거대 언어 모델을 사용한 비 학습 함수 이름 예측

김진환[1], 조영필[2]
[1]한양대학교 컴퓨터소프트웨어학과 (미래자동차-SW 융합전공) 석박통합과정
[2]한양대학교 컴퓨터소프트웨어학과 교수
adsll156@hanyang.ac.kr., ypcho@hanyang.ac.kr

Predicting Non-Learning Function Names Using Large Language Models

Jin-Hwan Kim[1], Yeong-Pil Cho[2]
[1]Department of Computer and Software (Automotive-Computer Convergence), Han-Yang University
[2]Dept. of Computer Science, Han-Yang University

요 약

심볼 정보가 제거된(stripped) 바이너리에서 학습 없이(비 파인튜닝) LLM을 활용해 함수 이름을 예측하는 프롬프트·파이프라인을 제안한다. Angr를 이용해 사람이 읽기 쉬운 심볼릭 표현식을 만들고, LIBC/데이터 흐름(DF)/제어 흐름(CF)/도메인 모듈이 각각 추론한 요약을 결합한 뒤, 피드백 루프(C2C, GV) 로 맥락을 보강해 최종 이름을 생성한다. 평가에서는 NERO 데이터셋의 바이너리(일부)로 사례 비교를 수행하고, 짧은 함수명을 공정하게 비교하기 위해 WRDScore(CodeBERTScore 임베딩 기반 토큰 유사도)라는 지표를 사용했다. 이 결과로 일부 예제에서 gpt-20b가 gpt-120b보다 높은 점수를 보였다. 거대 언어 모델에게 적절한 단서를 주면 학습하지 않더라도 함수 이름을 충분히 예측할 수 있음을 알 수 있다.

1. 서론

악성코드, 취약점 탐지나, 신뢰할 수 없는 바이너리 분석과 같이 바이너리 역공학에 대한 노력과 시도는 계속되고 있다. 특히 악성코드나 랜섬웨어 바이너리처럼 신뢰할 수 없는 바이너리를 분석하거나 그 목적을 분석하기 위해서 다양한 역공학 방법과 작업 및 연구가 지속되고 있다. 그러나 악성코드나 알려진 바이너리들은 대체로 심볼 정보가 제거되어 있으며, 이렇게 심볼이 제거된 바이너리는 각 함수의 영역을 구분지을 수 없어 전문적인 도구의 도움이 필요하다. 이에 대한 노력으로 IDA[5] 및 Ghidra[6]등의 전문화된 디컴파일러 도구 들이 심볼 정보가 제거된 바이너리의 제어 흐름 및 데이터 흐름 분석을 지원한다. 하지만, 이는 구조화를 쉽게 도와주는 것 이며 심볼 정보가 제거된 바이너리에 대해 완전한 정보를 제공해 주는 것은 아니다. 바이너리 내의 함수가 무엇을 하는지, 어떠한 목적의 바이너리인지를 알아내기 위해서는 역공학 전문가와 그에 대한 분석 시간이 반드시 필요하다.

또한 디컴파일 도구를 통해 함수를 구분 짓는다 하더라도 각 함수가 무엇을 하는지, 어떤 역할을 수행하는지를 알아내기 위해서는 제공된 구조를 바탕으로 제어 흐름과 데이터 흐름 분석 및 호출되는 라이브러리 등을 통해 추가로 분석을 수행해야 한다. 이를 알아내기 위해 외부 라이브러리를 분석하는건 정보가 한정적이며, 어셈블리어로 이루어진 내부는 레지스터와 함수별 스택 정보가 시시각각 변하게 되므로 분석에 오랜 시간이 걸리며, 특히 심볼 정보가 없는 함수가 대량으로 존재하는 경우 그 분석 시간은 더 길어진다. 이처럼 심볼 정보가 제거된 바이너리로부터 그 바이너리 내의 함수 정보를 쉽게 알아낼 수 있는 방법으로는 그 함수의 명칭을 빠르게 파악하는 것이다. 함수의 이름은 곧 그 함수의 추상화된 목적을 담고 있으므로, 내부의 모든 코드를 분석하지 않더라도 대략적으로 그 함수가 무슨 역할을 수행하는지 알 수 있게 된다.

함수 이름을 예측하는 연구는 AsmDepictor[1], SymLM[2], SymGen[3]을 비롯해 LLM과 기존 트랜스포머를 결합[4]하여 연구가 진행중에 있다. 이처럼 심볼 정보가 제거된 바이너리로부터 함수 이름을 예측하기 위해 머신 러닝을 통한 학습 및 언어 모델등을 사용해 이러한 목표를 달성하려 한다. 또한 거대 언어 모델의 발전으로 인해 그 적용범위가

점차 확대대고 있어, 다양한 연구에서도 활용되고 있다.

