목표 지향 프로토콜 취약점 탐지 도구 설계

엄재은[1], 전유석[2]
[1]울산과학기술원 컴퓨터공학과 석박통합과정
[2]울산과학기술원 컴퓨터공학과 교수
eomjaeeun@unist.ac.kr, ysjeon@unist.ac.kr

ProtoGo: State-aware Directed Fuzzing for Network Applications

Jaeeun Eom[1], Yuseok Jeon[2]
[1]Dept. of Computer Science Engineering, UNIST
[2]Dept. of Computer Science Engineering, UNIST

요 약

본 연구는 네트워크 프로그램의 취약점을 신속하게 탐지하기 위해 Directed Fuzzing 기법을 적용한 ProtoGo를 제안한다. Directed Fuzzing은 특정 목표 지점을 집중하여 탐색하는 데 효과적이지만, 기존 기법들은 네트워크 프로토콜의 복잡한 상태 전이를 충분히 고려하지 못하는 한계가 있다. 이를 해결하기 위해, 본 연구에서는 정적 분석을 통해 프로토콜의 상태 기계를 생성하고, 각 상태와 함수 호출 간의 관계를 추적하여 효율적인 탐색 경로를 제공하는 도구를 설계하였다. 향후 연구를 통해 ProtoGo의 프로토타입을 구현하고, 성능을 검증할 예정이다.

1. 서론

최근 소프트웨어 개발은 여러 사람이 협력하는 대형 프로젝트로 발전함에 따라 업데이트와 패치가 잦아지면서 새로운 취약점이 발생할 가능성이 커졌다. 특히 수정된 코드나 신규 기능은 취약점 발생 확률이 높아 이를 집중적으로 테스트하는 것이 중요하다. 그러나 복잡한 시스템에서 전통적인 Fuzzing 기법은 비효율적이며, 특정 코드에 집중할 수 있는 Directed Fuzzing의 중요성이 커지고 있다.

Directed Fuzzing은 입력값과 코드 위치 간 거리를 계산해 목표에 더 가까운 입력값을 우선 탐색하는 Fuzzing 기법으로, 패치 검증이나 취약점 재현에 유용하다. 그러나 Directed Fuzzing에 관한 연구가 활발하게 진행되는 것에 비해, 네트워크 프로그램과 같은 복잡한 시스템에서 Directed Fuzzing을 적용하는 연구는 여전히 부족한 상태이다. 네트워크 프로그램은 그 복잡한 특성 때문에 기존 Directed Fuzzing을 그대로 적용하기 어렵다.

네트워크 프로그램의 복잡성은 프로토콜의 상태에서 비롯된다. 프로토콜의 상태는 통신 과정에서 이전에 발생한 이벤트와 현재 시스템의 진행 상황을 반영하며, 특정 상태에 도달해야만 목표 취약점이 발생한다. 따라서 네트워크 프로그램에서 취약점을 효과적으로 탐지하려면, 단순히 코드 간의 거리를 기준으로 탐색하는 것이 아니라, 상태 전이를 추적하고 이를 기반으로 탐색 경로를 최적화해야 한다.

기존 Directed Fuzzer들은 상태 변화를 고려하지 않고, 코드 블록 사이의 거리만을 기준으로 해 탐색을 수행한다. 이로 인해 탐색 경로가 취약점과 거리상으로는 가까워 보일 수 있지만, 잘못된 상태에 진입해 목표 취약점에 도달하지 못할 가능성이 크다. 네트워크 프로그램은 여러 상태를 거치는 복잡한 경로를 가지기 때문에, 기존 Directed Fuzzing 기법은 쉽게 적용할 수 없다.

네트워크 프로그램의 취약점을 효과적으로 탐지하기 위해서는 Fuzzer에게 상태 정보를 제공하고 이를 바탕으로 탐색 경로를 안내하는 방식이 필요하다. 이를 위해서는 프로토콜의 상태 기계를 생성하고, 그 상태를 기반으로 탐색해야 한다. AFLNet [1]과 BLEEM [2] 같은 최신 연구들은 응답 정보를 활용해 상태를 정의하지만, 이러한 정의는 프로그램의 실제 실행 경로와 완전히 대응되지 않아 특정 목표 상태에 집중하기 어렵다.

