API 호출 분석을 이용한 패커 분류

허태광[1], 김연어[1], 천준석[1], 류샤오[1], 변석우[2], 우균[1]
부산대학교 전기전자컴퓨터공학과[1], 경성대학교 소프트웨어학과[2]
e-mail: {vhxpffltm,*yeoneo,jscheon,liuxiao,woogyun*}*@pusan.ac.kr*[1], *swbyun@ks.ac.kr*[2]

Packer Classification Using API Call Analysis

Taekwang Hur[1], Yeoneo Kim[1], Junseok Cheon[1], Xiao Liu[1], Sugwoo Byun[2], Gyun Woo[1]
Dept. of Electrical and Computer Engineering, Pusan National University[1],
Dept. of Computer Science, Kyungsung University[2]

Abstract

Packing and obfuscation tools are widely used to protect the copyright of programs. However, packing and obfuscation are being exploited to distribute malicious code by hiding in a program. This paper analyzes the entry point of the packed program and the API call at execution time and then classifies the packer by obtaining the similarity of the API call. As the result of the experiment, the similarity between UPX, Mpress, and Aspack was high, and they can be classified into similar packers.

I. 서론

최근 해킹 사례와 더불어 프로그램 분석을 힘들게 하는 패킹 및 난독화 도구가 많이 사용되고 있다. 이는 본래 프로그램의 저작권을 보호하고 배포 파일의 크기를 줄이는 데 사용되지만, 악성 공격자들이 프로그램에 악의적인 코드를 숨겨 배포하는 형태로 악용하고 있다. 공격 코드가 쉽게 드러나지 않도록 악성 코드의 탐지 회피 및 보호 수단으로 사용하는 사례가 늘어나고 있으며 패커는 쉽게 프로그램에 적용할 수 있고 분석하기 어려우므로 문제가 되고 있다.

최근에는 패킹된 실행 파일을 그냥 배포하지 않고 변형하여 배포하고 있다. 기존의 패킹 여부를 판단하는 방법은 실행 파일에서 특정 이진 코드를 저장하거나 PE 헤더의 세션 정보를 저장하여 이용하는 방법을 사용하고 있지만, 이 방법은 새로운 패커로 패킹하거나 PE 헤더의 정보를 변경한 경우 탐색하기가 힘들다.

그래서 특정 악성 코드나 패커에 의존적이지 않은, 독입적인 탐색 방법이 필요하다. 본 논문에서는 프로그램을 실행하여 API를 추출하여 패킹된 프로그램의 OEP(Original Entry Point)를 찾고 실행할 때 호출되는 API를 분석하여 어떤 패커를 사용했는지 분류하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 API 호출과 관련된 연구를 소개하고 3장에서 제안 방법에 대한 이론과 과정을 설명한다. 이후 4장에서 실험 과정과 결과를 설명하고 5장에서 실험 결과에 대해 고찰한다. 마지막으로 6장에서 결론을 맺는다.

II. 관련연구

Hartung의 연구는[1] 난독화된 닷넷 파일 분석에서 분석하기 힘든 런타임 기술에 직면했을 때, 분석 방법을 소개한다. 패커 암호 해독 또는 API 사용 후 문자열을 가져오는 간단한 기술뿐만 아니라 정교한 예제를 설명한다. 이것을 바탕으로 닷넷 샘플을 분석하기 위한 강력한 도구를 설계하는 데 도움이 될 수 있음을 보여주고 있다.

J.Y.C. 등의 연구는[2] 정보 검색 이론을 이용하여 악성 실행 파일을 분류하기 위한 새로운 접근법을 제안한다. Windows API 함수 호출 시퀀스를 동적으로 분석하면 해당 특성 표준의 TF-IDF 가중치 체계에 대한 입력으로 사용되어 악성 실행 파일을 식별하는

방법을 제안한다. 관련성이 없는 기능을 걸러내고 악성 실행 파일 분류의 정확성을 향상하는 결과를 보여준다.

Shank 등의 연구는[3] 커널 컴퓨터를 사용하여 자동 공격 및 침입으로 이어질 수 있는 악성 코드를 기능적으로 분류하는 방법을 제안한다. 정적 탐지 기법에는 제한이 있으며 Windows API 호출 시퀀스를 기반해 악성 코드를 분류하는데 정확성이 높으며 매개 변수 값에 의존적이므로 SVM에서 더 좋은 결과를 나타낸 것을 보여준다.

이들 연구는 API 호출 시퀀스를 분석할 때, 동적 분석의 한계로 실행 파일이 안티 디버깅 등으로 분석할 수 없는 경우 분류할 수 없다. 또한, 새로운 악성 실행 파일의 새로 분류할 수 없는 한계가 있다.