본 연구는 심볼 정보가 제거된 함수로부터 역공학을 통한 함수 이름 예측을 수행한다. 거대 언어 모델(LLM)을 사용해 심볼이 제거된 바이너리에서 학습 없이도 최대한 근접한 이름을 생성하고, 보다 LLM이 생성한 함수 이름에 대해 전반적인 검토를 수행한다.

2. 관련 연구

바이너리 리버스엔지니어링 분야에서 함수명 복원 및 생성은 프로그램 이해와 분석을 위한 핵심 과제로 인식되어 왔다. 본 절에서는 스트립된 바이너리로부터 의미 있는 함수명을 추론하거나 생성하는 기존 연구들을 검토하고, 각 접근법의 장단점을 분석한다.

AsmDepictor [1]는 어셈블리 코드를 함수명으로 변환하는 과정을 언어 번역 작업으로 접근한 최초의 Transformer 기반 sequence-to-sequence 모델이다. 이 연구는 디스어셈블된 명령어 시퀀스를 토큰화하고, 리터럴과 주소를 일반화하며, 필요시 호출 시그니처를 정제하여 입력으로 사용한다.

SymLM [2]은 함수의 실행 동작(micro-traces)과 호출 컨텍스트를 모두 학습하는 컨텍스트 인식 임베딩 기법을 제안한다. 이 모델은 새로운 신경망 인코더를 사용하여 함수 동작과 호출 컨텍스트를 융합하여 함수명을 예측한다. 입력 표현으로는 어셈블리 코드 시퀀스와 함께 프로시저 간 제어 흐름 그래프(CFG)와 마이크로 트레이스 특징에 기반한 호출 관계 및 실행 경로 정보를 활용한다.

SymGen [3]은 함수명 생성을 분류 문제가 아닌 자동회귀 생성 문제로 정의한 최초의 프레임워크이다. 사전 학습된 LLM을 도메인 적응(LLM 생성 요약 사용)과 경량 파인튜닝(LoRA 등)을 통해 바이너리 코드 요약에 적용한다. 디컴파일된 코드와 LLM이 생성한 함수 요약 문장을 결합하여 입력으로 사용하며, 추론 시에는 디컴파일된 함수 코드가 주요 입력이 된다.

3. 설계

별도의 학습이 없이 오로지 사전 학습된 LLM만을 통해 이름을 추론해야 하므로 프롬프트 설계를 수행 한다.

3.1 심볼릭 표현식

바이너리에 angr[7]를 사용해 심볼릭 표현식을 생성한다. 보통의 심볼릭 표현식은 SMT-Solver의 형태를 가진 수식의 형태이다. 본 연구의 심볼릭 표현식은 기호화된 표현식으로, 단순히 바이너리의 흐름을 기호로 나타내는 형식이다. 이는 어셈블리어보다 많은 의미를 담고 있으면서, 인간이 이해하기 쉬운 형태로 보여 모델 또한 직관적인 이해가 가능하다.

3.2 추론 및 생성 모듈

추론 모듈은 LIBC 등의 기본 라이브러리 함수의 인자 정보를 제공하는 LIBC 모듈, 함수 내의 데이터 흐름을 분석후 그 데이터 흐름의 목표를 추론하는 DF 모듈, 제어 흐름의 목적을 추론하는 CF 모듈, 해당 함수가 어떤 영역에서 수행되는지를 추론하는 도메인 모듈로 구성된다. 생성 모듈은 추론 모듈 내부의 4개의 모듈들이 생성한 결과를 넘겨받아 그 내용을 바탕으로 최종 함수 명을 추론한다.

3.3 피드백 모듈

초기의 결과는 오로지 4개의 추론 모듈로만 구성되어 출력되고 미 추론 함수들에 대해 이름을 모두 업데이트 한다. 업데이트된 결과가 반영되고, 각 함수의 Caller와 Callee 관계를 파악하는 C2C 모듈, 포인터 변수들에 대해 정보를 정리하는 GV 모듈 2개가 추가로 구성되며, C2C모듈의 결과는 기존 심볼릭 표현식에 더해져 CF 모듈의 인풋으로 전달된다. GV모듈은 기존 심볼릭 표현식에 더해져 DF 모듈의 인풋으로 전달되며, CF,DF 모듈은 위의 두 정보를 추가로 받아 새로운 요약 정보를 생성하고, 결과를 생성 모듈로 전달한다

