프레임워크는 앞서 조사한 도구들 중에서 1개의 정적시험 도구와 2개의 동적시험 도구를 선정하여 적용하였다. 선정된 정적시험 도구는 Grammatech에서 제공하는 CodeSonar이며 동적시험 도구는 오픈소스 도구인 RaceChaser와 상용 COTS 인 DT10이다. 정적시험 도구인 CodeSonar를 적용하기 위해서 FAR의 구성요소 중 Tool Provided Rule-Set에 CodeSonar 도구에서 제공하는 자료경합 탐지 규칙을 추가하여 정적시험 결과를 FAR로 분석할 수 있도록 설정한다.

ConDeWS 프레임워크의 개략적인 평가 절차는, 먼저 정적시험 결과를 FAR로 분석하여 다음 Phase 인 동적시험의 범위로 선정하고 그에 따라 동적시험을 수행한다. 다음으로 정적시험 결과와 동적시험 결과를 분석하여 기존의 문제점이 해결되었는지를 확인한다.

표 2. Phase 2 수행 결과
Table 2. Phase 2 execution results

Warning Class	Active Warnings
Null Pointer Dereference	8
Free Null Pointer	5
Unreachable Call	4
Unused Value	3
Redundant Condition	3
Ignored Return Value	1

2. 실험 결과

본 논문에서 제안한 ConDeWS 프레임워크를 무기체계 품목 중 전원 공급을 위한 부품에 적용하였다. 대상 소스코드는 총 1,614라인, 13개의 함수로 구성되며, 전체 분기에 따라 생성되어야 하는 총 Test Case는 69개로 구분되었다. 프레임워크 Phase 2에서 정적시험을 수행한 결과 총 24개의 결함이 탐지되었고 Null Pointer Dereference가 8건, Free Null Pointer가 5건, Unreachable Call이 4건이 탐지되었으며 세부 내용은 표 2와 같다.

세부 분석정보를 살펴보면 그림 2와 같이 CWE 규칙과 관계되는 정보를 얻을 수 있다. 즉, 시험도구 자체에서 탐지된 결함이 CWE 규칙과의 매핑 정보를 제공하는 것이다. 따라서 각각 Null Pointer Dereference 규칙은 CWE 476, Free Null Pointer는 CWE 590과 매핑되는 것을 확인할 수 있다. 이렇게 얻어진 결함 정보를 앞서 작성한 FAR와 비교하여 탐지된 결함이 자료경합과 관계되는지 확인한다.

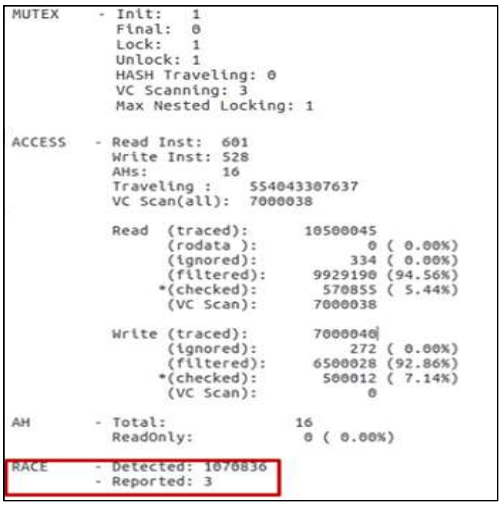

그림 3. Phase 3 수행결과 (RaceChaser)
Fig. 3 Phase 3 execution results of RaceChaser

앞서 CWE 규칙과 매핑한 정보를 FAR와 비교한 결과 시험 도구 자체에서 제공하는 규칙인 'Double Unlook'과 CWE Main Related Rule인 CWE 415 (Double Free) 항목의 Sub Related Rule내에 포함됨을 확인할 수 있었다. 실제 정적시험 결과에서 FAR의 Sub Related Rule과 일치되는 결과가 탐지되었으므로, 해당 결함이 도출된 5개 함수를 동적시험 대상으로 포함시켰고, 5개 함수의 분기문에 따른 테스트 케이스는 16개로 구분되었다.

ConDeWS 프레임워크에서는 동적시험 도구인 RaceChaser와 DT10 총 2개를 활용하여 동적시험을 수행하였다. 우선 RaceChaser로 시험한 결과는 그림 3과 같으며, 그림과 같이 자료경합이 탐지되었음을 확인할 수 있다.

