예외 처리를 피하는 정의되지 않은 명령에 의한 하드웨어 트로이 목마의 구현 및 대응책 연구

공선희, 김한이, 이보선, 서태원, 유헌창
고려대학교 컴퓨터교육학과
e-mail : nickong@korea.ac.kr

A Study on Implementation and Countermeasure for Undefined Instruction Hardware Trojan evitable from exception handling

Sunhee Kong, Hanyee Kim, Bosun Lee, Taeweon Suh, and Heon Chang Yu
Dept. of Computer Science Education, Korea University

요 약

Undefined Instruction 하드웨어 Trojan 은 정의되지 않은 명령어가 명령어 버스를 통해 CPU 에 유입될 경우 발현되어 CPU 의 전반적인 기능을 마비시킬 수 있는 하드웨어 Trojan 이다. 일반적으로 대부분의 상용화된 CPU 는 Undefined Instruction 에 대한 예외 처리를 지원하는데, ARM 의 경우 파이프 라인의 실행 단계에서 Undefined Instruction 임을 판별한다. 본 연구에서는 파이프 라인의 명령어 추출단계에서 발현되어서 명령어 해독단계에는 다른 명령어를 전달 시킴으로써 Undefined Instruction 예외처리를 피할 수 있는 하드웨어 Trojan 을 설계하고, 이를 방지하는 대응책을 제안한다.

1. 서론

지난 1 년간 국내 모바일 시장에서는 N-스크린 시대가 본격화 되면서, 개인이 스마트폰 뿐만 아니라 테블릿 PC, 노트북을 복수로 소유하는 경우가 만연하였다. 저명한 시장조사 기관인 IDC 의 발표[1]에 의하면, 지난 2012 년 3 분기에만 1 억 8 천만여대의 스마트폰이 판매되었다.

현재 스마트기기 시장에서는 한 칩에 CPU 뿐만 아니라 주변 장치를 집적하는 시스템온칩 (System on Chip) 프로세서가 내장되어 있는데, 시스템온칩의 대표적인 예로는 안드로이드 스마트폰에 내장되는 Exynos[2], Snapdragon[3] 가 있다. 최근에는 제대로 검증 받지 못한 중국산 저가 스마트기기의 공세마저 매서워지면서 시스템온칩 시장의 경쟁이 더욱 치열해지고 있다.

이와 같이 무수한 기기가 출시되고 있는 반면, 프로세서 및 주변 장치에 대한 보안 안정성 검증 연구는 미비한 실정이다. 실제 하드웨어 테스트 시에도 빠른 출시를 위해 기능위주의 검증만 이루어 질 뿐 보안을 위한 검증이 제대로 이루어지지 않고 있다.

하지만 사용자의 스마트 기기 내에 악의적인 목적으로 구현된 하드웨어가 존재한다면 그로 인한 피해 및 파급 효과는 상당할 것으로 예상된다.

이 논문은 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2012-008231)

설계자가 악의적인 목적으로 평소에는 올바른 동작을 하다가 특정발현 조건이 되면 올바른 동작을 하지 않도록 설계한 하드웨어를 하드웨어 Trojan 이라 한다. 최근과 같이 시스템온칩으로 프로세서를 개발하는 임베디드 분야에서는 각 하드웨어 구성요소마다 여러 설계자가 동원되기 때문에 어느 순간에 하드웨어 Trojan 이 내장되었는지 발견하는 것은 쉽지 않다. 실제로 미국 국방성 관리에 따르면, 유럽의 한 칩 제조사가 원격으로 제어 가능한 Kill switch 를 내장한 프로세서를 제조하고 있다고 한다[4].

이에 따라 하드웨어 Trojan 에 대한 대비책이 필요하며, 이를 위한 추가적인 하드웨어 구성, 또는 보다 정밀한 칩 검증 과정이 필요한 실정이다.

본 연구에서는 파이프라인 구조의 ARM CPU 가 가지고 있는 Undefined Instruction 예외처리 방법에 대한 취약점을 지적하고, 이를 공격할 수 있는 Undefined Instruction 하드웨어 Trojan 을 구현한다. 또한 Undefined Instruction 하드웨어 Trojan 으로 인한 시스템 오류를 방지할 수 있는 대응책으로 ARM 프로세서의 Undefined Instruction checker 을 제안한다.