SGFuzz [3]는 열거형 변수를 상태 변수로 사용해 실행 경로와 상태를 연관지을 수 있지만, 동적으로 상태 기계를 생성하는 방식으로 인해 비효율적이

다. 프로토콜의 상태는 순서대로 달성해야 하는데, 동적 상태 기계 생성은 경로 예측이 어렵고 목표 상태를 알고 있더라도 현재 코드 위치 및 상태와 어떤 관련이 있는지 알 수 없기 때문에 불필요한 경로 탐색이 필히 존재한다.

따라서 본 연구에서는 정적 분석을 통해 상태 기계를 구성하고, 프로토콜 구현체의 취약점에 도달하기 위한 최적의 상태 경로를 찾아 빠르게 취약점을 트리거할 수 있는 목표 지향 프로토콜 취약점 탐지 도구를 설계하고자 한다.

2. 이론적 배경

2.1 상태 저장 프로토콜 fuzzing의 어려움

프로토콜 구현체는 주로 메시지 기반(message-driven)으로 동작하며, 이 메시지는 패킷(packet)이라는 단위로 전달된다. 패킷은 네트워크를 통해 전송되는 데이터의 기본 단위로, 구문적 규칙과 의미적 규칙을 모두 포함하고 있다. 수신자는 패킷을 수신한 후, 규칙에 따라 파싱 (parsing)하여 패킷이 구문적 및 의미적으로 무결성을 유지하고 있는지 확인한다.

따라서 입력 전체에 대한 무작위적 변형(mutation)은 프로토콜의 구문 및 의미 규칙을 파괴하여, 패킷이 수신 후 파싱 과정에서 무결성 검증을 통과하지 못하고 폐기될 가능성이 높아진다.

많은 프로토콜에서 이러한 무결성 검증 단계는 파싱 루프 (parsing loop) 내에서 수행되며 상태를 이용하여 파싱 단계를 기록한다. 패킷이 검증 규칙을 통과할 때마다 프로토콜의 상태가 업데이트되는 방식이다. 대부분의 네트워크 프로그램을 목표로 하는 Stateful fuzzer들은 프로그램의 실행 경로와 함께 상태의 변화와 전이를 함께 저장하는 상태 기계를 활용한다. 새로운 코드를 발견하는 입력값의 우선순위를 높이는 전통적인 Fuzzing과 다르게 Stateful fuzzing은 새로운 상태를 발견하는 입력값의 우선순위를 높이는데, 그 이유는 목표 위치에 도달하기 위한 실행 경로를 제외한 다른 경로의 우선순위가 높아지면 불필요한 경로를 탐색할 가능성이 높아져 비효율적이기 때문이다.

2.2 Directed Fuzzing

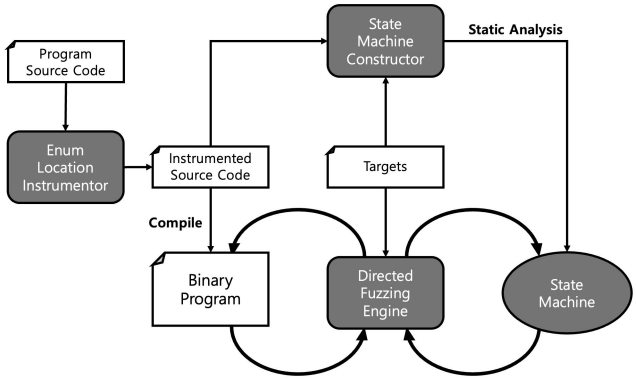
Directed fuzzing은 입력값과 목표 위치 간의 거리를 정의하고, 그 거리가 점차 줄어드는 방향으로 시드 입력값을 유도하여 원하는 타겟 위치에 도달하는 기법이다. Directed fuzzing의 대표적인 초기 연구로는 AFLGo [4]가 있으며, 이후 이를 발전시킨 다양한 Directed Fuzzer들이 등장하였다. Directed Fuzzing은 특정 버그를 탐지하거나 이미 알려진 취약점을 재트리거하는 데에 있어 유용하며 특히, 입력값의 무작위성을 줄이고 목표와 관련된 Mutation 및 스케줄링에 집중하는 Directed Fuzzing은 프로토콜의 제한적인 특성을 절충하는 데 유리하다. 하지만 프로토콜은 그 목적에 따라 상태의 정의 및 구현이 다양하며, 같은 프로토콜이라도 구현체에 따라 상태 전이 과정이 달라질 수 있다. Directed fuzzing을 위해서는 방향성을 제공하는 우선 순위의 기준이나 스코어가 필요하기 때문에, 대부분의 프로토콜에 적용 가능한 범용성을 고려한 스코어링 방법이 요구된다.