Ⅲ. 제안 방안

3.1 제안 방법

그림 1은 본 논문의 제안 방법을 나타낸다. OEP를 분석하기 위해 패킹과 언패킹 과정의 동작을 분석하고 API 추출에 Pin tool을 사용한다.

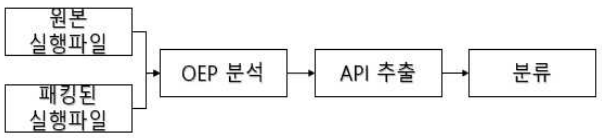

그림 1 제안 방법

분류를 위해 서열 정렬인 Smith-Waterman[4] 알고리즘을 이용하여 유사도를 계산한다. 이 알고리즘은 지역 정렬을 수행하는 알고리즘으로써 유전자 또는 단백질 사이의 관계를 보여줄 수 있다. 또한, 프로그램 유사도 검사에도 적용할 수 있다.

3.2 패킹, 언패킹

패킹은 프로그램의 실행 형식을 유지한 채로 파일을 압축하거나 난독화하는 방법을 뜻한다. 패킹은 있는 그대로 일반 프로그램처럼 실행 가능하며 실행 파일을 대상으로 파일 내부에 압축 해제코드를 포함하고 있어 실행되는 순간 메모리에서 압축을 해제시킨 후 실행시키는 기술이다. 패킹된 프로그램은 분석을 어렵게 하고 디버깅 지점을 찾기 힘들다.

패킹된 프로그램은 운영체제의 로더가 실행 파일을 메모리에 로딩하면 진입점(entry point)로부터 프로그램이 실행되며 패킹된 파일의 진입점은 언패킹 코드 영역에 들어 있다. 이 언패킹 코드는 패킹된 데이터를 하나씩 읽어 압축을 풀고 원본 데이터를 저장하고 모든 코드가 언패킹 되면 OEP에서부터 프로그램을 다시 시작한다. 패킹을 위한 도구로는 UPX, Aspack 등의 도구들이 있다.

패킹된 프로그램을 분석하기 위해서는 언패킹이 필요하다. 언패킹은 자동으로 할 수도 있고 수동으로 진행할 수 있다. 자동으로 진행하기 위해서는 각 패킹 도구의 언패킹 도구를 찾아서 사용하거나 PEID의 플러그인을 사용할 수 있다. 수동으로 언패킹을 하기 위해서는 디버거를 사용하여 OEP를 찾고 메모리를 덤프하여 IAT(Import Address Table)을 복구하여 언패킹을 수행할 수 있다.

그림 2는 패킹된 프로그램의 OEP 주소로 점프하는 명령어를 보여준다. OEP까지의 API 추출 시 프로그램 시작 지점을 판별하여 API 호출을 분석할 수 있다.

```
000AA8BE   . 61             POPAD
000AA8BF   . 8D4424 80      LEA EAX,DWORD PTR SS:[ESP-80]
000AA8C3   > 6A 00          PUSH 0
000AA8C5   . 39C4           CMP ESP,EAX
000AA8C7   .^75 FA          JNZ SHORT C_x86_up.000AA8C3
000AA8C9   . 83EC 80        SUB ESP,-80
000AA8CC   .-E9 AA72FFFF    JMP C_x86_up.000A1B7B
```

그림 2 OEP로의 점프 명령어

3.3 API 추출 및 유사도 계산

패킹된 프로그램의 OEP까지 사용된 API를 추출하는 과정이 필요하다. Pin Tool[5]은 DBI(Dynamic Binary Instrumentation)로 동적으로 실행 중인 바이너리를 분석하는 도구이다. 빌드한 도구를 사용하여 실행 중의 프로그램 동작을 추적할 수 있다. 여기서는 Pin tool을 사용하여 실행 파일과 패킹된 실행 파일의 API를 추출한다. 서열 정렬을 통해 추출한 모든 API 호출의 유사도를 계산하여 패커를 분류한다.

Ⅳ. 실험

실험은 C++ 언어로 작성된 32비트 실행 파일을 대상으로 패킹된 파일과 원본 파일의 API 추출한다. 실행 파일은 “Hello World”를 출력하는 프로그램이다. 패킹 도구는 Aspack, UPX, Mpress, Petite 등 4가지 패킹 도구를 사용하였으며, 이들을 대상으로 API를 추출한다. 그림 3은 패킹한 실행 파일에서 추출된 API 함수 일부를 나타낸다.

