RecurFI: 윈도우 바이너리에 대한 효율적 제어흐름 무결성 검사 시스템

홍 진 아†, 차 상 길‡, 김 광 조

정보보호대학원, 전산학부, 한국과학기술원

RecurFI: Practical Coarse-Grained CFI on Windows Binary Code

Jina Hong†, Sang Kil Cha‡,Kwangjo Kim

Graduate School of Information Security, School of Computing, KAIST

요 약

갈수록 진화하는 해킹 공격에 대응하기 위하여 등장한 제어흐름 무결성 기술은 강력한 소프트웨어 방어의 패러다임을 제시하였지만, 막중한 오버헤드로 인하여 실제로 적용하는 데에는 많은 한계가 있었다. 최근 들어 이러한 제어흐름 무결성 기술의 안전성과 성능을 절충하는 다양한 실용적 기술이 등장하였는데, 그 한 예로 MS가 2012년도에 공개한 EMET(Enhanced Mitigation Experience Toolkit)가 있다. 하지만 이를 위협하는 새로운 공격기법들 또한 등장하고 있으며, 이러한 공격을 효율적으로 방어하는 바이너리 기반의 시스템은 아직까지 실용화되지 못하였다. 본 논문에서는 제어흐름 무결성 기술을 우회하는 최신의 공격 기법을 효과적으로 방어할 수 있는 새로운 공격 탐지 시스템인 RecurFI를 제시한다. 우리는 RecurFI를 윈도우 익스플로러에 적용하였으며, SPEC 벤치마크를 통하여 그 효율성(평균 2%미만의 오버헤드)을 입증하였다.

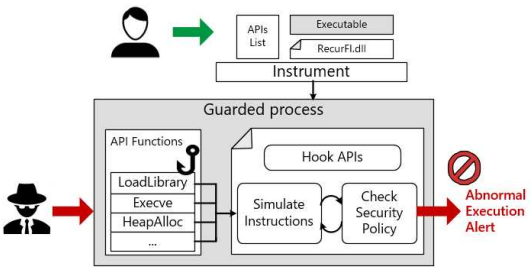
Ⅰ. 서 론

제어흐름 무결성(Control-Flow Integrity, 이하 CFI)기술 [1] 은 메모리 취약점에 의한 제어흐름 탈취(Control-Flow Hijack) 공격을 근본적으로 막고자 하는 방어기술 패러다임이다. 이는 프로그램을 사전에 허용된 제어흐름(Control Flow)으로만 실행되도록 제한하기 때문에 기존의 코드 주입(Code Injection)이나, 코드 재사용(Code-Reuse) 기반의 다양한 해킹 공격을 무력화할 수 있다. 그 대표적인 예가 ROP (Return-Oriented Programming) [2] 공격인데, ROP 공격의 경우 정상적인 프로그램이라면 발생할 수 없는 제어흐름을 통하여 가젯(ret 명령어로 끝나는 코드블락)을 실행하기 때문에 CFI 기술에 의하여 차단될 수 있다.

하지만 CFI기술을 실제 프로그램에 적용하는 데에는 두 가지의 근본적 한계점이 존재한다. 첫째, CFI 기술은 막중한 오버헤드를 발생시킨다. 왜냐하면 대상 프로그램의 모든 간접 분기문에 대한 계측(Instrumentation)이 필요하기 때문이다. 둘째, CFI기술은 바이너리코드에 직접 적용되기 어렵다. CFI기술을 적용하기 위해서는 제어흐름 그래프(Control-Flow Graph, 이하 CFG)가 필수적이지만, 바이너리코드로부터 완벽한 CFG를 복원하는 것은 불가능하기 때문이다.

최근에는 이러한 원천적인 문제점을 실용적으로 해결하는 완화된 CFI기술(Coarse-Grained CFI)이 점차 등장하고 있다. 예를 들어 GCC나 LLVM과 같은 컴파일러에서는 소스코드 기반의 완화된 CFI기술을 적용하고 있으며 [3], MS에서는 실용적인 제어흐름 무결성 기술인 ROPGuard를 EMET [4,5] 라는 도구를 통하여 공개하기도 하였다. 그 뿐 아니라 최근에는 하드웨어를 활용하여 완화된 CFI기술 [6]

† 주저자: jina3453@kaist.ac.kr
‡ 교신저자: sangkilc@kaist.ac.kr
이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원(No.B0717-16-0109, 바이너리 코드 분석을 통한 자동화된 역공학 및 취약점 탐지 기반기술 개발)과 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원(B0101-16-1270, 생체모방 알고리즘(Bio-Inspired Algorithm)을 활용한 통신 기술 연구)을 받아 수행된 연구임.



