바이너리 조작을 통한 정적 스택 보호 시 발생하는 명령어 밀림현상 방지 기법

이영림○, 김영필, 유혁
고려대학교
{yrlee○, ypkim, hxy}@os.korea.ac.kr

Instruction-corruption-less binary modification mechanism for static stack protection

Youngrim Lee○, Youngpil Kim, Chuck Yoo
Dept. of Computer Science & Engineering, Korea University

1. 서 론

센서환경에서 하드웨어를 제어하고 그 위에서 동작하는 응용프로그램의 관리를 위해서 운영체제의 존재는 필수적이다. 현재 개발되어 있는 센서 노드를 위한 운영체제는 프로그램 방식에 따라 이벤트 드리븐과 스레드 기반 방식으로 분류 할 수 있다. 스레드 기반방식이 동시성을 제공해 줄 수 있는 장점 때문에 각광 받고 있다.[1]

멀티 스레드 시스템을 지원하는 센서 운영체제에서는 다중 스레드들이 동작하는 스택 영역의 보호와 동적으로 메모리 블록을 할당 받거나 반납하여 메모리 공간 사용의 효율을 높이는 메모리 관리가 필요하다.

하지만 현재 많은 멀티 스레드 센서 운영체제에서는 이를 지원하지 못하고 있다. MANTIS[2], Nano Qplus[3]와 같은 멀티 스레드 센서 운영체제는 메모리 제약 때문에 스레드간의 스택을 공유하는데 대부분 대상 플랫폼에 하드웨어 MMU가 없어서 스택 보호가 이루어지기 어렵다. 이로 인해 센서 노드에서 동작하고 있는 프로그램들이 오동작하거나 시스템이 다운되는 문제가 발생한다.

위와 같은 문제는 스택 보호 기능의 추가로 해결 할 수 있다. 스택 보호 기능을 추가하는 정적인 방법에는 바이너리 이미지를 수정하는 방법이 있다. 바이너리 이미지에서 스택 영역에 접근하는 모든 명령어를 스택 보호가 가능한 스택 관리 모듈을 호출하는 명령어로 덮어쓰는 방법이다. 이 방법은 첫 번째 방법에서 발생한 단점을 가지지 않는다. 또한 커널 개발의 유연성을 줄 수 있다. 하지만 이 방법은 스택 영역에 접근하는 명령어들과 스택 관리 모듈로 분기하는 명령어들 간의 명령어 길이 차이에 의한 명령어 밀림현상이 발생한다. 본 논문에서는 밀림현상을 발생시키지 않고 원하는 모듈을 호출하는 알고리즘을 설계하고, 실제 스레드기반 운영체제에서 스택 보호를 위해 제안된 Post-Compile Processing Tool[4]에 이 기법을 적용한다.

2. 밀림 현상 방지 알고리즘

명령어 밀림현상이란 명령어들로 이루어진 바이너리에 새로운 명령어를 삽입할 때 이후의 명령어들의 메모리상의 위치가 밀리는 현상을 말한다. 이런 밀림현상에 의해 밀리는 명령어 중에 무조건 분기 명령어(unconditional branch), 조건 분기 명령어(conditional branch)가 있다면, 이 명령어의 오퍼랜드인 목표주소 값이 원래 정해진 곳이 아니라 밀림 현상에 의해서 엉뚱한 곳의 값을 가지고 있는 것으로 되는 문제가 발생한다. 이를 해결하기 위해서는 명령어 삽입이 아닌 덮어쓰기(overwriting)가 필요하다. 그러나 덮어 쓰고자 하는 명령어가 더 길면 이후의 명령어가 손상(corruption)된다. 대부분의 대상 플랫폼은 가변 길이 명령어 집합을 가지므로 이 문제는 피할 수 없다. 예를 들어 Tiny OS[5], MANTIS, Nano Qplus의 대상 플랫폼인 ATmega 128은 기본적으로 RISC구조로 2 바이트(byte) 명령어 길이를 가진다.[6] 그러나 예외적으로 분기 명령어는 4바이트 명령어 길이를 가진다. 본 논문에서는 덮어쓰기로 밀림 현상을 해결하고 이로 인한 손상을 막기 위해서 여러 번의 short call을 사용하여 추가된 임의의 모듈에 도달하는 방법을 제안한다.

2.1 short call을 사용한 명령어 밀림현상 방지 알고리즘

Short call이 작은 명령어 길이를 가지고 있기 때문에 short call 명령어로 덮어쓰기를 한다면 덮어쓰여진 명령어를 제외한 다른 명령어에 영향을 주지 않고, 바이너리 수정하는 것이 가능하다. 구체적인 알고리즘을 설명하기 위

해서는 몇 가지 용어가 정의 되어야 한다.

