3

Runtime Error의 정적 분석을 통한 소프트웨어 보안 취약성 검출

파수닷컴 ▌안성민 · 진민식 · 조규진

1. 서 론

정보 보안은 일반적으로 크게 네트웍 보안 / 시스템 보안 / 데이터베이스 보안 / 어플리케이션보안의 4부분으로 구분된다(그림 1 참조). 소프트웨어에 있는 보안 취약성에 대한 보안은 어플리케이션 보안의 한 부분으로 분류된다. Internet의 출현과 함께 하는 정보보안에 대한 관심과 연구는 이미 많은 대응방법과 솔루션을 낳고 있다. 그러나 어플리케이션 보안 분야는 데이터 암호화 부분을 제외하면 아직 기술적으로 초기 단계에 머무르고 있으며, 이는 상당부분 소프트웨어 보안 취약성 자동 검출의 기술적 어려움에 기인하고 있다. 대부분 동적 테스트 혹은 개발자에 의한 code review에 의존하고 있던 소프트웨어 보안 취약성 검출은 최근에야 정적 분석을 통한 자동 검출이 이루어지고 있다. 특히 프로그램의 실행 의미를 이해해야 하는 runtime 오류 검출은 그 기술적 난이도 때문에 관련 솔루션 역시 아직은 희박한 실정이다. 본 기고에서는 runtime 오류 정적 분석기 간의 비교를 통해, 분석 결과 차이가 어디에서 비롯하는지, 정적 분석이 왜 어려운지 등에 대하여 간략히 다루고자 한다.

2. 보안 취약점 측면에서 본 Runtime 오류의 성격 및 처리방안

소프트웨어 보안 취약점 측면에서 중요하게 다뤄지는 runtime 오류로는 가장 대표적인 buffer overflow를 비롯하여 memory leak, Null dereference, uninitialized variable 등이 있다. 이러한 runtime 오류들의 정적 분석을 위해서는 프로그램의 동작 흐름을 정확히 이해할 수 있는 깊은 분석력이 필요하다. 전체 프로그램 내에서 변수의 값들이 어떻게 변화하는지, 함수 간 호출 관계가 어떻게 되는지 등을 알 수 있어야 할 것이다. 더불어 이들 분석을 유한한 시간 내에 한정된 컴퓨터 자원을 통해서 끝내야 한다. 이러한 요구사항들은 runtime 오류의 정적 분석을 위한 분석기 설계를 어렵게 하고 있으며, 설계 철학 및 구현 난이도에 따라 분석기 마다 다른 결과를 낳는 원인이 된다. Runtime 오류 정적 분석기 간의 비교 분석은 이러한 요구사항들이 얼마나 잘 반영 되어 있는지를 알아 볼 수 있을 뿐만 아니라, 왜 정적 분석이 어려운지를 살펴 볼 수 있는 좋은 방법 중의 하나이다.

3.보안 취약성 관련 Runtime 오류 분석기 설계 및 성능 자료

아래 표 1에서는 분석기 간 비교를 위한 평가 기준을 보여 주고 있다. 평가 기준은 테스트 전문 업체의 검사 기준과 행정안전부 보안강화체계 구축 사업의 요