본 논문의 구성으로 ARM 의 Undefined Instruction 의 예외 처리 방식, 실험 환경, 설계한 Undefined Instruction Trojan 의 특징을 설명하고 이를 방지하기 위한 대응책을 제안한다. 이와 더불어 결론에서 현재 연구의 한계점과 향후 연구 방향으로 마무리 한다.

2. ARM 의 Undefined Instruction 예외 처리

ARM 은 다양한 예외처리를 지원하며 지원하는 예외처리와 우선순위는 <표 1>과 같다.

<표 1>과 같이 ARM 에서는 Undefined Instruction 에 대한 예외처리 우선순위가 가장 낮다. 현재 ARM 의 Undefined Instruction 예외처리는 ARM 이나 Thumb 에 정의되지 않은 명령어가 파이프라인의 실행 단계에 이르렀으며, 다른 높은 우선순위의 예외상황이 발생하지 않았을 경우 예외처리를 실행한다 [5].

<표 1> ARM 의 예외처리 우선 순위

예외처리	우선 순위
Reset	1
Data abort	2
Fast Interrupt Request	3
Interrupt Request	4
Prefetch Abort	5
Software Interrupt	6
Undefined Instruction	6

ARM 에서 실행단계는 파이프 라인에서 세 번째 단계이며, 이에 따라 Undefined Instruction 이 발견되기까지는 두 사이클의 시간이 소모된다. 따라서 Undefined Instruction 을 이용한 하드웨어 Trojan 이 실행단계 이전인 명령어 추출단계나 명령어 해독단계에 구현되어 있다면, Undefined Instruction 에 의해 시스템의 정상적인 작동을 방해할 가능성이 있다.

3. 실험 환경

본 연구에서는 Undefined Instruction 하드웨어 Trojan 의 구현 및 대응책 실험을 위해 Trojan 설계와 그에 따른 대응책의 유효성을 검증할 수 있는 테스트 CPU 로 5 단계 파이프라인 구조를 갖는 ARM CPU 를 구현하였다. 구현한 ARM CPU 는 ARM 의 IRQ 인터럽트 및 소프트웨어 인터럽트, Undefined Instruction 예외처리를 발생 시킬 수 있으며 코드는 모두 Verilog 코드를 기반으로 작성되어 있다. 구현된 ARM CPU 에서 수행 가능한 명령어군은 <표 2>와 같다.

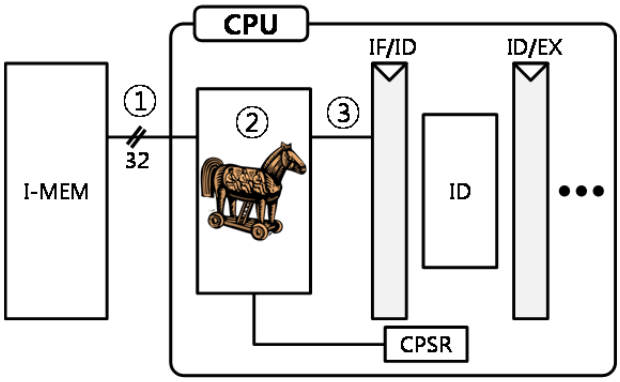
<표 2> 구현한 ARM CPU 의 수행 가능 명령어

Instruction
Data processing
Extra load/stores
Move immediate to status register
Load/store
Load/store multiple
Branch and branch with link
Software interrupt

실험에서 사용되는 테스트 CPU 에 Trojan 이식을 용이하게 하기 위해 Undefined Instruction 하드웨어 Trojan 은 Verilog 코드로 설계하였으며, Modelsim 시뮬레이션 툴을 활용하여 시간의 흐름에 따른 Trojan 및 CPU 의 내부 시그널들의 변이를 확인하였다.

4. 설계한 Undefined Instruction Trojan 의 특징

유입된 명령어의 파이프라인상의 전달은 (그림 1)과 같다. (그림 1)에서 Trojan 은 파이프라인 구조 중에 명령어 실행 단계의 앞 단계인 명령어 추출 단계 내에 위치한다.



