안티퍼징 기술 우회

송수환 [1], 이병영 [2]
[1] 서울대학교 전기정보공학부 석박사통합과정
[2] 서울대학교 전기정보공학부 교수

sshkeb96@snu.ac.kr, byoungyoung@snu.ac.kr

Breaking Anti-Fuzzing

Su-Hwan Song[1], Byoungyoung Lee[2]
[1]Dept. of Electrical and Computer Engineering, Seoul National University
[2]Dept. of Electrical and Computer Engineering, Seoul National University

요 약

이 논문은 시스템의 안전성을 강화하는 안티퍼징을 우회하는 방법을 제안한다. 먼저, 안티퍼징을 우회하기 위해 가짜 Basic Block 을 효과적으로 식별하는 방법을 제안한다. 이를 통해 공격자가 시스템의 취약점을 파악하고 이를 이용하는 것이 가능하게 할 수 있다. 또한, 이 논문은 안티퍼징 메커니즘을 우회하기 위해 커널의 Syscall signal 을 수정하여 프로그램이 크래시가 발생할 때 정상적으로 종료되지 않도록 하는 방법을 제시한다. 이를 통해 시스템이 안티퍼징 메커니즘을 우회하고 퍼징 공격을 가능하게 한다. 이러한 연구 결과는 향후 보다 안전하고 안정적인 시스템 보호를 위한 중요한 연구 방향을 제시하고자 한다.

1. 서론

최근 몇 년 동안, 퍼징 기술은 소프트웨어 보안 분야에서 큰 주목을 받고 있었다 [3, 4, 5]. 퍼징은 애플리케이션 또는 시스템의 입력에 대해 대규모의 임의 또는 부분적으로 변형된 데이터를 제공하여 보안 취약점을 찾는 기술이다. 그러나 이러한 퍼징 기술은 악의적인 공격자가 프로그램의 취약점을 먼저 찾아 exploit 할 수 있는 문제가 있다. 이 문제를 해결하기 위해 안티퍼징 (Anti-fuzzing) 기술이 제안되었다 [1, 2]. 안티퍼징(Anti-fuzzing)은 퍼징 공격에 대응하여 시스템 또는 소프트웨어의 보안을 강화하는 기술이다. 다양한 안티퍼징 기술이 제안되어 왔으며, 이 중 일부는 "Fake Coverage", "Anti-Crash"와 같은 기법을 사용하여 보호한다.

그러나 우리는 이 논문에서는 안티퍼징을 우회하고 무력화하기 위한 두 가지 기술에 대해 다룬다. 첫째로, "Fake Basic Block Identification via BB Trace Profiling" 이라는 기법을 제안하여 안티퍼징을 극복한다. 이 기법은 무작위로 생성된 입력에 의해 실행되는 모든 Basic Block 을 식별하고, 다양한 Basic Block 추적 프로필을 변화시켜 안티퍼징을 효과적으로 우회한다. 둘째로, Anti-Crash 를 극복하기 위해 커널의 Syscall signal 을 수정하여 프로그램이 crash 가 발생해도 정상적으로 종료하는 것을 방지한다. 이러한 두 가지 접근 방식을 통해 안티퍼징을 쉽게 우회할 수 있다.

2. 배경지식

1. 퍼징(Fuzzing)은 소프트웨어 테스트 및 보안 취약점 발견 기술 중 하나로, 대상 시스템 또는 프로그램에 무작위 또는 부분적으로 변형된 입력을 제공하여 예상치 못한 동작을 유도하고 취약점을 찾는 과정을 말한다. 이는 대규모의 데이터, 명령어 또는 이벤트를 사용하여 시스템의 응답을 검증하는 자동화된 테스트 기술이다. 퍼징은 주로 보안 취약점을 발견하고 이를 악용하는 공격자의 공격을 사전에 예방하기 위해 사용된다. 일반적으로 퍼징은 다음과 같은 단계로 이루어진다:

i) 입력 생성: 퍼징 과정에서 사용될 입력 데이터를 생성한다. 이는 무작위 또는 부분적으로 변형된 데이터일 수 있으며, 목표 시스템의 예상되는 입력 형식과 일치시켜야 한다.

ii) 입력 적용: 생성된 입력 데이터를 대상 시스템 또는 프로그램에 제공한다. 이 때 대상 시스템은 입력을 처리하고 그에 따른 동작을 수행한다.

