E9patch를 활용한 동적 패치 도구 Dyne9patch의 설계 및 구현

백지훈[1], 문현곤[2]
[1]울산과학기술원 컴퓨터공학과 석사과정
[2]울산과학기술원 컴퓨터공학과 부교수

qorwlgns444@unist.ac.kr, hyungon@unist.ac.kr

The Design and Implementation of Dyne9patch: A Dynamic Patching Tool Utilizing E9patch

Jihun Baek[1], Hyungon Moon[2]
[1]Dept. of Computer Science Engineering, UNIST
[2]Dept. of Computer Science Engineering, UNIST

요 약

동적 패치는 은행 시스템, 산업 제어 시스템 등 24시간 가동되는 프로그램에서 중요하며, 제어 흐름 정보를 최소화해 동적 패치의 효율성을 높일 수 있다. 논문에서는 제어 흐름 정보 없이 x86_64 바이너리를 재작성할 수 있는 도구인 E9patch를 기반으로, Dyne9patch라는 동적 패치 도구를 구현했다. 피보나치 수열을 출력하는 프로그램을 통해 Dyne9patch의 동적 패치 적용 가능성을 확인했으며, 후속 연구로는 자동화된 패치 적용 도구 개발을 제안하고 있다.

1. 서론

현재 사람들의 생활 속에서 다양한 프로그램들이 작동되고, 사용되고 있다. 그중에는 은행 서비스나 산업 제어 시스템(Industrial Control System) 등 24시간 가동이 되어야 하는 프로그램들도 존재한다. 정기적으로 진행되는 업데이트를 해야 할 때, 혹은 갑자기 발견된 취약점을 패치를 할 때 프로그램을 중단해야 하는 상황이 발생하는데, 이 프로그램의 중단으로 발생하는 비용은 생각보다 critical 하다.

이에 따라 프로그램의 중단 없이 패치를 진행할 수 있는 동적 패치(Dynamic patch)에 관련된 연구의 중요성이 강조되고 있다. 프로그램의 실행이 중지된 상태에서 소스 코드나 바이너리를 수정한 후, 다시 컴파일하는 정적 패치(Static patch)와는 다르게 동적 패치는 프로그램이 실행 중인 상태에서 직접 메모리에 있는 코드를 수정하는 방식이다.

실행 중인 프로그램에 직접 패치를 진행하는 동적 패치의 특성상, 동적 패치에서 제어 흐름 정보(Control flow information)를 아는 것은 중요하다. 코드 위치 파악 및 코드의 경로 수정, 안정성 확보 등등 제어 흐름 정보로부터 얻는 정보를 기반으로 동적 패치를 진행하기 때문이다.

이러한 제어 흐름 정보를 최소화하여 동적 패치를 진행할 수 있다면 동적 패치를 진행하기 위한 조건이 완화될 수 있는 효과를 얻을 수 있다. 이 논문에서는 제어 흐름 정보 없이 x86_64 바이너리에 사용할 수 있는 정적 바이너리 재작성 도구인 e9patch의 방법을 활용하여 Dyne9patch를 구현하였고, 피보나치수열을 계속 출력하는 테스트 프로그램에서 순조롭게 동적 패치가 진행됨을 확인함으로써 동적 패치 도구의 사용 가능성을 확인하였다.