〔그림 1〕 RecurFI의 시스템 개요

을 효율적으로 시행하는 기술도 개발되었다.

이러한 완화된 CFI 기술의 등장은 안전성과 성능을 절충함으로써 새로운 보안의 취약점을 야기하였는데 그 대표적인 예가 MS의 EMET를 우회하는 공격기법이다 〔7,8〕. EMET는 특정 Windows API 호출에 대하여 ret 명령어가 실행될 때, 그 리턴 주소가 call 명령어 다음의 주소인지 확인하는 기술이다. 이는 쉐도우 스택 〔9〕 과 비슷하지만 완화된 형태의 방어를 수행한다. EMET을 우회하기 위하여 공격자는 call 명령어와 ret 명령어 사이에 다른 분기문이 존재하지 않는 가젯만을 활용하여 ROP 공격을 수행한다. 실제 이러한 공격은 인터넷 익스플로러(IE)등의 프로그램에 대한 최근의 공격코드에서 확인할 수 있다.

본 논문에서는 완화된 CFI기술을 우회하는 최신의 공격 기법들을 효과적으로 방어하는 새로운 개념의 시스템, RecurFI를 제시한다. RecurFI는 기존의 컴파일러 기반 CFI기술과 달리 소스코드 없이 바이너리코드에 직접 적용 가능하며, EMET과 달리 최신의 우회기법을 효과적으로 방어할 수 있다. 현재의 시스템은 윈도우를 기준으로 구현되었지만, 본 방식은 리눅스에도 동일하게 적용이 가능하다.

RecurFI의 아이디어는 직관적이다. 대부분의 ROP 공격은 여러 개의 가젯이 연결되어 동작하는데 이때 공격에 사용된 모든 가젯이 일반적인 제어흐름을 갖기는 어렵다. 따라서 API 호출이 발생한 콜 스택상의 모든 리턴 주소를 *재귀적*으로 분석하여 나가면서 각각의 리턴 주소가 call 명령어 다음의 유효한 주소를 가리키고 있는지를 검사한다. 만약 하나의 리턴 주소라도 유효하지 않은 명령어를 가리키면 스택이 변조되었다고 판단할 수 있고, RecurFI는 예외를 발생시켜 사용자에게 통보한다.

이어지는 절에서는 RecurFI 시스템의 알고리즘과 그 설계 방법에 대하여 설명하며, 다양한 실험을 통하여 RecurFI의 안전성과 효율성을 입증한다.

II. 시스템 설계

본 논문에서 제안하는 RecurFI의 시스템 개요는 〔그림1〕과 같다. 사용자는 관찰하고자 하는 API 함수의 목록을 대상 프로그램과 함께 RecurFI의 입력으로 전달하면, RecurFI는 주어진 프로그램을 실행하면서 목록에서 명시한 API 함수에 대한 계측을 실시한다. 계측된 API 호출마다 보안정책(Security Policy)을 검사하는 알고리즘이 수행되며, 보안정책에 위배되는 실행이 검출되면 프로그램에 예외를 발생하여 사용자에게 보고한다.

2.1. RecurFI 보안 정책

RecurFI의 CFI 보안 정책은 EMET의 정책보다 강력하지만, 본래의 CFI에 비하여 느슨한 노선을 따르기 때문에 최신의 공격기법들을 실용적으로 방어할 수 있다.

RecurFI는 크게 두 가지 보안 정책을 갖는다. 첫째, 스택에 저장된 리턴 주소는 실행 가능한 주소이어야 한다. 실행이 불가능한 메모리 주소가 리턴 주소로 저장이 되어있다는 것은 스택의 내용이 변질되었음을 의미하기 때문이다. 둘째, 리턴 주소의 명령어는 call 명령어에 이어서 나오는 명령어이어야 한다. 정상적인 리턴명령은 call 명령어가 수행된 직후의 주소 값으로 실행흐름이 이동하기 때문이다.

