문법 기반 퍼징을 통한 WASI 런타임의 권한 취약점 탐지*

이준호[1], 박요한[1], 김진우[2,†]

[1,2]광운대학교(학부생, 교수)

Finding Capability Enforcement Gaps in WASI Runtimes via Grammar-based Fuzzing

Junho Lee[1], Yohan Park[1], Jinwoo Kim[2,†]

[1,2]Kwangwoon University(Undergraduate student, Professor)

요 약

최근 웹어셈블리(WebAssembly)는 클라우드 및 호스트 환경의 주요 기술 중 하나로 부상하고 있으나 웹어셈블리가 호스트 자원에 접근하게 해주는 WASI (WebAssembly System Interface)의 보안 연구는 아직 부족한 상황이다. 이에 본 연구에서는 WASI를 지원하는 주요 런타임들을 비교, 분석하고 CLI 옵션을 처리하는 과정에서 발생하는 취약점을 문법 기반 퍼징으로 탐지한다. 각 런타임의 옵션 형식을 모델링하여 다양한 조합의 입력을 생성하고 실행하였으며 로그를 바탕으로 권한 우회 여부를 판단하였다. 그 결과로 네트워크 옵션 파싱 오류로 모든 IP 주소에 대한 접근이 허용되는 취약점(CVE-2025-54126)을 비롯한 여러 문제점들을 발견하여 제보하였으며, 이를 통해 런타임 옵션의 엄격한 검증과 WASI 표준 명세의 강화가 필요함을 강조한다.

I. 서론

최근 웹어셈블리(WebAssembly, Wasm)는 클라우드, 엣지/서버리스 컴퓨팅 같은 호스트 환경에서 중요한 기술로 부상하고 있다. 웹어셈블리를 이러한 호스트 환경에 도입하기 위해 웹어셈블리 바이너리가 호스트 자원에 접근하게 해주는 요소인 WASI (WebAssembly System Interface)가 등장했다. 몇몇 런타임들은 이를 구현하여 지원한다(예: Wasmtime [1], WAMR [2], WasmEdge [3], Wasmer [4]).

WASI는 권한 기반 보안 모델(capability-based security model)을 사용한다. 본 논문에서는 WASI를 지원하는 런타임들이 이러한 모델을 올바르게 구현하지 않으면 공격자에 의해 우회될 수 있다고 가정하였다. 이를 위해 WASI 명세와 Wasm 런타임들을 면밀히 분석하여 구현 상의 차이점을 파악하였다. 이후 WASI 권한 우회 취약점들을 자동으로 분석하고 발견하기 위한 문법 기반 퍼징 도구인 WaCFuzz를 개발하였다. 실제로 본 연구에서는 상이한 Wasm 런타임들에서 여러 취약점, 버그들을 발견했으며 이를 분석하고 제보하였다.

* 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. RS-2024-00457937)

† 교신저자(jinwookim@kw.ac.kr)

II. WASI 분석

2.1 WASI 명세

WASI 0.1 버전의 명세에는 API만이 존재하고 권한에 관한 설명이 존재하지 않는다. 0.2 버전에서는 권한에 대한 제안들이 존재하지만, 여전히 일반적인 동작이나 간단한 구현 설명이 주를 이루고 있다. 따라서 이 장에서 설명하는 내용은 WASI 0.2 버전을 기준으로 한다. 실제로는 본 연구에서 조사한 런타임 중에서 Wasmtime 만이 WASI 0.2 버전을 지원한다.

2.2 파일 시스템

WASI에서는 파일 시스템의 경로 해석을 기본 디렉터리와 상대 경로의 조합으로 수행한다. 절대 경로를 허용하지 않으며 전역 네임스페이스도 존재하지 않는다. 모든 경로는 기본 디렉터리를 기준으로 해석되며, 기본 디렉터리 외부의 파일 시스템에는 접근할 수 없다.

2.3 네트워크

웹어셈블리 프로그램은 네트워크 권한 없이는 소켓을 열 수 없고, 기본적으로 모든 연결이 거부되어야 한다. 그리고 접근 권한은 가능한 세부적으로 부여되어야 한다.

III. 웹어셈블리 런타임 분석

본 연구에서는 주로 파일 시스템, 네트워크, 환경 변수와 관련 있는 CLI (Command Line Interface) 옵션들을 중심으로 분석하였다.