3.4 생성 모듈

생성 모듈은 앞의 추론 모듈과 피드백 모듈로부터 전달된 정보를 받아 분석해 역할을 가장 잘 나타낼 수 있는 함수명을 생성한다. 이때 함수의 구체적인 동작이나 목적을 담아내면서도 간결한 이름이 되도록, 앞서 제공된 라이브러리 호출 내용, 데이터/제어 흐름 요약, 도메인 추론 결과를 모두 활용한다. 또한 이름이 선정된 이유를 명확히 하기 위해, 생성된 이름과 함께 해당 이름을 뒷받침하는 설명 문장도 함께 작성한다. 예를 들어 생성 모듈이 어떤 함수를 "Init_config"라고 이름짓고자 한다면, 설명 부

분에 "이 함수가 전역 설정값들을 읽어와 초기화하는 역할을 수행하기 때문에 Init_config 명명함"과 같은 근거를 달아주는 식이다. 최종 출력으로는 각 함수마다 예측된 함수명과 간략한 설명이 쌍으로 제공된다.

4. 실험 및 평가

본 목적은 별도의 학습 없이(Zero-shot)도 함수 이름을 적절하게 추론해 내어 역공학 사용자에게 올바른 정보를 제공하는데 있다. 따라서, 대규모 파라미터를 가진 공개 LLM 모델들을 중심으로 실험을 수행한다. 공개된 모델에 별도의 학습을 수행하지 않고, 그 의미를 추론하므로, 그 단어를 정확하게 복원하기는 실제로 어렵다. 또한 함수명은 통일된 문법이 없고, 동일 기능에도 작명 다양성이 크다.

이처럼 원본 함수 이름을 완전 복원시키는 것이 불가능하므로, 문자열 일치 지표로는 유용성이 떨어진다. 또한 대부분의 의미 측정 지표들도 장문 자연어 텍스트에 기반한 의미 비교에 초점이 맞춰져 있어, 그보다 짧은 함수 이름에서 나타나는 의미적 유연성을 포착하는데 한계가 있다.

이에 우리는 모델이 예측한 함수명에 대해 의미를 비교하기 위해 짧은 문장 길이·표현 변화에 비교적 강건하며, 문장 내 단어들 간의 유사성을 측정할 수 있는 WRDScore[12]라는 지표를 사용해 평가했다. WRD[14]를 활용한 WRDScore가 자바 메서드명 예측에 사용된 지표임을 감안하면, 해당 지표를 사용하는 것이 현재 실험 평가에 적절함을 알 수 있다.

실험 데이터는 NERO[8] 데이터셋에 제공된 바이너리 중 하나를 선택한 단일 사례 연구로 실험을 진행하였다. 사용한 모델은 gpt-oss:120b와 20b[10], llama3.1:7b[11] 모델을 사용해 실험을 진행했다. WRDScore를 단순 지표로서 활용하므로, 코드 중심으로 학습된 모델인 CodeBERTScore[13]의 임베딩과 토크나이저를 사용해 해당 지표를 측정하였다. 해당 결과는 표 1에 표시했다.

원본 함수 이름	gpt-120b	gpt-20b	llama3.1:7b
tty_set_last_char_flag	set_tty_last_char_flag (0.9444)	tty_set_last_char_flag (1.0)	___get_tty_last_char_flag (0.9685)
xmemdup	duplicate_buffer (0.7891)	memdup (0.9742)	allocate_and_copy_buffer (0.8053)
tty_set_interrupt_char	___set_tty_interrupt_char (0.9301)	set_tty_interrupt_char (0.9301)	set_terminal_interrupt_char (0.9235)
tty_key_list_insert	insert_sorted_global_string_node (0.9154)	insert_sorted_string_node (0.9125)	buffer_copy_verify (0.8831)
get_login_name	___initialize_login_name (0.9377)	get_current_user_login (0.951)	___validate_login_name (0.9436)

<표 1> LLM기반의 함수 이름 예측 결과, 괄호의 수치는 WRDScore F1 점수 결과

측정 결과, 전반적으로 의미가 비슷하다고 평가된 함수들은 대체로 WRDScore F1 0.75 이상 범위에 분포했다. 예상외로 gpt-20b가 gpt-120b에 비해 더 높은 점수를 받았다. 모델이 클수록 더 많은 프롬프트 정보를 받는건 사실이나, 특정 부분에 집중할 수 있게 만든 gpt-20b의 소형 모델이 더 높은 점수를 받았다고 추측된다. 다만, 실험이 단일 바이너리에 초점이 맞춰져 있으므로 해당 결과만 가지고 판단하긴 어렵다. llama3.1:7b에서도 대체로 높은 점수를 기록 했으나 gpt-20b 만큼의 성능이 나오진 않았다.