2. 배경지식

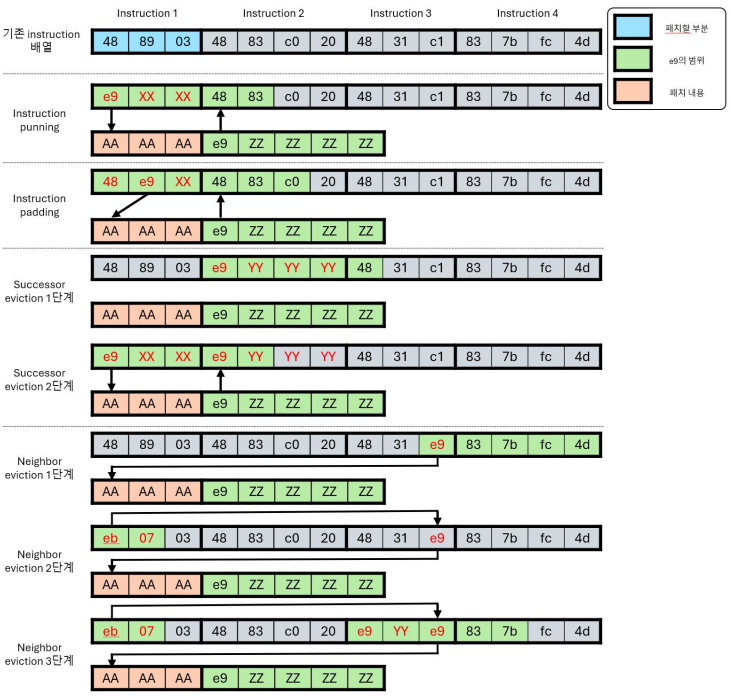
동적 패치는 프로그램이 실행 중인 상태에서 직접 메모리에 있는 코드를 수정하거나 업데이트하는 방식을 말한다. 즉, 애플리케이션이 실행 중인 동안에도 변경 사항을 적용할 수 있다. 동적 패치를 활용하면 시스템이나 프로그램을 재시작하지 않고도 패치를 적용할 수 있기 때문에 즉각적인 버그 수정이 가능하며, 다운타임이 거의 없어 사용자들에게 불편함을 주지 않는 장점이 있다. 하지만 실시간으로 메모리를 수정하기 때문에, 예기치 않은 오류가 충돌이 발생할 가능성이 있고, 보안 취약점이 발생할 수 있어 유의해야 한다.

E9patch [1]는 e9이라는 jump instruction을 활용

하여 제어 흐름 정보 없이 x86_64 바이너리를 정적으로 재작성하는 도구로, 기존 바이너리 수정 도구들이 제어 흐름 정보를 복원하는 데 의존하는 문제를 해결하는 논문이다. E9patch는 instruction punning, padding, eviction 등의 방법을 활용하여 바이너리의 점프 타겟을 이동시키지 않고 코드 수정을 가능하게 하였다. 따라서 제어 흐름 복구의 필요성이 사라져 심볼 정보나 디버그 데이터 없이도 작동이 가능하고, 특히 Chrome, Firefox와 같은 대규모 바이너리에서 확장이 가능한 것이 특징이다.

E9patch는 성능을 인정받고 다양한 곳에 활용되고 있다. 취약점 수리를 위한 반례 기반 귀납적 추론 절차를 활용한 프로그램 패치를 생성하는 도구인 VulnFix [2]의 계측 모듈(Instrumentation Module) 부분은 e9patch를 기반으로 구현되었다. 또한, 동시성 프로그램(Concurrent Program)에서 발생할 수 있는 버그들을 효과적으로 발견하기 위한 RFF [3]에서도 효율적으로 동적 계측을 하기 위해 e9patch를 활용하였다.

x86_64 아키텍처를 대상으로한 e9patch 뿐만 아니라 ARMore [4]과 같이 ARM, RISC-V 아키텍처에서도 정적 바이너리 재작성 도구 관련 연구들이 활발히 진행되고 있다. 하지만 이 논문에서는 x86_64 바이너리를 목표 대상으로 연구를 진행하였기 때문에, e9patch를 활용하여 동적 패치 도구를 구현하였다.



(그림 1) e9patch의 기법을 활용한 패치 프로세스.

3. E9patch의 방법

E9patch에는 재작성을 할 때 e9이라는 4byte의 주소값을 필요로 하는 총 5byte로 이루어진 jump instruction을 사용하며, 크게 세 가지의 방법을 활용한다. 첫 번째는 instruction punning으로 기존의 LiteInst [5]에서도 사용한 방법이다. 패치를 해야 할 부분이 5byte보다 작을 때, 다음 instruction과 일부 byte를 겹치게 jump instruction을 사용한 이후, 패치의 마지막 부분에는 다음 instruction의 시작하는 위치로 jump instruction을 사용하는 방법이다.

다음은 instruction padding의 방법으로, 처음 언급했던 instruction punning의 방법으로 점프할 도착 주소의 메모리가 사용할 수 없을 때 사용하는 방법이다. e9의 앞부분에 jump instruction의 내용을 해치지 않는 특정 값을 입력하여 e9의 instruction 길이를 6, 7byte 이상으로 늘리는 방법이다. 해당 방법을 사용하면 e9으로 점프할 수 있는 메모리 주소의 범위가 넓어지게 된다.