RecurFI의 완화된 보안 정책은 대부분의 고도화된 ROP공격을 막는 데 충분하나 공격자가 임의의 메모리영역을 임의의 값으로 조정할 수 있는 경우 register spilling등을 활용한 공격이 가능하다 〔8〕. 하지만 이러한 강력한 공격자 모델은 우리의 논문의 고려 대상이 아니다.

2.2. RecurFI 알고리즘

바이너리 기반의 완화된 CFI검사를 위하여 RecurFI는 〔그림 2〕의 알고리즘을 제안한다. SimulateRetn 함수는 계측된 API 호출마다 실행되며, 이때 프로그램 상태가 2.1절의 보안정책에 위배되는지 검사한다.

프로그램 상태는 계측된 API 함수가 반환된 이후의 명령어를 simulateInstruction 함수에서 정적으로 분석한 정보이다. 이 함수는 이 함수는 pc값과 스택포인터를 입력으로 받아 pc값에 해당하는 opcode를 분석하고 스택포인터를 추적하는 것이 주목적이다. pc값에 해당하는 opcode가 push, pop, leave 와 같이 스택포인터를 변경시키는 명령어라면, 명령어의 특성에 맞게 스택포인터를 계산한 후에 opcode, 다음에 살펴볼 pc값, 계산한 스택포인터를 반환한다. 이 과정은 opcode가 jmp 혹은 ret 명령어일 때까지 반복한다 (Line 3,4).

만약 opcode가 jmp 명령어라면, ROP 가젯을 이

Algorithm 1: RecurFI Enforcement Algorithm

```
1 Function simulateRetn(pc , esp):
2     pc ← esp + 4 ;
3     while true do
4         opcode , pc , esp ← simulateInstruction(pc , esp);
5         if opcode is return then
6             if checkPolicyViolation(esp) then return false;
7             else return simulateRetn(pc , esp);
8         else if opcode is jump then return true;
```

〔그림 2〕 RecurFI Enforcement 알고리즘

용한 실행흐름일 가능성이 적기 때문에 RecurFI는 해당 실행흐름을 정상으로 판단하고 SimulateRetn 함수를 종료한다 (line 8). 반면 opcode가 ret 명령어라면, checkPolicyViolation 함수가 리턴주소가 2.1절의 보안 정책 위반 여부를 검사한다 (Line 5-7). 그 결과 리턴주소가 유효하다면, simulateRetn 함수를 호출하여 콜 스택상의 리턴주소들을 *재귀적*으로 검사한다. 이 과정에서 만약 하나의 리턴 주소라도 유효하지 않은 주소를 가리키면 스택이 변조되었다고 판단하고, checkPolicyViolation 함수 내부에서 예외를 발생시켜 사용자에게 통보한다.

III. 실험 결과

본 논문에서는 RecurFI의 성능과 안전성을 측정하기 위하여 Intel Xeon E3-1231 v3 CPU, 2GB RAM, Microsoft Windows 7 SP1 32bit 환경에서 실험을 수행하였다.

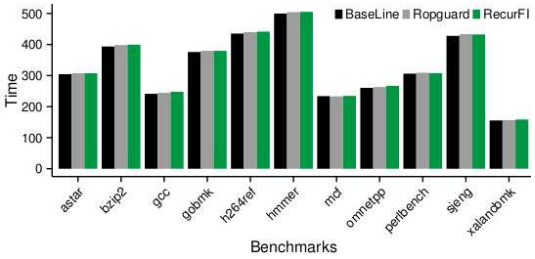
3.1. 성능 측정

RecurFI의 실용성을 평가하기 위위하여 RecurFI를 적용하였을 때와 적용하지 않았을 때의 성능을 측정하고 분석하였다. 이를 위하여 SPEC CINT 2006 벤치마크를 사용하였으며, [그림 3]은 그 실험결과를 보여준다. 그림 상의 검은 막대는 RecurFI가 적용되지 않은 상태의 벤치마크이고, 회색 막대는 EMET의 CFI기술인 ROPGuard가 적용된 벤치마크, 그리고 녹색 막대는 RecurFI가 적용된 벤치마크이다. 우리는 전체 실험을 동일한 환경에서 세 번을 반복 수행하였고, 총 9.2시간동안 진행하였다.

