2021년 한국정보보호학회 동계학술대회

CISC-W'21

Conference on Information Security and Cryptography-Winter 2021

2021년 11월 27일 (토)

온라인 컨퍼런스

※개회식, 최우수논문 및 우수 기관장상 논문 발표 촬영 실시간 중계: 가천대학교 글로벌캠퍼스 가천관 B101호

주최 한국정보보호학회 Korea Institute of Information Security & Cryptology

주관 가천대학교 Gachon University

후원 국가정보원 NATIONAL INTELLIGENCE SERVICE, 과학기술정보통신부, 행정안전부, KISA 한국인터넷진흥원, ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, NSR 국가보안기술연구소 National Security Research Institute, LG 히다찌, 쌍용정보통신주식회사 SsangYong Information & Communications Corp.

한국정보보호학회
Korea Institute of Information Security & Cryptology

방어 메커니즘 우회를 위한 CAGL 기반 난독화 컴파일러 디자인

주재경*, 박기웅†

* 세종대학교 시스템 보안 연구실, † 세종대학교 정보보호학과

Design of Obfuscation Compiler based CAGL for Defense Mechanism

Jaegyeong Ju*, Ki-Woong Park†

* System Security Laboratory, Sejong University

† Department of Computer and Information Security, Sejong University

요 약

방어 메커니즘을 우회하여 타겟 시스템에 익스플로잇 하기는 쉽지 않다. 이를 위해서는 방어 메커니즘을 우회하기 위한 취약점과 공격 기술에 대한 포괄적인 이해가 필요하다. 따라서 공격자는 기존 공격 방식을 재사용하는 것을 선호한다. 하지만 방어 메커니즘 및 보안 정책의 발전으로 이러한 공격은 대개 사전에 차단된다.

이에 본 논문에서는 공개된 악성코드를 활용해 컴파일 과정에서 기존 코드를 랜덤화된 구조로 변경하고 실행 가능한 파일로 변환한 뒤 메모리에 적재해 실행함으로써 방어 메커니즘을 우회하는 CAGL(Compile And Go Loader) 기반 난독화 컴파일러 기술을 제안한다.

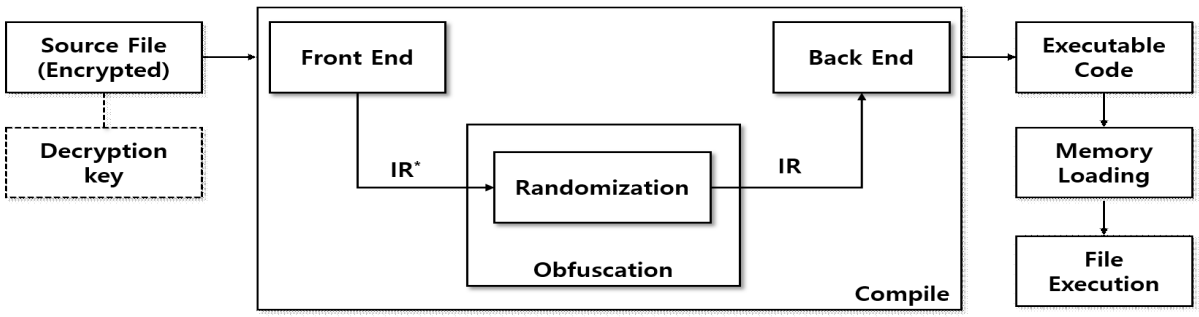
I. 서론

타겟 시스템을 익스플로잇 하기 위해서는 방어 메커니즘을 우회해야 한다. 공격자는 사전에 정찰을 통해 타겟 시스템의 정보를 수집하고, 수집한 정보를 기반으로 공격 시나리오를 구성해 익스플로잇을 수행하는 일련의 과정을 통해 방어 메커니즘을 우회해 타겟 시스템에 침투하여 시스템을 감염 및 원하는 정보를 추출한다[1].

하지만 타겟 시스템을 익스플로잇 하기 위해서는 관련 취약점과 공격 기술에 대한 이해가 필요하므로 그에 따른 시간과 비용이 증가한다. 이에 따라 공격자는 기존 공격 방식을 재사용하여 시간과 비용을 절약하는 방법을 선호하지만, 방어 메커니즘과 보안 정책의 발전으로 공개된 취약점을 통한 익스플로잇은 대개 사전에 차단된다.

이에 본 논문에서는 기존에 공개된 악성코드를 활용하여 방어 메커니즘을 우회해 타겟 시스템에 익스플로잇할 수 있는 기술을 제안한다. 공개된 악성코드를 컴파일하는 과정에서 기존 코드를 랜덤화된 구조로 변경, 실행 가능한 파일로 변환한 뒤 메모리에 적재해 실행시키는 CAGL[2]기반 난독화 컴파일러를 디자인한다. 해당 기술은 공격 관점에서 공격 테스트 시나리오내지, CTF(Capture-The-Flag)에 활용할 수 있다. 또한, 방어 관점에서 방어 메커니즘이 탐지하지 못한 원인 분석에 표본을 제공하므로 성능 향상에 도움을 줄 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 CAGL 기반 난독화 컴파일러가 동작하기 위한 가정 및 해당 기술디자인에 필요한 요구사항을 정의하고, 이를 바탕으로 해당 컴파일러를 디자인을 수행한다. 이어서 3장에서는 결론 및 향후 연구를 기술한다.