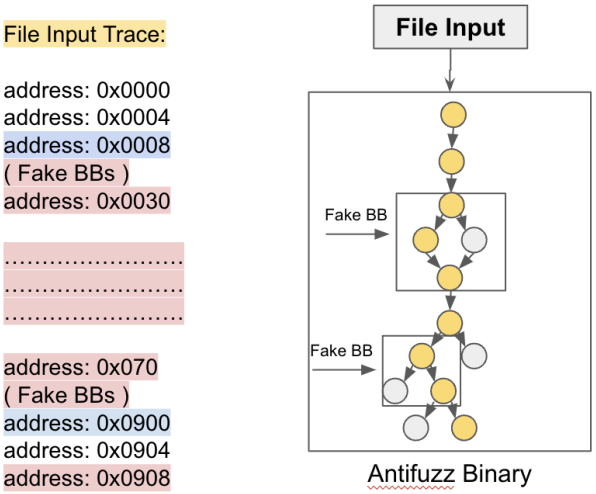
iii) 동작 확인: 퍼징 과정 중에 대상 시스템의 동작을 확인하고 기록한다. 예상치 못한 동작이나 오류가 발생하는 경우, 이를 취약점의 가능성으로 간주한다.

2. 안티퍼징(Anti-fuzzing)은 퍼징 공격에 대응하여 시스템 또는 소프트웨어의 보안을 강화하는 기술이다. 여러가지 안티퍼징 기술 중에서, "Fake Coverage",

"Delay Execution Speed", 그리고 "Anti-Crash (Anti-Fuzz)"이 있다:

i) Fake Coverage (그림 1) 는 퍼징 공격자가 시스템의 퍼징 결과를 판단하는 데 사용하는 커버리지 정보를 변조하는 것을 의미한다. 이는 퍼징 공격자가 정확한 커버리지 정보를 파악하기 어렵게 만들어 퍼징 시도의 효과를 저하시킨다. 예를 들어, 시스템이 가짜 커버리지 정보를 반환하여 퍼징 공격자를 속이는 방식으로 구현될 수 있다.

ii) Anti-Crash 또는 Anti-Fuzz 는 시스템이 입력 데이터의 예외적인 상황에 대응하여 크래시를 방지하는 기술이다. 이는 퍼징 공격자가 특정 입력 데이터를 이용하여 시스템을 크래시하거나 불안정하게 만드는 것을 방지한다.



(그림 1) **Example of Fake Basic Block**.

3. 안티퍼징 우회 기술

1. Fake Basic Block Identification

안티퍼징은 프로그램에 불필요한 Basic Block (가짜 Basic Block)을 무작위로 삽입하여 퍼징을 방해하는 기술이다. 이러한 안티퍼징을 우회하고 퍼징 방어 메커니즘을 무력화하기 위해 "BB Trace Profiling 을 통한 Fake Basic Block Identification"이라는 기법을 제안한다. 이 방법은 무작위로 생성된 입력에 의해 실행되는 모든 Basic Block 을 식별하고, 다양한 Basic Block 추적 프로필을 변화시켜 안티퍼징을 극복하는 것이다.

이 기법은 안티퍼징의 두 가지 컨셉에 근거하여 제시한다. "가짜 Basic Block"을 효과적으로 우회하기 위해서 두 가지 사실에 근거하여 제시한다. 첫 번째 사실은 가짜 Basic Block 이 입력 파일에 따라 극적으로 다양하다는 점이다. 두 번째 사실은 가짜 Basic Block 을 얻는 것이 매우 쉽다는 것이다. 이는 퍼징을 수행할 때, 다양한 무작위 입력 파일을 사용하여 가짜 Basic Block 을 파악하는 것이 가능해진다.

이를 두 가지 사실을 기반으로, 우선 무작위로 생성된 입력을 통해 실행되는 모든 쉽게 찾을 수 있는 Basic Block 을 식별하는 것이 가능해진다. 가짜 Basic Block 을 식별하는 과정은 다음과 같다 (그림 2).

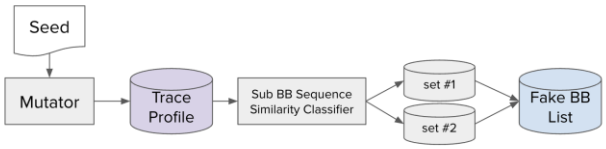
i) 무작위로 입력을 하나 생성한 뒤, 프로그램을 실행한다.

ii) 무작위로 생성된 입력은 보통 프로그램 초기에 Reject 되기에 해당 입력으로 찾은 Basic Block 들은 무의미한 Basic Block 일 가능성이 높다.

iii) 해당 Basic Block 들을 Fake BB List 에 추가한다.

iv) 더 이상 Fake BB 가 나오지 않을 때까지 반복한다.

