심볼릭 링크 공격에 취약한 코드 검출 기법

주성용○ 조장우
동아대학교 컴퓨터공학과
jheaven1@donga.ac.kr○, jwjo@dau.ac.kr

Detecting Code Vulnerable to Symbolic Link-Exploit

Seongyong Joo○ Jang-Wu Jo
Dept. of Computer Engineering, Dong-a University

1. 서 론

신뢰성 있는 소프트웨어를 위해서 보안 문제는 중요하다. 컴퓨터 시스템의 단-대-단 (end-to-end) 행위는 기밀성과 같은 중요한 보안 정책을 만족해야 하는데, 현재 표준 보안 정책들은 이런 본질적인 문제에 대해서 보장하지 못하고 있다[1]. 이와 같은 소프트웨어 보안 문제를 해결하기 위해서 최근 프로그래밍 언어에서 사용하는 기법들을 이용한 연구가 활발하게 진행되고 있다[1,2,3,4,5,6,7]. 보안 문제를 해결하기 위해서 이용되는 프로그래밍 언어에서 사용하는 기법들로는 타입 시스템과 정보 흐름 분석을 위한 정적 소스 코드 분석 기법들이 있다. 본 논문에서는 타입 한정자를 이용해서 잘 알려진 공격 기법인 심볼릭 링크 공격[8,9,10,11]에 취약한 코드를 탐색하는 기법을 제안한다.

2. 심볼릭 링크 공격 및 취약코드 분석

심볼릭 링크 공격은 경쟁 조건 공격(race condition attack)의 한 종류로서, Unix나 Linux에서 제공되는 심볼릭 링크를 이용한 공격이다[8,9]. C로 작성된 프로그램에서 임시파일을 생성하고, 생성한 임시 파일을 읽거나 쓰는 경우가 있다. 이 같은 연산을 수행할 때 프로세스 경합이 발생할 수 있는데 이런 특징을 이용하는 것이 경쟁 조건 공격이다. 심볼릭 링크 공격은 프로그램에서 `fopen()` 함수를 이용해서 생성하는 임시 파일을 공격 대상으로 한다.

심볼릭 링크 공격을 해결하기 위한 기존의 방법으로 임시 파일 사용 전 동일한 파일이 있는지 확인 후 사용하는 방법, 신뢰하는 파일 이름을 사용하는 방법, 공유 속성을 제거하는 방법 등이 있다[9]. 그러나 심볼릭 링크 공격을 회피하기 위한 기존의 기법들을 이용하기 위해서는 프로그램에서 파일 조작 연산을 하는 구문은 모두 수정되어야 하는 작업이 필요하다. 이것은 프로그래머에게는 과중한 부담이며 또한 임시 파일을 사용하는 모든 코드가 심볼릭 링크 공격의 대상이 아니다. 본 논문에서는 프로그래머의 부담을 줄이기 위해서 심볼릭 링크 공격의 취약점을 분석하고, 취약점으로 분석된 코드에 한해서 심볼릭 링크를 회피하기 위한 기존의 기법을 적용할 수 있도록 한다.

이를 위해서 본 논문에서는 심볼릭 링크 공격에 취약한 구문을 정의하였다. 심볼릭 링크 공격의 전제 조건은 링크할 대상 파일의 이름을 정확하게 알아야 한다는 것이다. 파일 이름은 공개된 소스 코드를 이용해서 알 수도 있고, 프로그램 내에서 파일 이름을 문자열 상수 형태로 사용하는 경우, 이진 코드 편집기를 이용해서 쉽게 알아낼 수도 있다. 그러나 실행 시 인수로 전달되는 파일명은 실행 이미지를 통해서 예측하기는 어렵기 때문에, 인수로 전달된 파일명을 사용하는 경우는 심볼릭 링크 공격의 대상이 아니다.

본 논문에서는 심볼릭 링크 공격의 취약점을 판별하기 위해서, 함수 `fopen()`의 첫 번째 인수로 사용되는 값이 가변적이거나 사용자 입력으로부터 전파된 것인지 내부에서 직접적으로 선언된 문자열 상수나 이 값으로부터 전파된 것인지를 식별한다. 만일 이 값이 내부에서 선언된 문자열 상수라면 오류를 보고한다. 이를 위해서 본 논문에서는 타입 시스템을 사용한다.

본 논문에서는 외부로 쉽게 노출될 수 있는 문자열

상수나 문자열 변수는 타입 한정자 unguarded로 선언하며, 가변적이거나 사용자 입력으로부터 전달된 문자열 포인터 변수는 guarded로 선언한다. 그리고 문자열 상수의 기본 속성은 unguarded로 정의한다.