프로그램별로 RecurFI는 BaseLine 대비 최대 2.6%와 최소 0.5%의 시간 오버헤드를 가지며, 11개의 벤치마크에서 평균 오버헤드는 1.34%였다. 기존 탐지방식인 EMET의 평균 오버헤드 0.9%였다.

3.2. 안전성 측정

RecurFI의 안전성을 평가하기 위하여 RecurFI



〔그림 3〕 RecurFI의 수행 시간 비교

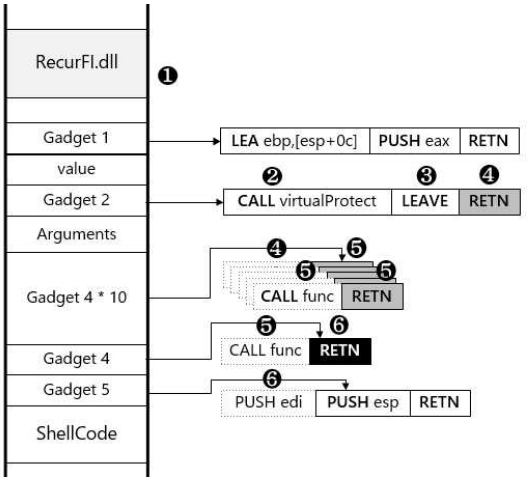
가 새롭게 등장한 ROP 공격기법을 효과적으로 막을 수 있는지 분석하였다. IE 8에서 임의의 코드를 실행할 수 있는 3가지의 취약점 〔10-12〕 을 사용하여, 새로운 ROP 공격기법을 적용한 3개의 서로 다른 공격코드를 만들었다. 해당 공격코드는 ROP 공격을 수행하며, 크게 두 가지 형태의 가젯을 가진다. 첫 번째는 virtualProtect 함수를 call 명령어로 호출하는 가젯이며, 두 번째는 call 명령어와 ret 명령어 사이에 분기문이 존재하지 않는 가젯이다.

[표 1]은 서로 다른 취약점을 가지는 IE를 새로운 ROP 공격기법을 적용한 공격코드로 공격하였을 때, ROPGuard와 RecurFI가 공격을 차단할 수 있는지 여부를 나타낸 표이다. EMET는 3번의 공격을 모두 정상적인 제어흐름으로 판단하였고, IE의 제어흐름 무결성을 보장하지 못하였다. 반면 RecurFI는 기존보다 강력한 탐지정책으로 모든 공격을 탐지하였으며 IE의 제어흐름 무결성을 보장하였다.

보다 구체적인 설명을 위하여 CVE-2013-1347 취약점을 가지는 IE에 새로운 ROP 공격을 수행하는 과정과 그 과정에 적용되는 RecurFI의 보안정책을 [그림 4]에 도식화 하였다. ❶ 가장 먼저, IE 프로세스에 삽입된 RecurFI.dll은 virtualProtect 함수를 포함하는 사용자가 지정한 API를 계측한다. 생성된 IE 프로세스가 정상적으로 수행되다가 RecurFI.dll에서 virtualProtect 함수를 후킹하였기 때문에 ❷ 가젯 2의 call virtualProtect 명령어가 수행되기 전에 RecurFI의 simulateRetn알고리즘이 동작하여 앞으로 수행하기 될 리턴주소의 무결성을 검사한다. call virtualProtect 명령어의 리턴주소는 leave 명령어를 가리키며, 해당 명령어는 call 명령어에 이어서 위치하기 때문에, RecurFI는 call virtualProtect 명령어의 리턴주소가 유효하다고 판단한다. 여기서 무

〔표 1〕 IE 의 제어흐름 무결성 보장 여부

	ROPGuard	RecurFI
CVE-2012-4969 〔10〕	✘	✔
CVE-2013-1347 〔11〕	✘	✔
CVE-2013-2551 〔12〕	✘	✔