UPX, Mpress, Aspack으로 패킹한 프로그램에 대해서는 그림 3과 같은 API가 호출된 후 프로그램을 실행하는 세션이 호출된다. Petite의 경우 그림 2와 같은 API 호출이 없으며 패킹된 세션을 실행하면서 원본과 같은 API를 호출하고 있다. 이것은 Petite 패커가 패킹 방식이 다른 패커와 다를것으로 판단할 수 있다.

유사도를 계산하기 위해 Smith-Waterman 알고리즘을 이용한다. OEP 이전까지의 모든 API를 대상으로 호출된 API의 함수명에 알고리즘을 적용하여 유사도를 구할 수 있다. 유사도는 해당 패커에서 추출한 모든 API를 다른 패커에서 추출한 모든 API와 비교한다. 유사도 정의는 식 (1)과 같이 정의한다.

그림 3 API 함수

$$S = \frac{2L}{M+N} \quad (1)$$

식 (1)에서 S는 유사도를 의미하며 L은 두 문자열에서 최대 길이를 가지는 부분 문자열이며 M과 N은 두 문자열의 길이이다. 위 식에 따라 유사도 값은 0에서 1사이의 값을 가지게 되며 1에 가까울수록 유사하다는 것을 의미한다. 식 (1)에서 계산된 모든 API 유사도 총합을 Smith-Waterman 알고리즘 연산 횟수에 나누어 결과를 구한다. 결과는 표 1과 같다.

표 1 유사도 결과

	Aspack	UPX	Mpress	Petite
Aspack		0.68	0.68	0.29
UPX	0.68		0.74	0.27
Mpress	0.68	0.74		0.28
Petite	0.29	0.27	0.28	

표 1의 결과를 보면 Aspack, Upx, Mpress는 유사도가 0.5 이상으로 유사도가 높게 나온 것을 확인할 수 있다. 그리고 UPX와 Mpress는 유사도 값이 0.74로 유사도 값이 가장 높다. 이를 토대로 두 패커는 매우 비슷한 패커로 분류할 수 있다.

Ⅴ. 고찰

본 논문에서는 전체 API 호출에 대해 유사도를 계산하였다. 단순한 프로그램에서 패커를 통해 패킹한 프로그램은 API 호출이 단순하고 간단하다. 복잡한 프로그램에 대해 API의 중복을 제거하고 패턴을 분석할 필요가 있다. 또한, 패커가 어떻게 패킹을 진행하는지 분석할 필요가 있디. 각 도구마다 어떻게 패킹을 하고 어떤 방법을 쓰는지에 따른 패커의 분류 방법에 대한 연구가 필요할 것으로 생각된다.

Ⅵ. 결론

본 논문에서는 API 호출을 통해 패커를 분류하는 방법을 제안한다. 먼저 패킹된 프로그램의 OEP를 수동으로 탐지하여 OEP에 대한 주소를 얻은 후, Pin Tool을 사용하여 프로그램의 API 호출을 추출한다. 추출된 API를 분석하고 서로 다른 패커의 유사도를 구하기 위해 Smith-Waterman 알고리즘을 사용하였다. 유사도를 바탕으로 유사도 값이 높을수록 해당 도구가 유사한 도구이며 패킹 방식이 비슷한 도구로 분류할 수 있다.

향후 연구로는 더 많은 패커를 사용한 실험이 필요하고 전체 API가 아닌 API의 패턴을 분석하여 유사도를 구하거나 클러스터링을 통해 패커를 분류할 필요가 있다. 또한, 추출한 API를 대상으로 패커의 특징을 연구하여 보다 정확하게 파악하는 연구가 필요하다.

ACKNOWLEDGEMENT

이 논문은 2020년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의지원을 받아 수행된 연구임 (No.2014-3-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구 (차세대 OS 기초연구센터)).
*교신 저자: 우균(부산대학교, woogyun@pusan.ac.kr).

참고문헌

[1] Hartung, Marcin. "UNPACK YOUR TROUBLE S1: .NET PACKER TRICKS AND COUNTER MEASURES."

[2] Cheng, Julia Yu-Chin, Tzung-Shian Tsai, and Chu-Sing Yang. "An information retrieval approach for malware classification based on Windows API calls." 2013 International conference on machine learning and cybernetics. Vol. 4. IEEE, 2013.

[3] Pektaş, Abdurrahman, and Tankut Acarman. "Malware classification based on API calls and behaviour analysis." IET Information Security 12.2 (2017): 107-117.

[4] Smith, Temple F., and Michael S. Waterman. "Identification of common molecular subsequences." Journal of molecular biology 147.1 (1981): 195-197

[5] Luk, Chi-Keung, et al. "Pin: building customized program analysis tools with dynamic instrumentation." Acm sigplan notices 40.6 (2005): 190-200.