타입 한정자 guarded와 함께 선언된 변수에 unguarded로 선언된 값을 배정하는 것은 허용되지 않지만, unguarded로 선언된 변수에 guarded로 선언된 값을 배정하는 것은 허용된다. 그러므로 unguarded는 guarded의 수퍼 타입이 된다. 타입 추론 시 수퍼 타입을 서브 타입에 배정하는 구문을 발견한다면 오류를 보고한다. 이것은 심볼릭 링크의 취약점임을 의미한다.

타입 검사를 위해서는 모든 표현식에 타입 한정자를 제공해야한다. 그러나 이것 역시 프로그래머에게 과중한 부담이기 때문에 본 논문는 타입 추론을 이용해서 프로그래머가 타입 한정자를 선언하는 부담을 줄였다.

타입 추론을 위해서 표준 라이브러리들은 타입 한정자와 함께 선언된다. 응용 프로그램 분석 시 타입 한정자와 함께 선언된 표준 라이브러리들을 기반으로 타입 제약을 생성한다. 만일 이 제약을 만족하지 못하는 표현식이 발견된다면 오류를 보고한다. 제약 기반의 타입 추론 기법을 심볼릭 링크 공격의 취약점을 찾기 위한 알고리즘에 적용함으로써 응용 프로그램 작성자는 별도의 타입 한정자 선언 없이 제안된 방법을 자신의 프로그램에 적용할 수 있다. 또한 기존의 작성된 프로그램에 대해서도 프로그램의 변경 없이 동일한 알고리즘을 적용할 수 있다.

6. 결론 및 향후 과제

본 논문에서는 타입 한정자를 이용한 심볼릭 링크 공격의 취약점 분석 기법을 제안하였다. 본 논문에서 제안한 방법의 장점은 기존에 작성된 프로그램의 코드를 크게 변경하지 않고 심볼릭 링크 공격의 취약점을 분석할 수 있다. 그리고 심볼릭 링크 공격을 회피하기 위한 기존의 방법은 파일과 관련된 연산을 수행하는 모든 코드를 수정해야 하지만, 제안한 방법으로 실제 심볼릭 링크 공격의 대상이 되는 부분을 검출한 후 기존 방법을 적용하면 코드 수정의 부담을 줄일 수 있다.

본 연구의 문제점은 한 번 추론된 타입은 프로그램의 동일한 영역에서는 동일한 타입을 갖는 것으로 판단하기 때문에 근사가 발생한다. 그러나 실제 프로그램 문자열 변수의 속성은 변경될 수 있기 때문에 정확한 추론을 위해서는 문맥 의존적(flow-sensitive)인 추론 방법이 필요하다. 이것은 향후 과제로 남겨둔다.

참고문헌

[1] Andrei Sabelfeld, Andrew C. Myers, "Language-Based Information-Flow Security". IEEE Journal on selected areas in communications, Vol. 21, No.1, January 2003.

[2] Kyung-Goo Doh, Seung Cheol Shin, "Detection of Information Leak by Data Flow Analysis", *ACM SIGPLAN Notices, Volume 37, Issue 8*, pages 66-71, 2002.

[3] Jeffrey S. Foster, Manuel Fähndrich, Alexander Aiken, "A Theory of Type Qualifiers", *ACM SIGPLAN Notices, Conference on Programming language design and implementation PLDI '99*, *Volume 34, Issue 5*, pages 192-203, 1999.

[4] Jeffrey S. Foster, Tachio Terauchi, Alex Aiken, "Flow-sensitive Type Qualifiers", *ACM SIGPLAN Notices, Conference on Programming language design and implementation PLDI '02, Volume 37, Issue 5*, pages 1-12, 2002.

[5] Jeffrey S. Foster, Robert Johnson, John Kodumal, Alex Aiken, "Flow-insenstive Type Qualifiers", *ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 28, Issue 6*, pages 1035-1087, 2006.

[6] Umesh Shankar, Kunal Talwar, Jeffrey S. Foster, and David Wagner, "Detecting Format-String Vulnerabilities with Type Qualifiers', *10th USENIX Security Symposium*, pages 201-218, 2001.

[7] Jeffrey S. Foster, *Type Qualifiers: Lightweight Specifications to Improve Software Quality*, Ph.D. thesis. University of California, Berkeley, 2002.

[8] John Viega, Gary McGraw, *Building Secure Software*, pages 209-265, 2001.

[9] Robert C. Seacord, *Secure Coding in C and C++ (한국어판)*, Addison-Wesley, pages 277-305, 2006.

[10] Umesh Shankar, Kunal Talwar, Jeffrey S. Foster, and David Wagner, "Detecting String Vulnerabilities with Type Qualifiers", *10th USENIX Security Symposium*, pages 201-218, 2001.

[11] 양대일, *정보 보안 개론과 실습*, 한빛 미디어, pages 227-234, 2004.

[12] Frits Henglein, Jakob Rehof, "The Complexity of Subtype Entailment for Simple Types", *In Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 352-361, 1997.