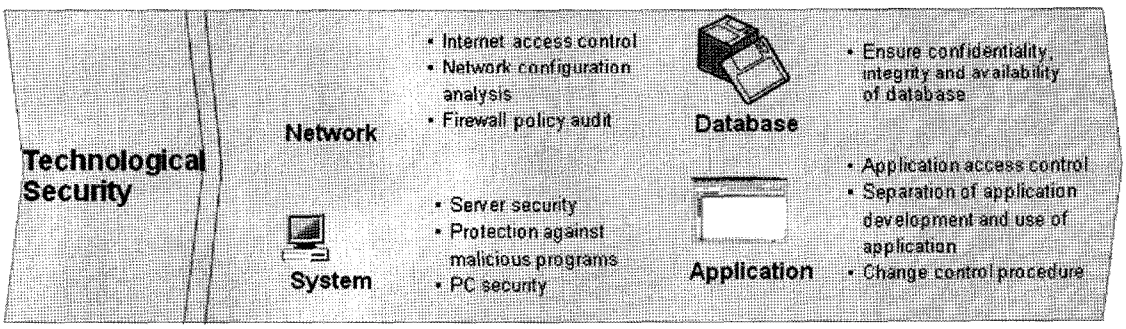


그림 1 정보 보안 중 기술 영역에 대한 분류

표 1 Runtime 오류 정적 분석기 비교를 위한 평가 기준

오류 종류	분석 요구사항	
Buffer Overflow	검출 메모리 영역	Stack Overflow, Heap Overflow 모두
	함수 호출 추적 여부	다수의 Function call을 통한 주소 전달을 추적(함수 호출 추적 안함 /3 depth 이내 추적/5 depth 이상 추적 등)
	버퍼의 구성 형태	단순 배열 구조체의 배열 멤버 배열의 시작주소를 받은 구조체의 포인터
	Overrun 방식	직접 read/write Loop 구문에 따른 off-by-one
	배열의 인덱스 이해 범위	상수값 함수의 반환값 조건문에 따른 인덱스 범위 고려 정수 및 변수 연산값
Null Dereference	함수 호출 추적 여부	다수의 Function call을 통한 주소 전달을 추적(함수 호출 추적 안함/ 3 depth 이내 추적/5 depth 이상 추적 등)
	포인터 이해 범위	단일 포인터, 더블 포인터, 포인터 배열
	Null assign 인식 범위	Macro NULL assign If 조건문의 NULL 비교
Resource(Memory) Leak	함수 호출 추적 여부	다수의 Function call을 통한 주소 전달을 추적(함수 호출 추적 안함/ 3 depth 이내 추적/5 depth 이상 추적 등)
	포인터의 구성 형태	단순 file/heap memory 포인터 다차원 포인터 포인터 배열 구조체 멤버
	변수의 scope	지역 변수 전역 변수
	할당/해지 인식 범위	Assign에 의한 주소 복사 인식 caller에서 할당, callee에서 leak 검출 callee에서 할당, caller에서 leak검출
	Leak pattern	해지 없이 return Scope 이탈로 가비지 발생
Uninitialized Local Variable	함수 호출 추적 여부	다수의 Function call을 통한 주소 전달을 추적(함수 호출 추적 안함/3 depth 이내 추적/5 depth 이상 추적 등)
	미초기화 변수 종류	정수형 배열 포인터 구조체
	미초기화 변수의 사용 여부 판단	산술 연산 배열의 index로 사용 함수의 입력값으로 사용 다른 변수에 assign Bit 연산 간접 참조 Return

구사항을 포함하여 자체적으로 작성 하였다.

표 2에서는 표 1의 기준을 포함하는 간단한 C 언어 예제 프로그램을 만들어 서로 다른 두 분석기를 비교한 결과 요약을 보여 주고 있다(자세한 비교 항목은 별첨 참조). 분석기로는 행안부 보안성 강화 구축 프로젝트에서 구현된 C 언어용 runtime 오류 정적 분석기와 외산 상용 보안 취약점 분석기를 선택하였다. 행안부 분석기가 총 93건 중 77건을 검출하여 82%의 분석력을 보인 반면, 외산 보안 취약점 상용 도구는 22건을 검출하여 23%의 분석 결과를 보여 주고 있다.

표 2 행안부 보안성 강화 구축 프로젝트 C 언어 runtime 오류 정적 분석기와 외산 상용 분석기 비교

	오류 유형				총계
	Buffer Overrun	Null Pointer	Memory Leak	Uninitialized variable	
테스트오류 건수	45	16	13	19	93(100%)
행안부 분석기	45	16	9	7	77(82%)
외산 상용 분석기	7	3	5	7	22(23%)

다음에서는 이러한 분석 결과 차이가 어디에서 발생하는지 예제 프로그램을 통해 간단히 살펴보도록 하자.

그림 2는 3개의 함수로 이루어진 간단한 프로그램이다. Buffer_overrun_03_02_05()에서 정의된 buffer 크기 (BUFF_SIZE) 보다 큰 값으로 buffer에 접근하고 있다 (BUFF_SIZE -1이 최대 값). 사람의 눈으로 살펴보면 단지 호출 관계에 따른 추적만으로 분석이 가능할 수 있으나, 해당 오류 검출을 위해서는 변수 assign으로 인한 주소 값 전달, 구조체 주소 값 전달, 함수 호출에 의한 주소 값 전달, 변수의 실제 값 분석, call chain tracking 등의 분석기 디자인 요구사항을 만족해야 한다. 이 밖에도 프로그램 분석을 어렵게 하는 요인으로는 loop 분석, path 분석 등이 있다(그림 3 참조). 실제 매우 간단하게 보이나, 일반적인 상용 보안 취약점 분석 도구에서는 만족시키지 못하는 기능이다. 일반적인 보안 취약점 분석 도구에서는 선택하기 힘든 값 비싼(많은 분석 리소스를 요구하는) 분석 기능이기 때문이다.