따라서 본 연구는 열거형 변수를 상태 변수로 사용한다. 대부분의 프로토콜 구현체들은 입력값을 처리하는 과정에서 진행 상황을 열거형 변수에 저장한다 [3]. 열거형 변수에 새로운 값이 할당되는 것을 상태의 변화로 가정하고, 정적 분석을 통해 해당 위치를 찾아 상태 기계 생성 및 탐색을 진행한다.

현재, 프로토콜 구현체를 대상으로 한 Directed Fuzzing 연구는 여전히 초기 단계이다. Greyhound [5]는 Wi-Fi 프로토콜을 대상으로한 Directed Fuzzer이다. Greyhound는 Wi-Fi 프로토콜이 여러 계층으로 구성된다는 점을 이용하여, 각 계층에 다른 변형 확률을 적용하고, 취약점 발견 가능성이 높은 계층에 에너지를 집중한다. 그러나 Fuzzing 과정 중에 상태 기계를 생성하는 것이 아니라 이미 만들어진 상태 기계를 사용하기 때문에 다른 프로토콜에 범용적으로 적용할 수 없다.

AFLNETGo [6]은 열거형 변수를 상태 변수로 사용하지만, SGFuzz처럼 시드 입력값에 의존해 동적으로 상태 기계를 구성한다. 이 방식은 시드 다양성이 부족할 경우, 목표 상태를 정확히 지정하지 못해 비효율적이다. 상태는 단순 실행 범위를 알려주는 독립적인 지표가 아닌, 서로 연관되고 순서에 종속적인 변수이다. 즉, 실행 경로에 존재할 수 있는 다른 상태 전이 과정을 충분히 고려하지 않고 발견된 상태에만 의존하는 방식은 복잡한 상태 저장 프로토콜에서 비효율적인 탐색을 야기할 수 있다. 본 연구는 정적 분석을 통해 사전에 상태 기계를 생성해 이러한 한계를 극복하고, 기존 방식보다 불필요한 경로 탐색을 줄여 목표 위치에 신속하게 도달할 수 있도록 한다.

3. 디자인



(그림 1) Design Overview

본 연구에서 제안하는 Fuzzer는 상태 저장 프로토콜의 특성을 반영하여 상태 전이를 추적하고, 이를 기반으로 효율적인 Directed Fuzzing을 수행하는 것을 목표로 한다. 이를 위해 정적 분석을 사용하여 상태 기계를 생성하고, 각 상태와 해당 상태에 대응하는 함수 간의 관계를 구축한 후, 목표 취약점에 도달하기 위한 상태 시퀀스를 기반으로 Fuzzing 에너지를 할당하는 방식을 제안한다. 그림 1은 본 연구에서 제안하는 도구의 디자인 개요를 보여준다.