〔그림 4〕 새로운 공격기법 사례 연구

결성 검사를 멈추지 않고, RecurFI는 ❸ call virtualProtect 명령어에 이어지는 leave 명령어를 디스어셈블리하여, leave 명령어로 인하여 변경되는 스택 포인터 값을 정적으로 계산하고, ❹ 그 다음 명령어인 ret 명령어가 수행될 때의 리턴주소의 유효성을 *재귀적*으로 검증한다. ret 명령어의 리턴주소는 ret 명령어에 해당하고, 해당 명령어 역시 call 명령어에 이어서 존재하기 때문에 RecurFI는 리턴주소는 유효하다고 판단한다. ❺ 가젯 4는 ret 명령어만 가지는 가젯이며, call 명령어에 이어서 존재하기 때문에, 가젯 4를 연속적으로 호출할 경우 RecurFI는 매 리턴주소가 유효하다고 판단한다. 이에 반해 ROPGuard는 API호출 이후 10개의 명령어 중에 존재하는 ret 명령어의 유효성만 검사하기 때문에, 가젯 4를 10번 이상 반복하면 ROPGuard는 해당 제어흐름을 정상으로 판단하여 가젯 5부터는 CFI기술을 적용하지 않는다. ❻ RecurFI는 재귀적으로 CFI기술을 적용하기 때문에 가젯 5로 반환하는 가젯 4의 리턴주소의 유효성을 검사한다. ret 명령어의 리턴주소는 push esp 명령어에 해당하고, 이 명령어는 push edi 명령어에 이어서 위치하기 때문에, RecurFI의 보안정책에 의거하여 해당 리턴주소는 유효하지 않기 때문에 스택이 변조되었다고 판단하고 프로세스를 종료시킨 후 사용자에게 통보한다. 이와 같이 RecurFI는 기존의 ROPGuard 보다 강력한 정책을 적용하기 때문에 위와 같은 진화된 공격도 막을 수 있었다.

IV. 결 론

본 논문에서는 제어흐름 탈취를 위한 코드 재사용 공격이 갖는 결과적 특성을 활용하여 최신 공격에 대응하는 시스템인 RecurFI를 제시하였다. 우리는 윈도우 익스플로러 환경에서, 제시된 방어 시스템의 안전성을 검증하였으며, SPEC 벤치마크를 사용하여 시스템의 오버헤드를 측정한 결과 평균 2% 미만의 실용적인 방어가 가능함을 볼 수 있었다.

참 고 문 헌

〔1〕 M. Abadi, et al., Control-Flow Integrity. In Proceedings of the ACM Computer and Communications Security, 2005.

〔2〕 Shacham, The Geometry of Innocent Flesh on The Bone: Return-into-Libc without Function Calls (on the x86), In Proceedings of the ACM Computer and Communications Security, 2007.

〔3〕 C. Tice, et al., Enforcing Forward-Edge Control-Flow Integrity in GCC & LLVM, In Proceedings of the USENIX Security Symposium, 2014.

〔4〕 I. Fratrić, ROPGuard: Runtime Prevention of Return-Oriented Programming Attacks, 2012.

〔5〕 Microsoft Enhanced Mitigation Experience Toolkit, https://www.microsoft.com/emet, 2014.

〔6〕 V. Pappas, et al., Transparent ROP Exploit Mitigation Using Indirect Branch Tracing. In Proceedings of the USENIX Security Symposium, 2013.

〔7〕 L. Davi, et al., Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection. In Proceedings of the USENIX Security Symposium, 2014.

〔8〕 M. Conti, et al., Losing Control: On the Effectiveness of Control-Flow Integrity Under Stack Attacks. In Proceedings of the ACM Conference on Computer and Communications Security, 2015.

〔9〕 M. Prasad, et al., A Binary Rewriting Defense Against Stack Based Buffer Overflow Attacks, In Proceedings of the USENIX Annual Technical Conference, 2003.

〔10〕 MS12-063 Microsoft Internet Explorer execCommand Use-After-Free Vulnerability, https://goo.gl/Y5r0et.

〔11〕 MS13-038 Microsoft Internet Explorer Cgenericelement Object Use-After-Free Vulnerability, https://goo.gl/1SoQM1.

〔12〕 MS13-037 Microsoft Internet Explorer Coalinedash stylearray Integer Overflow, https://goo.gl/votFNZ.