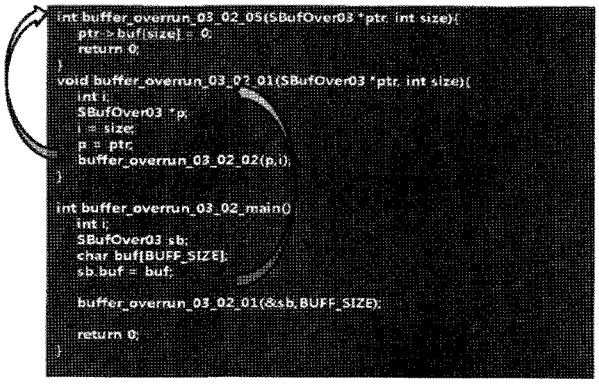


그림 2 Buffer overflow 프로그램 예제 1

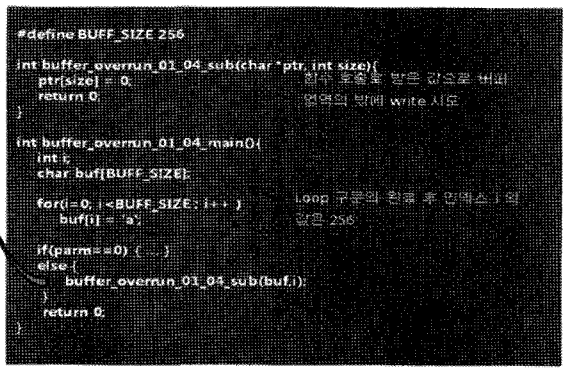


그림 3 Buffer overflow 프로그램 예제 2

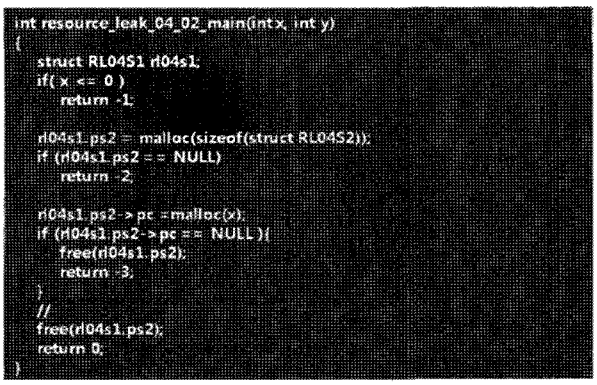


그림 4 Memory leak 프로그램 예제

또 다른 예제로 memory leak 검출을 살펴보자. Memory leak은 종종 보안 취약점에서 간과되기도 하지만, 실제 운영의 측면에서는 가장 중시되는 부분이다. Memory leak은 정상 작동을 방해하여 시스템의 성능 저하, 시스템 다운을 야기하며, 이는 Dos 공격에 대한 보안 취약점이 된다.

그림 4의 코드는 2단계의 메모리 동적 할당을 수행한다. 처음에는 struct 변수인 rl04s1의 ps2 field에 동적 할당을 한 후 할당 실패에 대한 예외 처리를 한다. 그 후 다시 rl04s1.ps2-의 pointer field인 rl04s1.ps2->pc에 동적 할당을 하고, 실패 시 부모 struct인 rl04s1.ps2까지 해지한다. 그러나 2번 모두 성공하면, 마지막의 free(rl04s1.ps2) 구문을 통해 rl04s1.ps2만 해지하게 되며, 자식 struct의 field인 rl04s1.ps2->pc 영역은 해지 되지 않기 때문에 memory leak을 야기한다. 이와 같은 구조체 field 분석은 많은 분석 resource를 요구하는 것으로 memory leak 분석의 어려움 중의 하나이다.

