논문 16-41-12-29 The Journal of Korean Institute of Communications and Information Sciences '16-12 Vol.41 No.12
https://doi.org/10.7840/kics.2016.41.12.1858

YARA 속도 개선을 위한 새로운 S/W 구조설계

김 창 훈◆

A New S/W Architecture for YARA Speed Enhancement

Chang Hoon Kim◆

요 약

논문에서는 YARA의 스캐닝 스레드 알고리즘을 개선하여 다수의 룰 파일 패턴 매칭을 수행할 수 있는 새로운 소프트웨어 구조를 제안한다. 제안하는 방식은 기존의 YARA에 비해 매칭을 위한 룰 파일의 메모리 적재 횟수를 감소시킨다. 따라서 제안된 구조를 적용할 경우 메모리 사용량은 룰 파일의 개수에 비례하여 증가하지만 패턴 매칭 수행에 따른 시간을 감소시킬 수 있다.

Key Words : YARA, Pattern Matching, Malware Detection, Signature

ABSTRACT

In this paper, a modified YARA software architecture that can perform pattern matching for multi-rule files is proposed. Based on a improved scanning thread algorithm, the new design reduces memory loading time of rule files for pattern matching. Therefore, the proposed architecture can reduce operation time for pattern matching while it requires an increased memory in proportion to the number of rule files.

Ⅰ. 서 론

YARA는 시그니처 기반 악성코드를 탐지하기 위한 오픈소스 도구이다[1,2]. YARA에서 룰 파일과 매칭에 사용되는 대상으로는 파일, 프로세스 식별자, 디렉토리가 있다. 여기서, 룰 파일은 YARA 컴파일러가 인식할 수 있는 포맷으로 작성할 수 있으며, 작성된 룰은 컴파일러에 의해 바이너리 형태의 시그니처로 변환된다. 변환된 시그니처는 패턴 매칭 수행 시 패턴 매칭 대상과의 비교를 통해 탐지를 수행한다[3].

그러나 기존의 YARA는 하나의 룰 파일만 사용할 수 있는 구조로 이루어져있어, 다수의 룰 파일을 패턴 매칭에 사용 할 경우 패턴 매칭에 사용되는 비교대상 파일을 반복적으로 메모리에 적재시켜야한다. 이는 패턴 매칭 수행 시간 증가의 주된 원인으로 작용할 수 있다.

본 논문에서는 YARA의 스캐닝 스레드 알고리즘을 개선하여 다수의 룰 파일 패턴 매칭을 수행할 수 있는 새로운 구조를 제안한다. 제안된 알고리즘은 룰 파일을 저장하는 배열을 생성하여, 다수의 룰 파일 처리 시 반복적으로 발생하는 메모리 적재 횟수를 감소시킨다. 따라서 기존 YARA에 비해 메모리 사용량이 증가하는 단점이 있다. Intel Core i7-4710HQ CPU 환경에서 600bytes 룰 파일 50개, 2.5GB 파일을 사용한 패턴 매칭 수행결과 약 1.11배의 성능 향상을 보인다.

Ⅱ. 본 론

2.1 YARA

그림 1은 YARA의 메인 함수의 패턴 매칭 수행절차를 나타낸다. 메인함수는 룰 파일과 패턴 매칭 대상

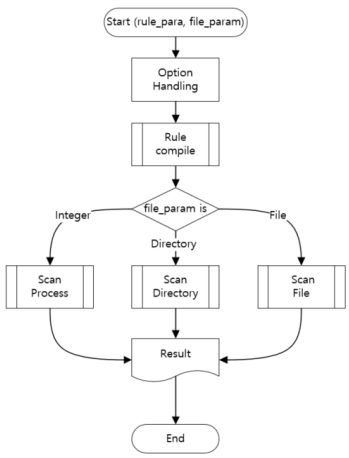

그림 1. YARA 메인 함수 알고리즘
Fig. 1. YARA main function algorithm

◆ First Author : School of Computer and Information Technology, Daegu University, kimch@daegu.ac.kr, 종신회원
논문번호 : KICS2016-11-351, Received November 12, 2016; Revised November 30, 2016; Accepted December 1, 2016

을 메모리에 적재하며, 멀티 스레드기반 스캐닝 알고리즘을 이용하여 탐지를 수행한다.