(그림 1) CPU 에 내장된 Trojan 다이어그램

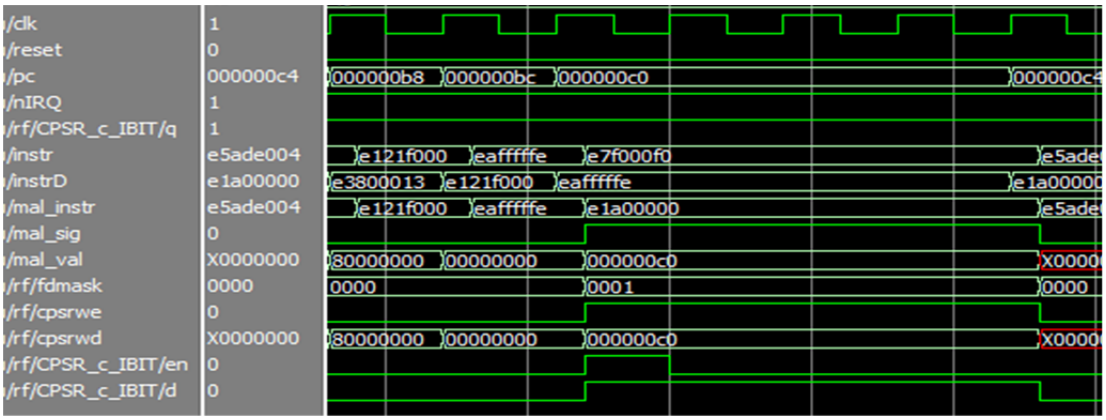
Trojan 은 명령어 메모리로부터 명령어를 받아 해당 명령어가 Trojan 을 발현시키는 Undefined Instruction 인지 아닌지를 판별한다. 만약 Trojan 을 발현시키는 Undefined Instruction 이라면, Trojan 은 ARM 의 현재 상태 레지스터인 CPSR 의 IRQ 와 FIQ 인터럽트 마스크를 1 로 변환하여 CPU 가 인터럽트를 처리하지 못하도록 조작한다. 또한 Undefined Instruction 으로 인한 예외처리가 발생하지 않도록 명령어 해독 단계에는 Undefined Instruction 대신 No operation 과 같이 시스템에 영향을 미치지 않는 명령어로 바꿔 전달하도록 한다.

실험에서는 Trojan 의 발현 동작을 확인하기 위해 Trojan 을 발현시키는 스위치 역할을 하는 Undefined Instruction (0xE7F000F0)을 코드에 추가하여 발현을 검증하였다.

<표 3> 파이프라인에서의 코드 흐름

5.	
0AC : E10F0000 ; 0B0 : E3C00092 ; 0B4 : E3800013 ; 0B8 : E121F000 ; 0BC : EAFFFFFE ; **0C0 : E7F000F0 ; #Trigger** 0C4 : E5ADE004 ; 0C8 : E59F0A6C ;	0AC : E10F0000 ; 0B0 : E3C00092 ; 0B4 : E3800013 ; 0B8 : E121F000 ; 0BC : EAFFFFFE ; **0C0 : E1A00000 ; #Nop** 0C4 : E5ADE004 ; 0C8 : E59F0A6C ;

<표 3>의 좌측 코드는 CPU 에 유입되어 명령어 추출단계에 전달되는 원본 코드의 순서이고, 우측 코드는 하드웨어 Trojan 에 의해 파이프라인의 명령어 해독단계에서 변경되어 받아들이는 코드의 순서이다. 만약 0xC0 번지의 0xE7F000F0 이 파이프라인의 명령어 추출단계에 유입되면 Undefined Instruction 하드웨어 Trojan 이 발현되어 ARM 의 CPSR 의 인터럽트 마스크들을 1 로 변경시킨다. 또한 명령어 추출단계에는 Undefined Instruction 예외처리가 발생하지 않도록 No



(그림 4) Trojan 이 구현된 시스템에서의 Waveform

operation 에 해당하는 0x21A00000 (MOV R0, R0) 으로 변경하여 전달한다.

<표 3>의 코드를 실행하였을 때, 시간의 흐름에 따른 Trojan 의 동작은 (그림 4)의 Waveform 과 같다.