4. 결 론

지금까지 runtime 오류 정적 분석기 비교를 통하여 정적 분석기 설계 시 요구되는 내용들을 간략히 알아보았다. 현실의 복잡하고 다양한 프로그램 속에 내재된 runtime 오류의 정적 분석은 사실 많은 기술적 요소를 담고 있으며, 그 구현 방법에 따라 다른 분석 결과를 보여 준다. 안전한 프로그램 작성을 위해서 이러한 분석기 간의 차이점을 이해하는 것은 중요하며, 보다 정밀한 분석기 개발을 위해 많은 관심과 노력이 필요한 때이다.

(별첨) 정적 분석기 상세 비교 내역

Buffer Overflow

배열의 확보대상	Stack	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○											
	Heap																											○	○	○	○	○	○	○	○	○	○	○
함수내/함수간	함수내					○				○					○				○				○				○					○					○	
	함수간 Nest 1	○					○	○	○		○					○	○	○		○				○	○	○		○					○	○	○	○		
	함수간 Nest3				○								○									○								○								
	함수간 Nest5		○	○								○	○							○	○							○	○									
Buffer	단순한 배열	○	○	○	○	○	○	○	○	○																		○	○	○	○	○	○	○	○	○	○
	구조체 멤버									○	○	○	○	○	○	○	○	○																				
	배열의 선두를 나타내는 Pointer																	○	○	○	○	○	○	○	○	○												
결합 패턴	단순대입에 의한 갱신	○	○	○	○			○	○	○	○	○	○	○			○	○	○	○	○	○	○	○			○	○	○	○	○	○	○			○	○	○
	for문 뒤의 off-by-one					○	○							○	○									○	○							○	○					
첨가글자	정수																																					
	변수(정수를 대	○	○	○	○					○	○	○	○					○	○	○	○							○	○	○	○							
	변수(if문에서 값 한정)								○								○								○													○
	변수(자변수의 복귀값)							○	○						○	○							○	○									○	○				
	변수 (연산의 결					○	○						○	○							○	○									○	○					

Tool명	검출율																																				
신규 개발 엔진	100%	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
외산 상용 도구	16%	○				○				○				○				○																				○

Null Pointer 참조

함수내/함수간	함수내	○				○	○	○	○				○	○	○	○	○	
	함수간 (Nest1)		○		○					○		○						
	함수간 (Nest3)			○							○							
	함수간 (Nest 5)																	
(변수)	Pointer	○	○	○	○													
	Double Pointer					○	○	○	○	○	○	○						
	Triple Pointer												○	○	○	○	○	
	Pointer의 배열																	
	구조체의 멤버																	
Null 설정 패턴	(Null Macro 대입)		○	○	○	○	○	○		○	○	○	○	○	○			
	(Null값의 별도 변수를 대입)																	
	if문 판정에 의한 값의 한	○							○								○	○
	(Switch문 판정에 의한 값의 한정)																	

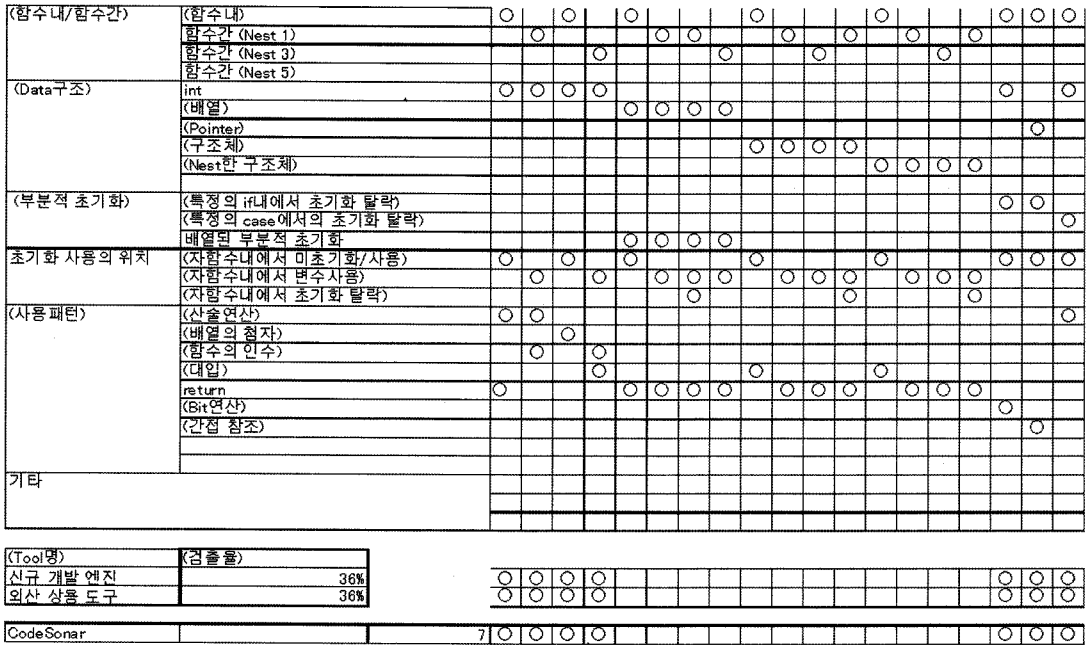
(Tool명)	(검출율)																	
신규 개발 엔지	100%	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
외산 상용 도구	18%	○				○							○					