기존의 YARA는 다수의 룰 파일을 패턴 매칭에 사용 할 경우 비교대상 파일을 반복적으로 메모리에 적재 시켜야하는 문제가 발생한다. 이러한 문제점을 해결하기 위해서는 룰 파일의 메모리 적재에 소요되는 시간을 단축시킬 필요가 있으며, 이를 위해서는 다수의 룰 파일을 한번에 입력 받는 기능이 요구된다. 또한 다수의 룰 파일을 메모리에 적재시켜 패턴 매칭 대상과 비교하는 새로운 소프트웨어 구조가 필요하다.

2.2 새로운 소프트웨어 구조

본 세부 절에서는 YARA에서 다수의 룰 파일 사용 시 메모리 적재 횟수를 줄일 수 있는 새로운 소프트웨어 구조를 제안한다.

2.2.1 다중 룰 컴파일

그림 2는 다수의 룰 파일을 컴파일하고 별도로 할당한 배열에 룰 파일을 저장하는 과정을 나타낸다. 기존의 YARA는 룰 파일을 컴파일한 후 컴파일된 룰 파일을 메모리에 적재하는 과정이 싱글 스레드로 수행되지만 새롭게 제안하는 소프트웨어 구조는 룰 파일을 멀티 스레드를 사용해 동시에 컴파일 하며 각 스레드는 별도로 할당한 배열에 컴파일 된 룰 파일을 저장한다. 배열에 저장된 다수의 룰 파일을 사용하기 위해서는 룰 파일을 순차적으로 불러오는 원형 큐가 필요하다. 따라서 제안한 소프트웨어 구조에서는 룰 파일을 저장하는 별도의 배열과 룰 파일을 불러오는 원형 큐를 추가한다. 원형 큐는 룰 파일을 순차적으로 불러 스캐닝 스레드 구조에 의해 패턴 매칭이 수행될 수 있도록 룰 파일을 메모리에 적재한다.

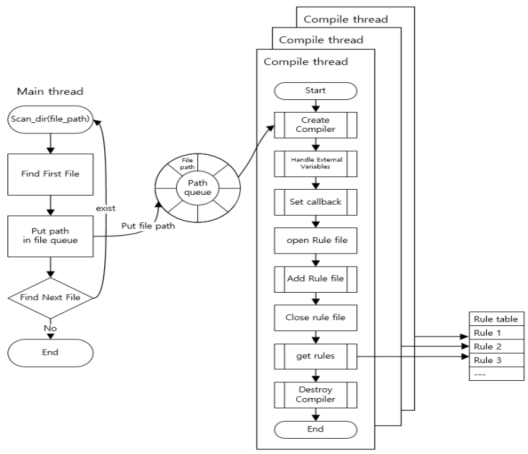

그림 2. 다중 룰 파일 컴파일과 저장 과정
Fig. 2. Procedures to multi rule file compile and store

2.2.2 개선된 스캐닝 스레드 알고리즘

그림 3은 개선된 스캐닝 스레드 알고리즘을 나타낸다. 다중 패턴 매칭 수행을 위해 멀티 스레드 구조를 사용한다. 메인함수에서 수행되는 메인 스레드는 디렉토리에 존재하는 패턴 매칭 대상의 경로를 원형 큐에 전달하는 작업을 수행한다. 패턴 매칭 대상의 경로를 전달 받은 원형 큐는 경로를 메모리에 적재한다. 따라서 메모리에는 다수의 룰 파일과 패턴 매칭 대상의 경로가 존재하게 된다. 이후 개선된 스캐닝 스레드 알고리즘에 의해 다중 패턴 매칭이 수행된다. 다수의 룰 파일을 동시에 메모리에 적재하고 비교하기 때문에 기존의 YARA에 비해 룰 파일을 메모리 적재하는 시간을 단축시킬 수 있다.

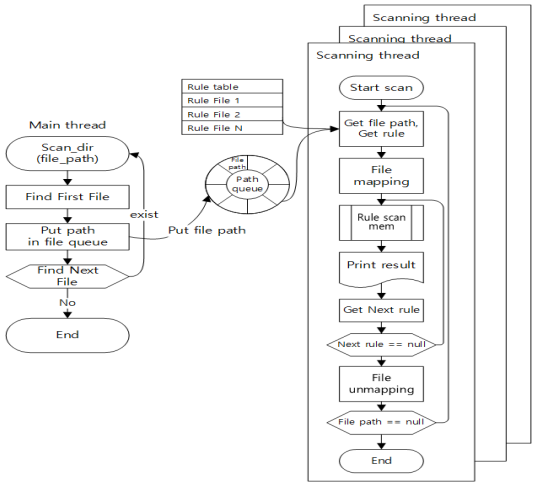

그림 3. 개선된 스캐닝 스레드 알고리즘
Fig. 3. Improved thread algorithm for scanning

2.3 성능 평가

제안한 시스템의 구조와 기존의 YARA 체계의 성능 차이를 확인하기 위해 실험 환경을 표 1과 같이 구성하여 시뮬레이션 하였다.

