ARM MTE를 이용한 메모리 보호 기술 연구 동향 분석

김경환[1], 강정환[2], 권동현[3]
[1]부산대학교 정보컴퓨터공학부 학부생
[2]부산대학교 정보융합공학과 박사과정
[3]부산대학교 정보컴퓨터공학부 교수 (교신저자)
{kyounghwankim, jeonghwan, kwondh}@pusan.ac.kr

A Study Trend on Memory Protection Techniques with ARM MTE

Kyoung-Hwan Kim[1], Jeong-Hwan Kang[2], Dong-Hyun Kwon[1]
[1]School of Computer Science and Engineering, Pusan National University
[2]Dept. of Information Convergence Engineering, Pusan National University

요 약

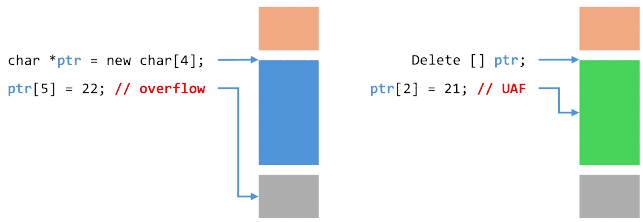
ARM MTE(Memory Tagging Extension)는 안전하지 않은 언어로 작성된 코드에 존재할 수 있는 메모리 취약점을 탐지하는 것을 목표로, ARMv8.5-A 아키텍처에 도입된 새로운 하드웨어 기능이다. MTE는 메모리 포인터에 대해 추가 메타데이터로 각 메모리 할당/해제 시에 태그를 지정한다. 런타임 시에 포인터와 메타데이터의 태그를 확인하는 절차를 거치게 되며, 이를 통해 가장 일반적인 메모리 버그의 원인이 되는 Use-After-Free, Buffer Overflow와 같은 취약점을 탐지할 수 있다. 본 논문에서는 MTE를 이용한 메모리 보호 기술 연구들의 동향을 분석하고, 해당 연구의 지속적인 필요성을 제시한다.

1. 서론

메모리 기반의 취약점은 컴퓨팅 시스템에 내포되어 있는 심각한 위협 중 하나이다. 메모리에 저장된 데이터를 손상시키거나, 제어 흐름을 탈취하여 공격자는 임의의 코드를 실행시키거나 권한을 승격해 컴퓨팅 시스템의 보안을 손상시킬 수 있다.

이러한 위협에 대응하기 위해 ARMv8.5-A 아키텍처부터 새로운 하드웨어 기능인 MTE(Memory Tagging Extension)가 도입되었다. ARM MTE의 동작 방식은 그림 1과 같다. MTE는 메모리 영역의 각 16바이트에 4비트 태그를 할당하고, 해당 태그를 포인터의 사용하지 않는 상위 비트에 저장한다. 이후 메모리 접근 연산이 일어날 때, 포인터의 태그와 메모리 영역에 할당된 태그를 비교하여 접근 여부를 결정한다. 이러한 방법을 활용해 Use-After-Free, Buffer Overflow와 같은 취약점을 탐지해낼 수 있고, 비슷한 원리로 Out of Bounds 등과 같은 취약점도 탐지할 수 있다. 또한, MTE는 하드웨어 기반으로 구현되었기 때문에 기존의 소프트웨어 기반의 메모리 오류 탐지 도구인 AddressSanitizer[1] 등과 비교할 때 성능적인 이점을 제공한다. 위와 같은 이유로 C/C++ 소프트웨어의 보안 개선을 위한 방법 중 하나로 촉망받고 있다.

본 논문에서는 MTE를 이용한 메모리 보호 기술 연구들의 동향을 분석하고, 해당 연구의 지속적인 필요성을 논한다.



(그림 1) ARM MTE의 동작 방식

2. MTE 관련 연구

ARM MTE가 공개되고 난 뒤, 이를 활용한 다양한 연구들이 진행되어 왔다. 그 중 MTE를 활용한 결정론적 메모리 보호 기법을 제안[2][3]하거나, C/C++ 바이너리에서 메모리 오류 탐지를 가속화하는 방법을 제안하는 연구[4]가 대표적이다. 추가적으로, MTE 기능의 보안 취약점을 발견한 연구[5]도 존재한다. 본 논문에서는 이러한 연구들을 분석한다.