다음으로 또 다른 동적시험 도구인 DT10으로 대상 소프트웨어의 동일 파일에 대하여 동적시험을 수행하였다. 대상 소프트웨어의 함수 수행은 함수 A (첫번째 for문, '①'로 표기) - 함수 A (두번째 for문, '②'로 표기) - 함수 B ('③'으로 표시) 순으로 진행된다. 하지만 수행 결과를 보면 그림 4와 같이 실제 함수 수행순서 (①→②→③)와는 다르게 불규칙한 순서대로 함수가 수행되고 있음을 확인할 수 있다.

시험결과, 본 프레임워크에서 동적시험 범위를 한정함으로써 시험해야 하는 범위를 함수기준 61% (8개/13개), Test Case 기준 75% (53개/69개) 감소시킬 수 있었다. 또한 무기체계 개발 시 자료경합에 대해서 기존에 수행되던 정적시험 뿐만 아니라,

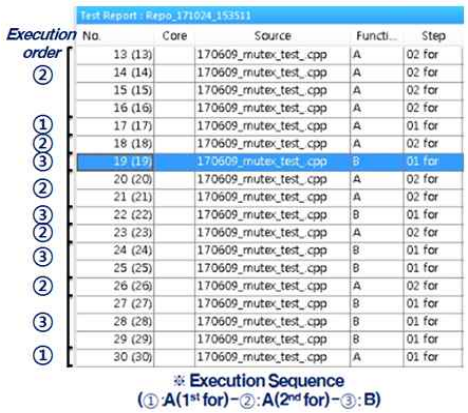

그림 4. Phase 3 수행결과(DT10)
Fig. 4 Phase 3 Execution Results of DT10

동적시험 중 코드 실행률 외에 자료경합이 특화된 프로세스를 포함하는 새로운 프레임워크를 제시하였다.

Ⅵ. 결 론

미래 전장환경이 첨단 복합화로 진화됨에 따라 무기체계에서 소프트웨어를 이용한 기능구현이 강화되고 있으며, 소프트웨어 개발과 검증의 중요성도 증가하고 있다. 이러한 무기체계 소프트웨어의 높은 성능요구조건을 만족시키기 위하여 설계 시 멀티스레드를 통해 연산능력을 향상시키기 위한 노력이 진행되고 있으나, 필연적으로 자료경합에 노출된다.

본 논문에서는 무기체계 소프트웨어의 자료경합의 탐지율을 향상시키기 위하여 정적시험과, FAR의 분석 결과를 토대로 수행되는 동적시험을 포함하는 ConDeWS 프레임워크를 제안하였다. 프레임워크에 대상 소프트웨어를 적용한 결과 실제 프로그램 수행 중에 발생될 수 있는 결함을 도출해내는 방식인 동적시험으로만 탐지되는 자료경합에 대해서 탐지할 수 있음을 확인하였다. 무기체계 개발 시 적용할 경우, 제한된 시험기간 동안에 정적시험과 동적시험을 모두 포함하여 수행할 수 있고, 동적시험 범위 역시 선정할 수 있는 방안을 마련할 수 있었다.

앞으로 FAR의 Main, Sub Related Rule간의 관계를 분석하여 향후에는 동적시험 범위선정 정확도를 향상 시키는 방안과 대상 소프트웨어에 적합한 동적시험 도구 추천에 대한 연구를 진행할 것이다. 또한 현재는 FAR가 CWE를 기반으로 작성되었으나 향후에는 타 국제규격을 추가하여 좀 더 강화된

ConDeWS 프레임워크를 제안할 것이다. 마지막으로 FAR를 생성하고 분석하는 프로세스의 자동화를 연구함으로써, FAR 제작 시 일관성 있는 결과를 도출해내고 누구나 활용할 수 있도록 개발할 것이다.

References

[1] G.M. Choi, Y.H. Kim, G.I. Woo, "Software Reliability Assurance for Weapon System Quality Improvement(2)," Journal of Defense & Technology, Vol. 404, pp. 74-85, 2012. (in Korean)

[2] O.K. Ha, Y.K. Jun, "A Survey on Dynamic Detection for Data Races in Multithread Programs," Journal of KIISE, Vol. 33, No. 12, pp. 45-51, 2015. (in Korean)

[3] N. G.Leveson, C. S.Turner, "An Investigation of the Therac-25 Accidents," Journal of IEEE Computer Society, Vol. 26, No. 7, pp. 18-41, 1993.