(그림 4)는 설계한 Trojan 의 올바른 동작을 검증하기 위해 Modelsim 하드웨어 시뮬레이션 툴을 이용해 Undefined Instruction 을 CPU 에 유입한 부분이다. PC 값이 0x000000C0 에 해당하는 명령어는 Undefined Instruction 인 0xE7F000F0 이기 때문에 Trojan 이 발현되었음을 나타내는 시그널인 mal_sig 값이 1 로 변경되었다. 또한 CPSR 의 8 번째와 7 번째 비트에 해당하는 IRQ 마스크와 FIQ 마스크를 1 로 세팅하기 위해 Trojan 은 CPSR 의 write enable 시그널인 cpsrwe 값을 1 로 변경시켰고, CPSR 에 쓰여지는 값인 cpsrwd 은 0x000000C0 이 된다. (그림 4)의 instr 시그널은 명령어 추출단계에서의 Instruction 으로 Undefined Instruction 인 0xE7F000F0 의 값을 가지고 있으나, 명령어 해독 단계에서의 Instruction 시그널인 instrD 에서는 Trojan 의 발현으로 인해 No operation 인 0xE1A00000 으로 변환되어 전달됨을 알 수 있다.

6. 결론 및 향후 연구방향

본 연구의 실험과 같이 Undefined Instruction 예외처리 만으로는 완벽하게 Undefined Instruction Trojan 을 방지할 수 없다. 만약 시스템온칩 프로세서에서 CPU 내에 이와 유사한 악의적인 Trojan 이 집적되어 있다면 시스템 전체에 문제를 초래할 수 있으며, 보다 진보된 Trojan 의 내장을 통해 원격으로 모바일 디바이스를 자유자재로 조종하도록 만들 수도 있다. 따라서 설계과정 중 AMBA 버스와 같은 버스레벨단계에서 CPU 에 Undefined Instruction 이 유입되는 것 자체를 차단할 수 있는 명령어 해독기와 유사한 Undefined Instruction checker 를 만드는 것이 비교적 간단한 해결책이 될 수 있다.

현재 실험은 ARM CPU 를 기반으로 구현한 실험용 CPU 를 사용하여 RISC 프로세서 환경을 가정하였다. 32 비트 RISC 프로세서의 경우 가능한 명령어 인코딩 조합은 2^{32} 가지로 모든 가능한 명령어 조합에 대해 Undefined Instruction Trojan 이 발현되는지에 대한 테스트가 충분히 가능하다는 점에서 Undefined Instruction Trojan 에 대한 대응책 연구가 무의미해 보일 수도 있다. 하지만 CISC 프로세서의 경우, 특히 x86 의 경우 명령어가 최대 17 바이트에 이르는 복잡한 구조이기 때문에 모든 Undefined Instruction 의 조합에 대하여 Trojan 발현 여부를 테스트 해 보기가 어렵다. 본 연구에서는 RISC CPU 인 ARM 프로세서를 통한 실험을 하였다. 추후에는 복잡한 명령어 구조를 가진 CPU 를 통해 보다 효과적으로 다양한 하드웨어 Trojan 의 검출과 대응책에 대한 연구를 진행할 예정이다.

참고문헌

[1] IDC. Android Marks Fourth Anniversary since Launch with 75.0% Market Share in Third Quarter, According to IDC2012. Available from: http://www.businesswire.com/news/home/20121101006891/en/Android-Marks-Fourth-Anniversary-Launch-75.0-Market.

[2] Se-Hyun Y, Seogjun L, Jae Young L, Jeonglae C, Hoi-Jin L, Dongsik C, et al., editors. A 32nm high-k metal gate application processor with GHz multi-core CPU. Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International; 2012 19-23 Feb. 2012.

[3] Qualcomm. Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age2011. Available from: http://www.qualcomm.com/media/documents/files/snapdragon-s4-processors-system-on-chip-solutions-for-a-new-mobile-age.pdf.

[4] Adee S. The Hunt For The Kill Switch. Spectrum, IEEE. 2008;45(5):34-9.

[5] Sloss AN, Symes D, Wright C. ARM system developer's guide: designing and optimizing system software: Morgan Kaufmann; 2004.