- 목표 명령어 : 바이너리에 원래 있던 명령어. 추가된 임의 모듈을 호출하는 명령어로 덮어써지는 명령어.
- Primary 분기 명령어 : 목표 명령어를 추가된 임의 모듈로 분기하기 위해 덮어쓰는 명령어. short call 명령어
- 보조 Victim 명령어 : 바이너리에 원래 있던 명령어로써 징검다리역할을 하는 short call명령어에 의해 덮어써지는 명령어.
- 보조 분기 명령어 : 징검다리 역할을 하는 short call명령어. 보조 Victim 명령어를 덮어쓰는 명령어

Short call을 이용해서 전체 도달 범위를 가지는 jump와 같이 역할을 수행하기 위해서는 목표 명령어를 Primary 분기 명령어(short call 명령어)로 바꾸고 도달 범위 안에 가장 멀리에 있는 보조 Victim 명령어를 보조 분기 명령어로 바꾼다. 추가된 임의의 모듈에 도달할 때까지 반복적으로 보조 Victim 명령어를 보조 분기 명령어로 바꾸어 간다. 보조 Victim 명령어는 INSTRUCTION 테이블에 유지되고 추가된 임의의 모듈에서 관리된다.

보조 Victim 명령어를 원래 실행 흐름(flow)과 같게 실행하기 위해서는 보조 Victim 명령어에서 시작이 되어 추가된 임의의 모듈로 왔는지 아니면 Primary 분기 명령어(원래 추가된 모듈을 호출하는 명령어)에서 시작 되어 추가된 임의의 모듈로 왔는지 구별이 필요하다. 즉 추가된 임의의 모듈로 오게 한 시발점이 되는 명령어를 찾아야 한다.

이를 위해 본 논문에서는 스택을 사용한다. Short call을 사용하기 때문에 복귀 주소(return address)는 스택에 있다. 이 복귀주소들을 보고 시발점이 된 명령어를 구별한다. 원래 추가된 임의 모듈을 호출하기 위한 명령어에서 시작되었는지, 중간에 징검다리 역할을 하는 명령어에서 시작되었는지를 구분한다. 구분의 결과, Primary 분기 명령어에서 시작되었다면, 모듈을 수행하고, 보조 Victim명령어에서 시작되었다면, 원래 명령어를 실행하고 최초의 실행 흐름으로 돌아간다. 이 방법을 통해 추가된 메모리 보호 래퍼 루틴을 명령어 밀림현상 없이 호출하는 것이 가능하다.

3. 결론

본 논문에서는 밀림현상을 발생시키지 않고 여러 번의 short call을 사용하여 추가된 임의의 모듈을 호출하는 알고리즘을 제안하였다. 실제 스레드 기반 운영체제에서 스택 보호를 위해 제안된 Post-Compile Processing Tool설계에 이 기법을 적용 하였다.

본 논문에서 제안한 알고리즘에 의해 수정된 바이너리는 재 컴파일 없이 바로 센서 노드에 올라갈 수 있는 장점을 가진다. 이는 개발의 유연성을 살릴 수 있고, 정적인 조작만을 가하므로 실행시간 오버 헤드가 적다. 또한 임베디드 환경과 같은 잦은 빌드를 요하는 구조에 적합하다. 본 논문에서 제안한 알고리즘을 사용하면 센서 노드의 소프트웨어 보안 패치와 소프트웨어적 유지 보수가 용이할 것이다.

참고 문헌

[1] Rob von Behren, Jeremy Condit and Eric Brewer, "Why Event Are A Bad Idea", the 9th Workshop on Hot Topics in Operating System, Lihue, Hawaii, USA, May 18-21, 2003

[2] H. Abrach , S. Bhatti , J. Carlson , H. Dai , J. Rose , A. Sheth , B. Shucker , J. Deng , R. Han, MANTIS: system support for multimodAl NeTworks of in-situ sensors, Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, September 19-19, 2003, San Diego, CA, USA

[3] Seungmin Park, Jin Won Kim, Kwangyong Lee, Kee-Young Shin, Daeyoung Kim, "Embedded Sensor Networked Operating System", Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06) pp. 117-124

[4] 이영림, 김영필, 유혁, "MMU가 없는 Thread기반 운영체제에서 스택 보호를 위한 메모리 관리 기법", 한국정보과학회 추계 학술대회, 2006년 10월

[5] LEVIS, P., MADDEN, S., GAY, D., POLASTRE, J., SZEWCZYK, R., WOO, A., BREWER, E., AND CULLER, D. The emergence of networking abstractions and techniques in tinyos. In Proceedings of the First Symposium on Networked Systems Design and Implementation (2004), USENIX Association, pp. 1-14.

[6] atmega128 datasheet, available : http://www.atmel.com/