[4] K. Poulsen, Software Bug Contributed to Blackout, SecurityFocus, 2004.

[5] M. Farrell, Facebook IPO : Wall Street's losses mount, CNNMoneyInvest, 2012

[6] S. A. Asadollah, D. Sundmark, S. Eldh, H. Hansson, "Concurrency Bugs in Open Source Software : a Case Study," Journal of Internet Services and Applications, Vol. 8, No. 1, 2017

[7] G.R. Lee, "Study on Management Improvement Plan of Ground Weapon System," Journal of Defense & Technology, Vol. 438, pp. 72-83, 2015. (in Korean)

[8] DAPA, Weapon System Software Development and Management Manual, DAPA, pp. 7-1 - 7-14, 2016 (in Korean).

[9] K.Y. Kwon, J.S. Joo, T.S. Kim, J.W. Oh, J.H. Baek, "A Study on Quality Assurance of Embedded Software Source Codes for Weapon Systems by Improving the Reliability Test Process," Journal of KIISE, Vol. 42, No. 7, pp. 860-867, 2015. (in Korean)

[10] Y.G. Song, Y.S. Park, Y.G. Lee, H.J. Jeong "Defense Field Weapon System Reliability Improvement Plan," Journal of Defense & Technology, Vol. 449, pp. 116-129, 2016. (in Korean)

[11] J.H. Noh, J.M. Lee, Y.H. Park, "Defect- Type Analysis of Regional SW Development Companies using CodeSonar," Journal of KIICE, Vol. 19, No. 3, pp. 683-688, 2015. (in Korean)

[12] MITRE, Common Weakness Enumeration (CWE), 2017, MITRE

[13] J. W. Kim, "A Case Study on Reliability Test of Embedded Software in the Multi-Function Radar," Journal of IKEEE, Vol. 19, No. 3, pp. 431-439, 2015. (in Korean)

[14] NASA, NASA Software Engineering HandBook (NASA-HDBK-2203), NASA, 2013.

[15] B. Norris, B. Demsky, "CDSchecker: Checking Concurrent Data Structures Written with C/C++ Atomics," Journal of ACM SIGPLAN Notices, Vol. 48, No. 10 , pp. 131-150, 2013.

[16] C. Flanagan, S. N. Freund, "FastTrack: Efficient and Precise Dynamic Race Detection," Journal of ACM SIGPLAN Notices, Vol. 44, No. 6, pp. 121-133, 2009.

[17] I. Kuru, H. S. Matar, A. Cristal, G. Kestor, O. Unsal, "PaRV : Parallelizing Runtime Detection and Prevention of Concurrency Error," Proceedings of Runtime Verification, pp. 42-47, 2012.

[18] J. Yu, S. Narayanasamy, C. Pereira, G. Pokam, "Maple: A Coverage-Driven Testing Tool for Multithreaded Programs," Journal of ACM SIGPLAN Notices, Vol. 47, No. 10, pp. 485-502, 2012.

Jin-Woo Oh (오 진 우)

He is received the MS degree in Informatics from Gyeongsang National University in 2018. He is currently an researcher in DTaQ.

Email: jwoh@dtaq.re.kr

Eu-Teum Choi (최 으 뜸)

He is received the MS degree in Informatics from Gyeongsang National University in 2015. He is enrolled PhD candidate from the department of Informatics in Gyeonsang National University.

Email: slateblue33@gnu.ac.kr

Yong-Kee Jun (전 용 기)

Yong-Kee Jun received his BS degree in Computer Engineering from Kyungpook National University, and the MS and PhD degrees in Computer Science from Seoul National University. He is now a full professor in Department of Aerospace and Software Engineering, Gyeongsang National University (GNU). He is the founding director of GNU Embedded Software Center for Avionics (GESCA), one of the national Information Technology Research Centers (ITRCs) of South Korea. His professional interests include dependable software, embedded software, and airborne software. Prof. Jun is a member of Association for Computing Machinery (ACM).

Email: jun@gnu.ac.kr