표 1의 내용을 바탕으로 동일한 룰 파일과 악성코

표 1. 성능 평가 환경
Table 1. Performance Evaluation Spec.

	Spec
CPU	Intel Core i7-4720HQ
Memory	16GB
OS	Windows 10 (64bit)
Malwares	500MB
Rule File	1개 (600 Byte)

드를 사용하여 성능 차이를 확인한다. 실험의 공정성을 위해서 순차적인 프로그램 실행을 수행하도록 배치 파일로 제작하여 진행한다.

그림 4는 룰 파일 10개를 기준으로 악성코드 양을 증가시킨 실험결과를 나타낸다. 악성코드 양이 많아질수록 성능 차이를 보인다. 2.5GB 악성코드를 대상으로 패턴 매칭 시 수행 시간이 약 3초 단축된다.

그림 5는 악성코드 양을 일정하게 고정시킨 후 룰 파일 개수를 증가시켜 메모리 사용량을 확인한 결과를 나타낸다. 기존에 YARA는 하나의 룰 파일을 사용하므로 적은 메모리를 사용하나 제안된 구조는 250개의 룰 파일을 사용할 경우 메모리 사용량이 약 6MB 증가한다.

그림 6은 룰 파일과 악성코드 양을 증가시켜 실험한 결과를 나타낸다. 룰 파일 50개와 악성코드 2.5GB를 사용하여 작업을 수행한 결과 제안된 구조는 수행

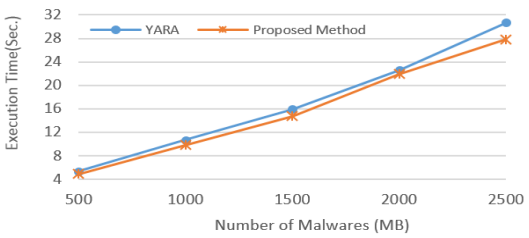

그림 4. 악성코드 양에 따른 수행시간 차이
Fig. 4. Time performance by a malware's quantity

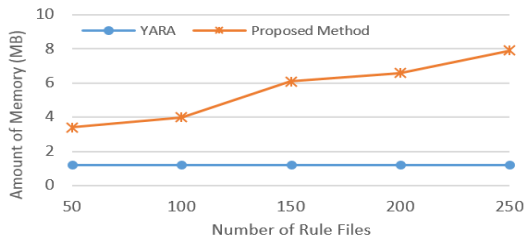

그림 5. 룰 파일 수에 따른 메모리 사용량
Fig. 5. Memory usage according to rule file numbers

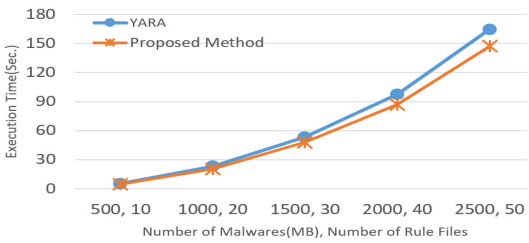

그림 6. 악성코드 양과 룰 파일 개수에 따른 수행 시간 차이
Fig. 6. Time performance by a malware's quantity and rule file number

시간이 약 17초 단축된 약 1.11배의 성능향상을 보인다.

Ⅲ. 결 론

본 논문에서는 YARA에서 다수의 룰 파일을 사용할 수 있도록 구현한 새로운 소프트웨어 구조를 제안하였다. 제안된 구조는 크게 다음과 같은 두 가지 특성을 가진다. 1) 기존의 YARA에 비해 다중 룰 파일 사용의 편리성을 가진다. 2) 메모리 적재 최소화를 통해 패턴 매칭 속도가 증가한다. 따라서 위의 두 가지 특징으로부터 본 논문에서 제안된 새로운 소프트웨어 구조는 다수의 룰을 사용한 악성코드 탐지에 적합하다 할 수 있다.

References

[1] S. K. Pandey and B. M. Mehtre, "Performance of malware detection tools: A comparison" *ICACCC 2014*, pp. 1811-1817, May 2014.

[2] I. S. Kim, J. H. Jung, H. C. Lee, and J. H. Yi, "Analysis method and response guide of mobile malwares," *J. KICS*, vol. 35, no. 4, pp. 599-609, Apr. 2010.

[3] Victor M. Alvarez, Revision (2015), Retrieved Aug., 3, 2016, from http://virustotal.github.io/yara/