MTE의 태그는 메모리 포인터의 상위 비트에 저장되기 때문에, 메모리 포인터를 덮어쓸 수 있는 공격자로부터 안전하지 않다. Color My World[2] 논문에서는 이러한 MTE의 확률적인 메모리 보호를 개선하기 위해 결정론적 메모리 보호 설계를 제안하여 개발 후에도 안전한 메모리 환경을 제공한다고 설명한다. 이를 위해 정적 분석을 통해 메모리 할당을 안전한 할당과 안전하지 않은 할당으로 분류한 뒤, 안전한 태그(예: 0b1100)와 안전하지 않은 태그(예: 0b0XXX)를 부여한다. 안전한 태그가 부여된 포인터만이 메모리에 접근할 수 있고, 안전하지 않은 태그가 부여된 메모리는 접근할 수 없도록 한다. 공격자가 태그를 위조하여 메모리에 접근하는 것을 방지하기 위해 안전하지 않은 포인터에서 접근이 발생하는 경우 해당 포인터의 상위 비트를 지워 태그 위조 방지를 구현한다. 저자들은 SPEC CPU 2017 벤치마크를 통해 성능을 평가한 결과, 런타임 오버헤드는 13.6%, 코드 사이즈 오버헤드는 21.7%로 상대적으로 낮은 오버헤드를 보였다.

ARM MTE를 활용해 메모리의 안전을 결정론적으로 보장하는 또 다른 연구로는 Sticky Tags[3]가 있다. Sticky Tags 논문에서는 Size Class라는 개념을 도입하여 메모리 영역의 태그는 미리 결정된 패턴으로 한 번만 초기화하게 되며, 이로 인해 메모리 영역의 태그 재할당이 일어나지 않아 굉장히 효율적이다. 메모리 할당 요청이 일어나면 Size Class별로 미리 초기화된 공간을 할당해준다. 해당 논문에서는 SPEC CPU2006 및 2017 벤치마크를 사용하여 StickyTags의 성능을 평가했다. 그 결과, 기존의 MTE 기반의 메모리 보호 기법보다 약 4% 이하의 성능 오버헤드로 더 높은 성능을 보였다.

MTE의 보안 취약점을 발견한 TikTag[5] 논문에서는, 추측 실행을 통해 임의의 메모리 주소의 MTE 태그를 유출할 수 있는 가젯을 식별했다. 논문의 저자들은 MTE 태그가 일치하지 않을 때 추측 실행이 중단될 수 있다는 점을 활용하여 태그 유출 템플릿을 설계했다. 유효한 주소와 태그를 포함하는 템플릿을 활용해 분기 예측기를 훈련하고, 태그를 유출하려는 포인터와 추측된 태그를 통해 학습된 분기 예측 템플릿을 실행해 추측 실행을 유발한다. 이후 메모리 접근 및 저장 명령어를 실행해 캐시 상태를 관찰해 태그를 유출해낸다. 추가적으로 저자들은 이러한 MTE의 취약점을 완화하기 위한 접근 방식 또한 제시했다. Google Pixel 8 장치에서 V8 Javascript 엔진, Chromium 애플리케이션에 대한 MTE 태그 유출 가젯을 실험한 결과 매우 높은 성공률로 태그를 유출할 수 있음을 보였다.

3. 결론 및 향후 연구 방향

ARM MTE를 기반으로 한 메모리 보호 기술 연구가 활발히 진행되고 있다. 기존의 연구들에서는 MTE가 가지고 있는 태그 충돌 문제(인접한 태그가 같은 태그를 가질 경우 등)나 태그의 제한된 비트 수(4비트)로 인해 한계가 명확하거나, 문제점이 발생할 수 있다. 또한, MTE가 가지고 있는 취약점을 악용해 공격자가 시스템에 대한 공격을 가할 수 있다. 이를 보완하기 위해 MTE에 대한 엄격한 정책을 기반으로, 제한된 태그 비트 수를 효율적으로 활용하는 결정론적인 메모리 보호 기술에 관한 더 많은 연구가 필요할 것이다.

사사문구

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 융합보안핵심인재양성사업의 연구 결과로 수행되었음 (IITP-2024-2022-0-01201).

참고문헌

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

[2] Liljestrand, Hans, et al. "Color My World: Deterministic Tagging for Memory Safety." arXiv preprint arXiv:2204.03781 (2022).

[3] Gorter, Floris, et al. "Sticky Tags: Efficient and Deterministic Spatial Memory Error Mitigation using Persistent Memory Tags." 2024 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024.

[4] Hager-Clukas, Andreas, and Konrad Hohentanner. "DMTI: Accelerating Memory Error Detection in Precompiled C/C++ Binaries with ARM Memory Tagging Extension." Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. 2024.

[5] Kim, Juhee, et al. "TikTag: Breaking ARM's Memory Tagging Extension with Speculative Execution." arXiv preprint arXiv:2406.08719 (2024).