비 학습된 모델이므로 다른 연구들 처럼 함수 이름을 정확하게 맞출수는 없었다. 그러나 임베딩 공간내의 유사도를 측정한 점수는 대체로 높은 점수를 기록했다. 하지만 전혀 관련이 없는 일부 토큰들 역시 0.7~0.8 수준의 높은 점수를 기록했다. 이는 WRDScore의 계산 방식으로 인해 발생한 것으로, 임베딩 공간내 문제로 보인다.

5. 결론

본 연구는 대규모 파라미터를 가진 공개 LLM 모델을 학습시키지 않고 함수 이름을 맞추는 연구를 수행했다. 생성된 함수 이름에서 볼 수 있듯, 어떤 부분에선 학습되지 않았음에도 같게 맞추거나, 그 의미가 미묘한 차이를 가질 뿐, 심볼릭 표현 기반으로 해당 함수 가 무엇을 하는지, 그 의미를 추론할 수 있음을 보였다. 다만, 현재로써 실험 표본이 단일 바이너리뿐이며, 대량의 함수명을 평가한게 아니므

로 일반화는 신중해야 하며, WRDScore 임계값 0.75는 본 데이터에서 관찰된 경험적 기준(heuristic) 이다. 또한 WRDScore에 결합해 함수 이름 길이나 구조를 반영하여 별도의 측정 지표가 후속 연구로써 필요할 것으로 보인다.

이 논문은 과학기술정보통신부의 재원으로 정보통신기술 기획평가원(IITP)의 (연구과제번호 RS-2024-00337414 , SW공급망 운영환경에서 역공학 한계를 넘어서는 자동화된 마이크로 보안 패치 기술 개발)지원을 받아 수행된 연구임.

참고문헌

[1] Hyunjin Kim, Jinyeong Bak, Kyunghyun Cho, and Hyungjoon Koo. A Transformer-Based Function Symbol Name Inference Model from an Assembly Language for Binary Reversing. Proceedings of the 18th ACM ASIA Conference on Computer and Communications Security (ASIACCS), Melbourne, Australia, 2023, pp. 951–965.

[2] Xin Jin, Kexin Pei, Jun Yeon Won, and Zhiqiang Lin. SymLM: Predicting Function Names in Stripped Binaries via Context-Sensitive Execution-Aware Code Embeddings. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS), Los Angeles, CA, USA, 2022, 15 pages

[3] Linxi Jiang, Xin Jin, and Zhiqiang Lin. Beyond Classification: Inferring Function Names in Stripped Binaries via Domain Adapted LLMs. Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, 2025, 18 pages

[4] Remus M. Petrache and Camelia Lemnaru. A Hybrid Transformer-LLM Pipeline for Function Name Recovery in Stripped Binaries. Proceedings of the 2025 IEEE International Conference on Cyber Security and Resilience (CSR), Chania, Crete, Greece, Aug. 2025.

[5] Hex-Rays. IDA Pro – Powerful Disassembler, Decompiler & Debugger. (Software Tool) – Official Website: https://hex-rays.com/ida-pro.

[6] National Security Agency (NSA). Ghidra – Open-Source Software Reverse Engineering Suite. (Software Tool) – Official Website: https://ghidra-sre.org.

[7] UC Santa Barbara Security Lab. angr – Open-Source Binary Analysis Platform. (Software Framework) – Documentation and Source: https://angr.io.

[8] Y. David, U. Alon, and E. Yahav, “Neural reverse engineering of stripped binaries using augmented control flow graphs,” Proceedings of the ACM on Programming Languages, vol. 4, no. OOPSLA, pp. 1–28, 2020.

[9] OpenAI. Introducing OpenAI o3 and o4-mini. (Model Release Blog) – April 16, 2025. Available online: https://openai.com/index/introducing-o3-and-o4-mini/.

[10] OpenAI. Introducing gpt-oss 120B. (Model Release Blog) – August 5, 2025. Available online: https://openai.com/index/introducing-gpt-oss/

[11] Katie Paul. Meta unveils biggest Llama 3 AI model, touting language and math gains. Reuters News, July 23, 2024

[12] Ravil Mussabayev, “WRDScore: New Metric for Evaluation of Natural Language Generation Models”, 2024 20th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS 2024), Novosibirsk, Russia, 2024, 20–23 pages

[13] Shuyan Zhou, Uri Alon, Sumit Agarwal, Graham Neubig, “CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code”, Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), Singapore, 2023, 13921–13937 pages

[14] Yokoi, Sho, et al. "Word rotator's distance." arXiv preprint arXiv:2004.15003 (2020).