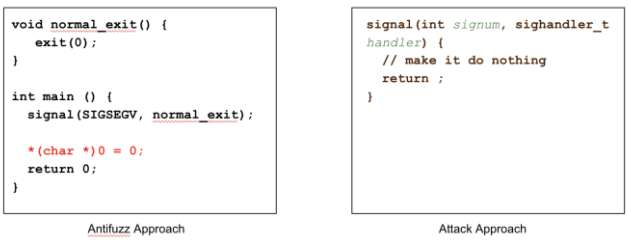
이러한 접근 방식을 통해 안티퍼징 공격자는 안티퍼징을 우회하고 퍼징 방어 메커니즘을 무력화할 수 있습니다. 이러한 과정에서 발생하는 다양한 실행 경로는 안티퍼징을 우회하는 데 중요한 역할을 합니다.



(그림 2) **Fake Basic Block Identification via BB Trace Profiling**.

2. Bypassing Anti-Crash

다음으로는 Anti-Crash 를 극복하기 위해 커널의 Syscall signal 을 수정하여 프로그램이 crash 가 발생해도 정상적으로 종료하는 것을 방지한다 (그림 3). 아래의 그림과 같이 프로그램이 안티퍼징 기술을 활용하여 SIGSEGV 같은 신호를 정상 종료 시키는 경우, 우리는 syscall signal 에서 해당 신호를 처리 하지 않게 하여 커널을 수정하여 프로그램에서 crash 가 발생하면 비정상 종료를 하게 한다. 이러한 방식을 통하여 퍼징이 다시 가능하게 된다.



(그림 3) **Anti-Crash and Our Approach**.

4. 실험결과

표 1 은 우리의 기법을 최신 안티퍼징 기법에 적용하여 우회 가능한지 실험한 결과이다. 우리 디자인은 최신 안티퍼징 기법은 Antifuzz [2]의 두가지 기법을 전부 우회할 수 있었다. 또 다른 안티퍼징 기법인 Fuzzification [1]에서는 Anti-crash 는 완전 우회가능 했지만, Fake Basic Block 를 완벽하게 우회 하지는 못하였다. 그러나 대부분의 Fake Basic Block 을 탐지하여 우회할 수. 있다.

(표 1) **안티퍼징 기법에 대한 우회 실험**

	Fake Basic Block	Anti-Crash
Antifuzz [2]	Bypassed	Bypassed
Fuzzification [1]	Almost Bypassed	Bypassed

5. 결론

결론적으로, 이 연구는 안티퍼징 공격에 대응하여 시스템의 안전성을 강화하는 방법을 탐구하였습니다. 우리는 먼저 가짜 Basic Block 을 효과적으로 식별하여 안티퍼징 방어를 우회하는 방법을 제안했습니다. 이를 통해 공격자가 시스템의 취약점을 파악하고 이를 악용하는 것을 어렵게 만들었습니다. 또한, Anti-Crash 기술을 이용하여 프로그램이 크래시가 발생해도 정상적으로 종료되지 않도록 하는 방법을 소개했습니다. 이를 통해 시스템이 안티퍼징 메커니즘을 우회하고 퍼징 공격을 가능하게 하였습니다. 이러한 연구 결과는 향후 보다 안전하고 안정적인 시스템 보호를 위한 중요한 연구 방향을 제시하고자 합니다.

6. 사사

2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-01840,스마트폰의 내부데이터 접근 및 보호 기술 분석)

참고문헌

[1] Jung, Jinho, et al. "Fuzzification:{Anti-Fuzzing} techniques." 28th USENIX Security Symposium (USENIX Security 19). 2019.

[2] Güler, Emre, et al. "{AntiFuzz}: Impeding Fuzzing Audits of Binary Executables." 28th USENIX Security Symposium (USENIX Security 19). 2019.

[3] Yun, Insu, et al. "{QSYM}: A practical concolic execution engine tailored for hybrid fuzzing." 27th USENIX Security Symposium (USENIX Security 18). 2018.

[4] Li, Yuekang, et al. "Steelix: program-state based binary fuzzing." Proceedings of the 2017 11th joint meeting on foundations of software engineering. 2017.

[5] She, Dongdong, et al. "Neuzz: Efficient fuzzing with neural program smoothing." 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.