부트로더 에뮬레이션 내 런타임 메모리 오류 탐지 기술 연구

명철우 [1], 이병영 [2]
[1] 서울대학교 전기정보공학부 박사과정
[2] 서울대학교 전기정보공학부 부교수

cwmyung@snu.ac.kr, byoungyoung@snu.ac.kr

A Study on Runtime Address Sanitizer for Bootloader Emulation

Cheolwoo Myung[1], Byoungyoung Lee[2]
[1,2] Dept. of Electrical and Computer Engineering, Seoul National University

요 약

메모리 오류는 소프트웨어 개발 과정에서 발생할 수 있는 가장 일반적이면서도 치명적인 문제 중 하나다. 이러한 문제를 효과적으로 탐지하고 수정하기 위해, 개발자들은 다양한 메모리 오류 탐지 도구를 활용한다. 그러나, 이 기술들은 소스 코드가 필요하다는 중대한 제약이 있다. 특히 임베디드 시스템의 개발 과정에서는 종종 소스 코드 대신 컴파일된 바이너리 형태로만 펌웨어가 제공되곤 한다. 이러한 배경을 바탕으로, 본 연구는 임베디드 환경에서 발생할 수 있는 메모리 오류를 실시간으로 탐지하기 위한 새로운 접근 방식을 제안한다. 이를 위해, Dynamic ASan 이라는 기술을 QEMU 가상화 기술에 적용함으로써, 메모리 접근 시 메모리 안정성을 지속적으로 검증하는 시스템을 구축하였다. 이러한 접근 방식은 임베디드 시스템의 안정성과 보안을 개선하는 데 중요한 기여를 할 수 있다.

1. 서론

메모리 오류는 소프트웨어 개발 과정에서 발생할 수 있는 가장 일반적이면서도 치명적인 문제 중 하나다. 이러한 문제를 효과적으로 탐지하고 수정하기 위해, 개발자들은 다양한 메모리 오류 탐지 도구를 활용한다. Address Sanitizer (ASan) [1]와 Memory Sanitizer (MSan) [2]는 이러한 도구들 중에서 특히 널리 사용되며, 바이너리 빌드 과정에서 컴파일러를 통해 메모리 접근 검사 로직을 소스 코드에 삽입한다. 이 로직은 프로그램이 실행될 때 메모리 접근이 타당한 범위 내에서 이루어지는지를 점검함으로써, 메모리 오류를 식별하고 해결하는 데 핵심적인 역할을 한다.

이러한 메모리 오류 탐지 기술은 특히 퍼징 (Fuzzing) 기술과 같은 자동화된 소프트웨어 테스팅 방법에서 중요한 역할 [3,4]을 한다. 퍼징은 소프트웨어에 예기치 않은 입력을 제공하고 예상치 못한 동작이나 오류를 감지하는 기술로, 메모리 오류 탐지 기능을 통해 효율적으로 버그를 찾아낼 수 있다. 이런 방식으로 메모리 오류 탐지 기술은 소프트웨어의 안정성을 크게 향상시킬 수 있으며, 다양한 분야에서 소프트웨어의 신뢰성을 확보하는 데 기여한다.

그러나, 이 기술들은 소스 코드가 필요하다는 중대한 제약이 있다. 바이너리만 존재하고 소스 코드가 없는 경우, 메모리 누수나 취약점을 탐지하기가 매우 어렵다. 이는 널 포인터 역참조와 같은 명백한 하드웨어 폴트를 일으키는 오류는 비교적 쉽게 탐지할 수 있지만, 스택 (stack) 이나 힙 (heap) 오버플로우, Use-after-free 와 같은 시간적 (temporal) 및 공간적 (spatial) 메모리 안정성 (memory safety) 위반은 소스 코드 없이는 탐지하기 어렵다는 것을 의미한다.

2. 임베디드 펌웨어의 메모리 오류 탐지의 필요성

임베디드 시스템은 현대 기술 생활의 근간을 이루며, 다양한 기기에서 핵심적인 역할을 한다. 이러한 임베디드 시스템의 심장부에 해당하는 펌웨어는, 기기의 안전성과 신뢰성을 결정짓는 중요한 요소다. 그

러나 임베디드 시스템의 개발 과정에서는 종종 소스 코드 대신 컴파일된 바이너리 형태로만 펌웨어가 제공되는 경우가 많다. 이는 메모리 오류 탐지에 있어서 중요한 도전 과제를 제시한다. 바이너리 형태로만 존재하는 펌웨어에서 메모리 오류를 탐지하기 위해서는 전통적인 소스 코드 기반 메모리 오류 탐지 도구를 사용할 수 없기 때문이다.