Resource Leak

함수내/함수간	함수내		○		○		○	○	○		○				○			
	함수간 (Nest1)			○		○				○		○	○	○				
	함수간 (Nest3)																	
	함수간 (Nest 5)																	
데이타구조	Pointer		○				○				○	○	○	○				
	Double Pointer			○														
	Triple Pointer																	
	Pointer의 배열								○	○								
	구조체의 멤버 Pointer				○	○				○					○			
	(Double Pointer)																	
	(Triple Pointer)																	
	(Pointer의 배열)																	
	(복잡한 Data구조)														○			
(Scope)	(Local 변수)		○	○	○	○		○	○	○	○	○		○	○			
	(Global 변수)						○						○					
address copy	(없음)				○	○	○	○	○		○				○			
	(통상의 대입)	○																
	(실인수에서의 인도) (부모 ---> 자식)									○								
	(자식 ---> 부모)		○									○		○				
	(복귀값에서의 인도)											○						
	(Global변수 경유)												○					
leak pattern	(해방하지 않고 Return)		○	○	○	○		○	○	○	○	○			○			
	(해방하지 않고 Scope를 이탈한다)													○				
	(Pointer변수의 표시)	○					○						○					
	(Loop내에서의 표시)																	
	(Loop내에서의 해방)								○	○		○						
기타																		

(Tool명)	(검출율)																	
신규 개발 엔지	69%	○	○	○			○		○	○	○		○	○				
외산 상용 도구	38%	○							○	○	○		○					

미초기화 변수



참고문헌

[1] Robert C. Seacord: Secure Coding in C and C++, published by Person Education, Inc., publishing as Addison Wesley Professional, Copyright ⓒ 2006

[2] Kwangkeun Yi, Jaewhang Kim, Yungbum Jung: Design and Implementation of Static Program Analyzer Finding All Buffer Overrun Errors in C Programs), 정보과학회 2006.5

[3] Yungbum Jung: Practical Memory Leak Detector Based on Parameterized Procedural Summaries, 2007

[4] Jaeho Shin: An abstract Interpretation with the Interval Domain for C-like Programs, 2006.7

[5] Yungbum Jung, Kwangkeun Yi: Practical Memory Leak Detector Based on Parameterized Procedural Summaries, ISMM 08, 2008.6

안 성 민

2001 KAIST 수학과(학사)
2007 서울대학교 수학과(석사)
2007~2008 ㈜ITPLUS CM 개발팀
2008~현재 ㈜파수닷컴 PA개발팀
E-mail : asm@fasoo.com

진 민 식

2001 서울대학교 컴퓨터공학(학사)
2007 서울대학교 컴퓨터공학(석사)
2005~2006 국가보안연구소 연구원
2007~현재 (주)파수닷컴 PA개발팀 팀장
E-mail : msjin@fasoo.com

조 규 진

1988 연세대학교 전기전자공학(학사)
1991 Stevens Institute of Technology 전기전자공학(석사)
1995 Rutgers-The state university of New Jersey 전기전자공학(박사)
2000~2004 ㈜시큐어소프트 CTO
2004~2007 ㈜다산네트웍스 기술기획/마케팅
2007~현재 ㈜파수닷컴 PA (Program Analysis) 사업본부장
E-mail : kyucho@fasoo.com