[그림 1] CAGL 기반 난독화 컴파일러 구조

II. CAGL 기반 난독화 컴파일러

본 장에서는 1장에서 제안한 기술을 디자인함에서 필요한 사항을 살펴보고, 이를 바탕으로 CAGL 기반 난독화 컴파일러를 디자인한 내용을 소개한다.

2.1. 가정

제안하는 기술은 Exploit-DB[3] 등 익스플로잇 코드를 PoC 형태로 제공하는 사이트를 참고하며, 해당 코드를 컴파일 및 실행은 모두 엔드포인트 단에서 일어나는 것만 고려한다. 또한, 해당 호스트에는 최소한의 방어 메커니즘인 안티바이러스가 동작 중이나 사용자의 부주의로 Trojan-Downloader[4] 통해 공격자 서버로부터 시스템 내부로 익스플로잇 코드와 난독화 컴파일러가 다운로드되어 최종적으로 해당 호스트 PC 내부에 존재하고 있는 상황을 가정한다.

2.2. 요구사항

앞 절에서 가정한 상황을 바탕으로 향후 CAGL 기반 난독화 컴파일러 디자인을 위한 요구사항은 다음과 같다.

- 안티바이러스의 실시간 탐지 고려: 앞 절의 가정에서 익스플로잇 코드와 난독화 컴파일러가 있는 호스트 환경에 안티바이러스가 동작 중이므로 실시간 탐지에 따른 악성파일로 제거되는 것을 미연에 방지하기 위해 익스플로잇 코드를 암호화하여 해당 소스 코드 파일을 보호한다.
- 컴파일 및 실행을 위한 권한 획득: CAGL을 사용하는 목적은 일반적으로 호스트 PC에서 프로그램 실행 시 동작하는 로더가 아닌 공격자 서버로부터 다운로드된 별도의 로더인 CAGL를 통해 익스플로잇 코드를 실행 가능한 파일로 컴파일하고 메모리에 적재시켜 프로그램을 실행할 경우, 기존 안티바이러스의 감시 경로에서 벗어나 방어 메커니즘으로부터 탐지되기까지의 시간을 지연시키고자 한다. 상기 시나리오를 수행하기 위해서는 컴파일 및 로더 수행을 위한 최소한의 권한을 획득해야 한다.
- 컴파일 시 소스 파일의 구조 랜덤화: 익스플로잇 코드(소스 파일)를 컴파일하는 과정에서 파일 내부의 구조를 실행은 가능하나 분석하기 어렵도록 컴파일할 때마다 구조를 랜덤으로 변경하는 알고리즘을 추가한다. 이는 안티바이러스 등 방어 메커니즘이 디컴파일을 통해 익스플로잇 코드가 탐지되지 않도록 하기 위해 필요한 부분이다.

2.3. 설계

본 절에서는 앞에서 정의한 가정과 요구사항을 기반으로 설계한 CAGL 기반 난독화 컴파일러 구조는 [그림 1]과 같다. 암호화된 익스플로잇 코드(소스 파일)을 컴파일을 위해 먼저 복호화키를 사용해 해당 파일을 읽을 수 있는 파일로 복호화한 뒤 일반적인 2단계(two-phase) 컴파일러를 통해 실행 가능한 코드로 만드는 과정에서 Front End가 끝나고 생성된 IR을 Back End로 보내기 전에 내부 구조를 랜덤으로 변경하는 알고리즘을 통해 안티바이러스의 분석은 우회하고 실행은 가능한 파일을 생성한다. 그 후 해당 파일은 별도로 구현한 로더를 통해 메모리에 적재된 후 실행됨으로써 타겟 시스템에 따른 익스플로잇을 수행한다. 해당 구조는 LLVM 아키텍처[5]를 기반으로 설계한 부분이다.

III. 결론

본 논문은 엔드포인트 단에서 안티바이러스(방어 메커니즘)를 우회하기 위해 익스플로잇 코드를 실행(익스플로잇)하기 위한 방안을 모색한다. 그에 따라 암호화된 익스플로잇 소스코드 파일을 파라미터로하여 컴파일 및 실행을 수행하는 CAGL 기반 난독화 컴파일러를 제안한다. 향후 연구로는 디자인한 내용을 토대로 CAGL 기반 난독화 컴파일러를 구현 및 확정할 예정이다.

[참고문헌]

[1] Lockheed Martin. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[2] Compile And Go Loader. https://techblogmu.blogspot.com/2018/05/explain-compile-and-go-loader.html

[3] Exploit-DB. https://www.exploit-db.com/

[4] Trojan-Downloader. https://www.f-secure.com/v-descs/trojan-downloader.shtml

[5] Lattner, Chris, and Vikram Adve. "LLVM: A compilation framework for lifelong program analysis & transformation." International Symposium on Code Generation and Optimization, 2004. CGO 2004.. IEEE, 2004.