특히 임베디드 펌웨어의 경우, 이러한 문제는 더욱 심각하다. 임베디드 기기가 부팅될 때 최고 권한으로 실행되는 펌웨어 내의 취약점은 해커에게 시스템 전체를 위협할 수 있는 기회를 제공한다. 예를 들어, 삼성전자와 같은 대기업에서 제공하는 핸드폰 펌웨어는 다양한 제품에 걸쳐 인터넷 [5]을 통해 널리 배포되며, 이러한 펌웨어 내의 취약점은 사용자의 개인 정보와 같은 중요한 데이터를 위험에 빠뜨릴 수 있다.

이러한 상황에서는 새로운 메모리 오류 탐지 기술의 개발이 필수적이다. 바이너리 분석과 같은 기술은 소스 코드 없이도 메모리 오류를 탐지할 수 있는 가능성을 제공하지만, 여전히 이 분야에서는 많은 연구와 개발이 필요하다. 특히 임베디드 시스템의 보안과 안전성을 확보하기 위해서는 이러한 기술의 효율성과 정확성을 높이는 것이 중요하다.

3. 런타임 메모리 오류 탐지 기술

본 연구는 임베디드 환경에서 발생할 수 있는 메모리 오류를 실시간으로 탐지하기 위한 새로운 접근 방식을 제안한다. 특히 ARM 아키텍처를 기반으로 하는 임베디드 펌웨어에 초점을 맞추어, 이 분야에서의 기존 연구 [6]와 차별화된 접근 방식을 채택하였다. 본 연구는 삼성 핸드폰 부트로더 바이너리를 대상으로 하여, 펌웨어 내부 라이브러리를 활용한 메모리 오류 탐지 기술을 개발하였다. 이를 위해, Dynamic ASan 이라는 기술을 QEMU 가상화 기술 [7]에 적용하여, 메모리 접근이 이루어질 때마다 메모리 안정성을 검증할 수 있는 시스템을 구축하였다.

메모리 오류 탐지 기술의 핵심은 메모리 접근 시점에서의 안정성 검증에 있다. 본 연구에서는 Dynamic ASan 을 통해, QEMU 의 Tiny Code Generator (TCG)를 활용하여 메모리 접근 명령어가 실행될 때마다 메모리 안정성을 검사한다. 이 과정에서 메모리 할당자의 malloc() 및 free() 함수 호출을 추적하여, 함수의 진입점과 탈출점에서 추가적인 메모리 검사 로직을 삽입한다. 이를 위해 섀도우 메모리 (Shadow Memory) 와 레드 존 (Red Zone) 이라는 두 가지 중요한 개념을 도입하여, 메모리 할당 시 발생할 수 있는 오류를 예방하고 탐지한다. 섀도우 메모리는 실제 메모리 할당 영역을 모니터링하는 별도의 메모리 공간을 의미하며, 레드 존은 메모리 할당 영역의 양 끝에 추가적으로 할당되는 버퍼 영역으로, 메모리 오버플로우를 탐지하는 데 사용된다.

그러나 이러한 접근 방식은 기존의 부트로더 바이너리 내 메모리 할당자와의 호환성 문제를 발생시킨다. 특히, 레드 존을 위한 추가적인 메모리 공간 할당이 기존 할당자의 연속적인 메모리 할당 방식과 충돌하기 때문에, 본 연구에서는 소스 코드가 없는 상황에서도 적용 가능한 대안적인 방식을 모색하였다. 최종적으로, TCG 를 활용하여 부트로더의 기존 메모리 할당자를 수정하는 방식을 채택하였으며, 이를 통해 메모리 할당 과정에서 발생할 수 있는 오류를 실시간으로 탐지하고 대응할 수 있게 되었다.