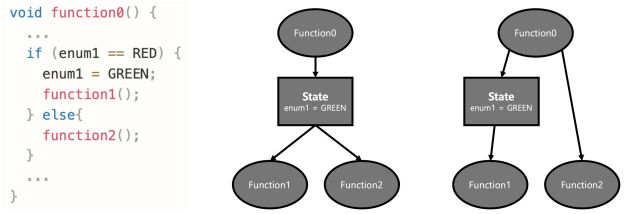
3.1 열거형 변수 Instrumentation을 통한 상태 추적

(그림 2) 열거형 변수 위치의 Instrumentation 전후, Live555 코드 중 일부

본 연구에서는 열거형 변수에 새로운 값이 할당되는 시점을 State로 정의한다. 첫 번째 단계에서는 소스코드에서 열거형 변수가 새로운 값으로 할당되는 지점을 LLVM의 Clang Tooling API [7]를 사용해 AST를 탐색하고, 상태가 변화하는 시점에 Instrumentation으로 함수 호출을 삽입한다. 열거형 변수는 IR에서 정수형으로 변환되므로, AST 수준에서 정적 분석을 통해 할당 위치를 파악해야한다. 이 Instrumentation은 Fuzzing 실행 시 상태 전이의 발생을 Fuzzer에게 알리기 위함이다. 그림 2는 Instrumentation 전후의 코드를 보여준다.

3.2 상태 기계 생성

프로토콜의 상태 전이를 모델링하기 위해 Call Graph (CG)와 Interprocedural Control Flow Graph (ICFG)를 사용해 상태 기계를 생성한다. 탐색 전, 함수와 상태 노드를 저장하는 State map을 초기화하고, CG에서 깊이 우선 탐색을 통해 main 함수에서 시작한다. 탐색 중 Instrumentation이 추가된 상태 전이 함수를 만나면 새로운 상태 노드를 생성하고, 다른 상태 전이 함수를 만날 때까지 해당 상태를 유지한다. 이후 새로운 상태 전이 함수를 만나면 상태 노드를 추가하고 간선을 연결하며, 기존 상태 전이 함수를 만나면 State map에서 대응 상태를 찾아 간선을 추가한다. 예를 들어, 함수 P에서 상태 A가 생성되면, P가 호출하는 Q와 R도 상태 A와 매핑된다. 이후 열거형 변수에 새로운 값이 할당되면 상태 B가 생성되고 상태 A와 B를 연결하는 간선을 추가한다.



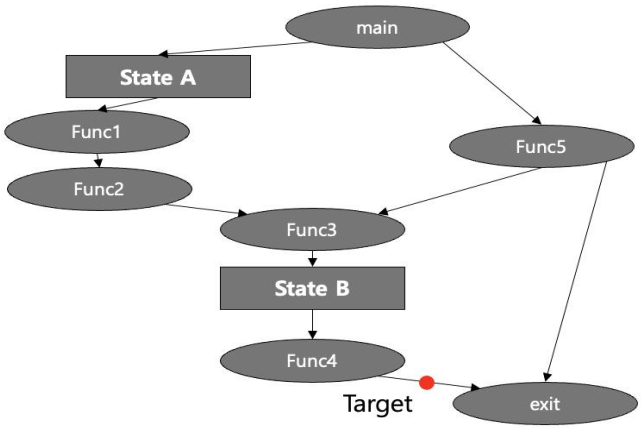
(ㄱ) (ㄴ) (ㄷ)

(그림 3) (ㄱ): 상태 전이가 일어나는 코드, (ㄴ): CG를 이용해 생성한 (ㄱ)의 상태 기계, (ㄷ): CG와 ICFG를 모두 고려해 생성한 (ㄱ)의 상태 기계

열거형 변수의 새로운 값 할당은 일반적으로 if 또는 switch 분기 안에서 발생한다. 이로 인해, 같은 함수 내에 상태 전이와 함수 호출이 존재하더라도, 그림 3-(ㄱ) 처럼 하나의 실행 경로 내에서 동시에 일어나지 않는 경우가 많다. 따라서 상태 기계 생성 중 상태 전이 함수를 만나면 해당 함수의 ICFG를 탐색해, 그림 3-(ㄴ) 처럼 CG를 기반으로 상태 기계를 업데이트한다(그림 3-(ㄷ)). 이렇게 구축된 상태 기계는 상태 전이 경로 추적에 사용된다.