마지막으로는 eviction이라 부르는 방법이다. Instruction padding의 방법으로도 사용할 수 있는 메모리가 없을 때 사용하는 방법이다. Eviction은 패치할 instruction이 아닌 다른 instruction에 e9 instruction을 작성하여 해당 바이트 값들을 변경한 다음, 패치 위치의 e9 instruction을 재작성하는 방법이다. 이때, 바로 다음의 instruction을 재작성하는 방법을 successor eviction이라 표현한다. 그리고 근처의 instruction에 e9 jump instruction을 재작성한 뒤, 패치 위치에는 1byte jump instruction인 eb를 활용해 e9 jump instruction의 위치로 점프를 하는 방법도 있으며 이를 neighbor eviction이라 부른다. e9 점프를 2번 시도하여 패치를 진행하는 방법이며 neighbor eviction의 경우에는 e9 점프 2번과 eb 점프 한 번으로 총 3번의 점프를 활용하여 패치를 진행한다.

4. 디자인

동적 바이너리 패치 도구 Dyne9patch는 e9patch의 재작성 방법을 활용하여 c++에서 구현하였다. 프로세스의 부착 및 분리는 ptrace 시스템을 활용하여 진행했고, 메모리 읽기와 쓰기 작업도 ptrace 명령인 PTRACE_PEEKDATA와 PTRACE_POKEDATA를 사용하였다. 패치 방법은 e9patch에서의 punning과 padding을 활용하였고, 현재 단계에서는 punning 방법을 사용할지 padding 방법을 사용할지 직접 선택

하여 패치를 진행할 수 있도록 구현하였다.

입력 값으로 현재 실행 중인 프로세스의 이름, 패치를 해야 할 이름의 함수, 패치 함수의 이름과 패치 할 방법을 입력하여 Dyne9patch를 사용할 수 있다. Dyne9patch는 먼저 프로세스의 이름을 사용하여 pid 값을 얻은 후, pid 값과 패치 해야 할 이름의 함수를 활용하여 패치 해야 할 메모리 위치를 찾는다. 그 후, 해당 메모리 위치의 e9 instruction 재작성을 시작하여 패치 함수로 점프할 수 있도록 패치를 진행한다.

이 모든 과정은 동적으로 진행되며 피보나치수열의 다음 값을 계속 출력하는 테스트 프로그램으로 Dyne9patch를 사용해보았다. 피보나치수열의 다음 값을 다양한 방법으로 구할 수 있다. 기존 프로그램은 재귀함수를 활용하여 피보나치수열의 값을 구하는 방식으로 진행이 될 때, Dyne9patch를 활용하여 이를 동적 프로그래밍 방법을 사용하여 피보나치수열의 값을 구하는 함수를 패치 코드로 설정하였고, 특정 문자를 출력함으로써 패치 성공 유무를 확인하였다. 프로그램이 실행하는 중에 패치를 진행하였고, 프로그램의 중단 없이 동적으로 패치가 진행되는 것을 확인할 수 있었다.

5. 후속 연구

현재 구현한 Dyne9patch는 패치 방법을 사용자가 직접 선택해야하며, 패치해야할 함수의 위치와 패치 함수를 입력해야하는 간단한 동적 패치 도구이다. 하지만, 제어 흐름 정보 없이 재작성이 가능한 e9patch를 활용함으로써 필요한 제어 흐름 정보를 최소화하여 동적 패치가 가능함을 확인할 수 있었다. 이후 후속 연구에서는 Dyne9patch를 확장하여 패치 대상 바이너리와 패치 바이너리를 입력할 때, 두 바이너리를 비교하여 패치 위치와 내용을 분석한 뒤, e9patch의 3가지 방법 중 사용하기에 적합한 방법을 직접 찾아 동적 패치를 진행하는 도구로 구현할 계획이다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음(IITP-2024-2021-0-01817)

참고문헌

[1] Duck, Gregory J., Xiang Gao, and Abhik Roychoudhury. "Binary rewriting without control flow recovery." Proceedings of the 41st ACM SIGPLAN conference on programming language design and implementation. 2020.

[2] Zhang, Yuntong, et al. "Program vulnerability repair via inductive inference." Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. 2022.

[3] Wolff, Dylan, et al. "Greybox Fuzzing for Concurrency Testing." Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2024.

[4] Di Bartolomeo, Luca, Hossein Moghaddas, and Mathias Payer. "{ARMore}: Pushing Love Back Into Binaries." 32nd USENIX Security Symposium (USENIX Security 23). 2023.

[5] Chamith, Buddhika, et al. "Instruction punning: Lightweight instrumentation for x86-64." Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2017.