3.1 파일 시스템

웹어셈블리 프로그램은 기본적으로 샌드박스 환경에서 실행되기 때문에 호스트 파일 시스템에 접근하려면, 접근하려는 디렉터리에 대한 권한을 명시적으로 설정해야 한다. 본 연구에서 조사한 런타임들에서는 특정 디렉터리에 대한 권한을 주는 기능과 실제 호스트 경로를 가상의 경로로 대응시켜주는 기능이 존재한다.

차이점으로는 Wasmtime, WAMR에서 여러 디렉터리에 권한을 부여할 수 있고 '..'과 심볼릭 링크에도 권한을 부여할 수 있다. Wasmer는 마찬가지로 여러 디렉터리에 권한을 부여할 수 있지만, 가상 경로를 지정하는 경우에만 '..'과 심볼릭 링크에 권한을 부여할 수 있다. WasmEdge는 앞서 말한 기능들을 지원하지 않고 권한을 부여한 디렉터리에 세부적으로 읽기 또는 쓰기 권한을 부여하는 기능이 존재한다.

3.2 네트워크

WASI 0.1 버전에서는 모든 소켓 API를 지원하지 않는다. 따라서, WASI 0.1 버전을 지원하는 런타임들은 소켓을 지원하기 위해서 추가적인 함수를 구현해야 하고 런타임 별로 구현이 다르므로 서로 호환이 되지 않는 문제점이 존재한다. 반면에 WASI 0.2 버전에서는 TCP, UDP, IP lookup 기능을 지원한다.

WAMR, Wasmer는 특정 주소에 권한을 부여하는 세부적인 네트워크 접근 제어를 제공한다. Wasmtime은 네트워크 기능을 허용할지 거부할지를 선택할 수 있고 주소 수준의 권한 설정은 불가능하다. 반면에 WasmEdge는 기본적으로 모든 네트워크 접근을 허용한다. 이는 WASI의 기본 거부 정책에 위배된다. 현재 WasmEdge는 WASI 0.1 버전만 지원하지만 0.2 버전을 지원하게 된다면 추가적인 옵션이 필요할 것으로 보인다.

3.3 환경 변수

마찬가지로 샌드박스 환경이기 때문에 초기에 어떤 환경 변수도 존재하지 않는다. 본 연구에서 조사한 런타임들에서는 변수명과 값을 가지는 쌍을 명시하는 것으로 해당 환경 변수를 웹어셈블리 프로그램으로 전달할 수 있다. 추가로, Wasmtime과 Wasmer는 모든 호스트 환경 변수를 전달하는 옵션이 존재한다. 또한, Wasmtime에서는 변수명만 명시된 경우, 해당 변수명을 가지는 호스트 환경 변수가 존재한다면 이를 전달하는 기능도 존재한다.

IV. WASI 취약점 분석

4.1 WaCFuzz 개요

런타임의 권한 기반 보안 모델의 구현을 검증하기 위해 문법 기반 퍼징(grammar-based fuzzing) 도구인 WaCFuzz를 설계하고 구현하였다.

해당 도구는 Fig. 1과 같이 세 가지 주요 요소로 구성된다. Generator는 각 런타임에 맞게 CLI 옵션 형식을 기반으로 만든 문법으로 올바르거나 변형된 옵션 조합을 생성한다. Executor는 앞서 만든 옵션을 이용해 웹어셈블리 프로그램을 목표 런타임에서 실행하고 표준 입출력, 오류 등을 수집한다. Oracle은 수집된 로그를 바탕으로 분석하여 권한이 우회되었는지를 판단한다.

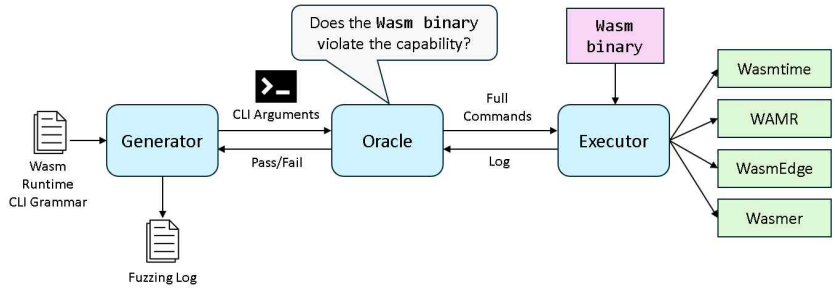

Fig.1 WASI 런타임 옵션 퍼징 테스트 도구 WaCFuzz 구조

퍼징 테스트를 통하여 조사한 런타임들에서 다양한 문제점들을 발견하였으며, 이하에서는 발견된 문제점 중 대표 사례를 서술한다.