3.3 목표 상태 탐색

상태 기계를 통해 각 함수 노드와 상태 정보를 저장하고, 이를 기반으로 취약점이 발생하는 함수의



(그림 4) 상태 기계와 목표 취약점

상태를 식별한다. 상태와 함수 간의 매핑 정보를 저장하는 State map을 사용해, 특정 취약점이 발생하는 함수가 속한 상태를 쉽게 찾을 수 있다. 예를 들어, 그림 4에서 타겟에 도달하기 위한 목표 상태는 State B이며, 상태 전이 경로는 (main-A-B)와 (main-B)이다. 식별된 목표 상태는 취약점 경로 추적에 활용된다.

3.4 Directed Fuzzing을 통한 목표 취약점 탐색

상태 전이 경로가 파악된 후에는 Directed Fuzzing을 수행하여 목표 상태에 도달할 수 있는 입력값을 탐색한다. 이때 각 상태 전이를 달성하는 시드 입력값에 더 많은 에너지를 할당하여 목표 취약점에 더 효율적으로 도달하도록 한다.

이와 같은 설계를 통해, 본 연구에서 제안하는 도구는 상태 저장 프로토콜에서 신속하게 취약점을 찾아낼 수 있을 것으로 예상된다.

4. 결론

본 연구는 Directed Fuzzing이 많은 연구에서 사용되고 있지만, 프로토콜에 제대로 적용되지 못한 한계를 지적하고 이를 해결하기 위한 도구를 설계하였다. 제안한 도구는 정적 분석을 통해 상태 기계를 생성하고, 목표 취약점 경로를 추적하여 프로토콜의 복잡한 상태 전이를 효과적으로 탐지할 수 있게 한다. 이를 통해 상태 저장 프로토콜에서 취약점을 신속하고 정확하게 탐지할 수 있을 것으로 기대된다. 추후 연구를 통해 도구의 프로토타입을 구현하고 성능을 향상시킬 예정이다.

Acknowledgement

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구 결과임 (RS-2024-00437306, 메모리 안전 언어의 적용 확대 및 안전 적용을 위한 통합 플랫폼 기술 개발 및 RS-2024-00341722, 지능형 서비스 로봇의 사이버 레질리언스 확보를 위한 보안기술 개발) 과 2024년도 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음 (IITP-2024-2021-0-01817).

참고문헌

[1] Pham, Van-Thuan; Böhme, Marcel; Roychoudhury, Abhik. "AFLNET: A Greybox Fuzzer for Network Protocols." 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 2020, pp. 449-454.

[2] Luo, Zhengxiong; Yu, Junze; Zuo, Feilong; Liu, Jianzhong; Jiang, Yu; Chen, Ting; Roychoudhury, Abhik; Sun, Jiaguang. "Bleem: Packet Sequence Oriented Fuzzing for Protocol Implementations." 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, 2023, pp. 4481-4498.

[3] Ba, Jinsheng; Böhme, Marcel; Mirzamomen, Zahra; Roychoudhury, Abhik. "Stateful Greybox Fuzzing." 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, 2022, pp. 3255-3272.

[4] Böhme, Marcel; Pham, Van-Thuan; Nguyen, Manh-Dung; Roychoudhury, Abhik. "Directed Greybox Fuzzing." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17), Dallas, Texas, USA, 2017, pp. 2329-2344.

[5] Garbelini, Matheus E.; Wang, Chundong; Chattopadhyay, Sudipta. "Greyhound: Directed Greybox Wi-Fi Fuzzing." IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 2, 2022, pp. 817-834.

[6] Tang, Xiaofeng; Wang, Yongjun; Xu, Fangliang. "AFLNETGO: A Directed Fuzzer for Stateful Network Protocol Implementation." 2023 8th International Conference on Signal and Image Processing (ICSIP), Suzhou, China, 2023, pp. 882-886.

[7] https://clang.llvm.org/docs/LibTooling.html