본 연구에서 제안한 방식은 임베디드 펌웨어의 메모리 오류 탐지에 있어서 새로운 가능성을 열어주었다. malloc() 및 free() 함수의 사용 예를 들어, 부트로더 내 임의의 함수에서 메모리 할당 및 해제 과정을 추적하고, 메모리 할당 시 레드 존의 크기를 고려하여 실제 요청된 메모리 크기보다 더 큰 공간을 할당함으로써 메모리 안전성을 강화하였다. 또한, 메모리 해제 시에는 레드 존의 크기를 고려하여 실제 할당된 메모리 주소를 복원함으로써, 메모리 할당자와의 호환성 문제를 해결하였다. 이러한 접근 방식은 특히 소스 코드 없이 바이너리만 존재하는 임베디드 시스템 환경에서 메모리 오류를 효과적으로 탐지하고 대응하는 새로운 기술적 방법을 제시한다. 해당 기술에 대한 자세한 정보는 아래의 그림 1 로 표현한다.

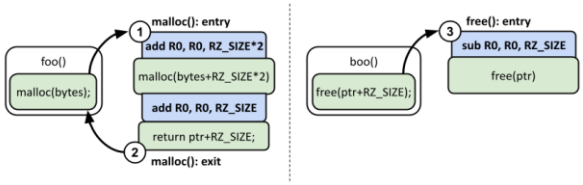

그림 1. TCG 기술을 활용한 ASan 메모리 할당자 설계

임의의 부트로더 함수 foo() 및 boo() 함수에서 malloc() 및 free() 함수를 실행하였다고 가정해보자. 먼저, malloc() 및 free() 함수에 입력되는 파라미터 정보를 살펴보도록 한다. malloc()의 경우, 할당할 메모리 바이트 크기를 인자로 설정하며, R0 레지스터에 저장된다. 그리고, 할당한 메모리의 주소를 리턴하게 되는데, R0 레지스터에 저장된다. free()의 경우, 할당을 해

제할 메모리의 주소를 인자로 설정하며, R0 레지스터에 저장된다. 해당 정보를 바탕으로, 그림 1 를 살펴보면, ① malloc() 함수의 entry point 에서 R0 에 레드존 사이즈를 더하여 실제보다 더 큰 메모리를 할당하도록 한다. 참고로, 레드 존은 메모리의 양끝 쪽, 두 곳에 위치하기 때문에, 레드 존 사이즈의 2 배를 더하게 된다. ② malloc() 함수의 exit point 에서는 리턴하는 메모리 주소가 레드 존을 가리키고 있기 때문에, 레드존 크기를 더해 주소를 리턴하도록 한다. 이를 통해, 부트로더의 일반적인 동작 내에서 레드존 메모리 접근을 피하도록 하였다. ③ free() 함수에서는 메모리 할당자가 실제로 할당했던 메모리 주소를 넘겨주기 위해, malloc() 함수 exit point 에서 더한 레드존 크기를 다시 빼주는 동작을 추가하였다.

4. 결론

ARM 기반 임베디드 펌웨어를 위한 런타임 메모리 오류 탐지 기술을 개발함으로써, 소스 코드 없는 바이너리 환경에서도 메모리 오류를 효과적으로 탐지할 수 있는 새로운 방법을 제시했다. Dynamic ASan 과 QEMU 를 통합하여 메모리 오류 검사를 실시간으로 수행함으로써, 메모리 관련 취약점을 사전에 파악하고 수정할 수 있는 가능성을 탐색했다. 이러한 접근 방식은 임베디드 시스템의 안정성과 보안을 개선하는 데 중요한 기여를 할 수 있다.

사사문구

이 논문은 2024 년도 삼성전자의 재원으로 “퍼징을 활용한 Chipset 및 firmware 보안 검증 기술” 과제의 지원을 받아 수행된 연구임.

참고문헌

[1] Serebryany, Konstantin, et al. "{AddressSanitizer}: A fast address sanity checker." 2012 USENIX annual technical conference (USENIX ATC 12). 2012.

[2] Memory Sanitizer. https://clang.llvm.org/docs/MemorySanitizer.html

[3] American Fuzzy Lop (AFL). https://lcamtuf.coredump.cx/afl/

[4] Andrea Fioraldi, Dominik Maier, Heiko Eißfeldt, and Marc Heuse. 2020. AFL++: Combining incremental steps of fuzzing research. In 14th USENIX Workshop on Offensive Technologies (WOOT’20)

[5] Samsung Mobile Firmware Download https://www.sammobile.com/firmwares/

[6] Fioraldi, Andrea, Daniele Cono D’Elia, and Leonardo Querzoni. "Fuzzing binaries for memory safety errors with QASan." 2020 IEEE Secure Development (SecDev). IEEE, 2020.

[7] QEMU Emulator. https://www.qemu.org/