4.2 네트워크 권한 구성 오류[5]

WAMR의 '--addr-pool' 옵션은 CIDR (Classless Inter-Domain Routing) 표기법을 인자로 받아 해당 주소 범위에 접근 권한을 부여한다. 마스크 없이 IP 주소만 입력되는 경우, 런타임은 이를 올바르게 처리하지 못하였다.

이 취약점을 설명하기 위해서 WAMR의 소켓 예제 프로그램[6]을 사용하였다.

```
$ iwasm --addr-pool=1.2.3.4 tcp_client.wasm
```

해당 실행에서 로컬 서버(127.0.0.1)가 열려 있다고 가정한다. 'iwasm'은 WAMR 런타임 바이너리 파일이고 'tcp_client.wasm'은 TCP를 사용해서 로컬 서버와 통신하는 프로그램이다. 위 경우에서 '--addr-pool' 옵션을 통하여 '1.2.3.4'라는 주소에만 권한이 부여되는 것을 예상했다. 하지만 마스크를 생략하는 경우 '/0'으로 설정되어 모든 연결이 허용되고 명시하지 않은 로컬 서버로의 연결이 가능하게 되었다.

4.3 심볼릭 링크에 권한 부여 시 혼동 가능성

심볼릭 링크에 '--dir' 옵션으로 권한을 부여하는 경우 권한이 부여된 디렉터리의 파일에 접근하는 것 자체는 문제가 없는 동작이나, 권한을 부여하려는 디렉터리를 사용자 또는 공격자가 임의로 설정 가능하다는 점에서 보안상 위험 요소가 될 수 있다. 따라서 런타임이 심볼릭 링크에 접근 권한을 부여할 때 해석된 실제 경로를 사용자에게 명확히 알려주거나 허용 여부를 확인하도록 하여 사용자가 의도치 않은 권한 부여를 방지하도록 해야 한다.

V. 결론

본 연구에서는 WASI를 지원하는 여러 웹어셈블리 런타임의 권한 기반 보안 모델을 조사, 비교, 분석하고 퍼징을 통해 다양한 CLI 입력을 생성하여 런타임의 보안 모델을 우회할 수 있는지를 검증하였다. 그 결과, 런타임 간의 옵션 및 권한의 범위와 세부적인 구현에 있어서 일치하지 않는 경우를 발견하여 권한의 차이가 실제로 존재함을 확인하였고 WaCFuzz를 통해 취약점과 버그를 발견하여 제보하였다.

이로부터 WASI 명세가 더 논의되어서 정확한 명세가 필요하다는 것과 CLI 옵션 등의 외부 입력에 대한 엄격한 구문 분석과 검증이 필요하다는 점을 지적한다. 다만, 본 연구는 특정 런타임과 문법 기반 퍼징 전략에 초점을 맞추어 실험 범위가 제한되었으므로, 향후에는 더 넓은 WASI API 영역과 복합적인 옵션, 환경 조합에서의 취약점 탐지와 동적 환경에서의 실질적 공격 시나리오 분석을 통해 연구 범위를 확장할 필요가 있다.

[참고문헌]

[1] bytecodealliance, "wasmtime", https://github.com/bytecodealliance/wasmtime, 2025

[2] bytecodealliance, "WAMR", https://github.com/bytecodealliance/wasm-micro-runtime, 2025

[3] WasmEdge, "WasmEdge", https://github.com/WasmEdge/WasmEdge, 2025

[4] wasmerio, "wasmer", https://github.com/wasmerio/wasmer, 2025

[5] bytecodealliance, "`--addr-pool` option allows all IPv4 addresses when subnet mask is not specified", https://github.com/bytecodealliance/wasm-micro-runtime/security/advisories/GHSA-vh64-mfvw-pxqp, 2025

[6] bytecodealliance, "socket-api sample", https://github.com/bytecodealliance/wasm-micro-runtime/tree/main/samples/socket-api, 2025