서열 정렬 기법을 이용한 악성코드 유사도 분석의 성능 개선

Improvement of Performance of Malware Similarity Analysis by the Sequence Alignment Technique

저자
(Authors)

조인겸, 임을규
In Kyeom Cho, Eul Gyu Im

출처
(Source)

정보과학회 컴퓨팅의 실제 논문지 21(3), 2015.03, 263-268 (6 pages)
KIISE Transactions on Computing Practices 21(3), 2015.03, 263-268 (6 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE06204192

APA Style

조인겸, 임을규 (2015). 서열 정렬 기법을 이용한 악성코드 유사도 분석의 성능 개선. 정보과학회 컴퓨팅의 실제 논문지, 21(3), 263-268.

ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 21, No. 3, pp. 263-268, 2015. 3
http://dx.doi.org/10.5626/KTCP.2015.21.3.263

서열 정렬 기법을 이용한 악성코드 유사도 분석의 성능 개선

(Improvement of Performance of Malware Similarity Analysis by the Sequence Alignment Technique)

조 인 겸 [†]　　임 을 규 [††]
(In Kyeom Cho)　　(Eul Gyu Im)

요 약 변종 악성코드는 그 기능에 있어 차이가 없으나 구조적인 차이가 존재하는 악성코드로, 같은 그룹으로 분류하여 처리하는 것이 유용하다. 변종 악성코드 분석을 위해 본 논문에서는 바이오인포매틱스 분야에서 사용하는 서열 정렬 기법을 사용하여 악성코드들의 API 호출 정보 간의 공통부분을 찾고자 하였다. 서열 정렬 기법은 API 호출 정보의 길이에 대해 의존적인 성능을 가지며, API 호출 정보의 길이가 커짐에 따라 성능이 매우 떨어진다. 따라서 본 논문에서는 서열 정렬 기법 적용 이전에 API 호출 정보에서 발견되는 반복 패턴을 제거하는 방법을 적용함으로써 성능이 보장될 수 있도록 하였다. 최종적으로 서열 정렬 기법을 통한 악성코드 간의 유사도를 구하는 방법에 대하여 논하였다. 또한 실제 악성코드 샘플에 대한 실험 결과를 제시하였다.

키워드: 변종 악성코드, 서열 정렬, 반복 패턴 제거, 악성코드 유사도, 악성코드 분류

Abstract Malware variations could be defined as malicious executable files that have similar functions but different structures. In order to classify the variations, this paper analyzed sequence alignment, the method used in Bioinformatics. This method found common parts of the Malwares' API call information. This method's performance is dependent on the API call information's length; if the length is too long, the performance should be very poor. Therefore we removed the repeated patterns in API call information in order to improve the performance of sequence alignment analysis, before the method was applied. Finally the similarity between malware was analyzed using sequence alignment. The experimental results with the real malware samples were presented.

Keywords: malware variations, sequence alignment, repeated patterns removal, malware similarity, malware classification

· 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대 정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. 한국연구재단에서 부여한 과제번호 2011-0029923)
· 이 논문은 2014 한국컴퓨터종합학술대회에서 '서열 정렬 알고리즘을 적용을 통한 악성 코드 유사도 분석 및 분류'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 한양대학교 컴퓨터 소프트웨어 전공
dlsrua1004@hanyang.ac.kr
†† 종신회원 : 한양대학교 컴퓨터공학부 교수(Hanyang Univ.)
imeg@hanyang.ac.kr
(Corresponding author임)

논문접수 : 2014년 9월 5일
(Received 5 September 2014)
논문수정 : 2014년 12월 1일
(Revised 1 December 2014)
심사완료 : 2014년 12월 22일
(Accepted 22 December 2014)

정보과학회 컴퓨팅의 실제 논문지 제21권 제3호(2015. 3)

1. 서 론

시간이 지남에 따라 악성코드는 단순히 그 수가 증가하는데 그치지 않고, 다양한 변종이 존재하게 되었다. 변종 악성코드는 기본적인 기능에 있어 차이가 거의 없으나, 구조적인 차이가 존재하는 악성코드들을 가리킨다.

이러한 변종 악성코드는 행위가 유사한 변종끼리 분류하여 처리하는 것이 일반적이다. 변종 악성코드 분석에는 악성코드의 행위 분석이 필요하며, 동적 분석 기법이 주로 사용된다[1]. 그러나 단순히 행위 분석 기법을 통해서 얻어지는 정보만으로는 변종 악성코드를 분류하는데 있어 충분하지 않으며, 추가적인 과정이 필요하다. 따라서 본 논문에서는 바이오인포매틱스(Bioinformatics) 분야에서 사용되는 '서열 정렬(Sequence Alignment)' 기법을 이용하여 변종 악성코드 간의 공통부분을 찾고, 이를 이용하여 악성코드 간 유사도를 구하는 기법을 제시하고자 한다.

서열 정렬 기법은 악성코드의 행위 정보를 나타내는 API 호출 리스트를 분석 대상으로 한다. 서열 정렬 기법의 성능은 API 호출 리스트의 길이에 따라 좌우된다. 따라서 서열 정렬 기법을 이용한 유사도 분석의 신뢰성과 성능이 최적이 되도록 API 호출 리스트를 가공하여야 한다. 이를 위해 본 논문에서는 API 호출 리스트 내의 반복되는 패턴을 제거하는 방법을 이용하였다.

2장에서는 본 논문에서 수행한 연구와 관련된 연구들에 대하여 간단히 소개한다. 이어서 본 연구에서 이용하고자 하는 서열 정렬 기법에 대하여 3장에서 설명하며, 서열 정렬 기법의 성능 개선을 위한 API 호출 리스트의 반복 패턴 제거에 관하여 4장에서 설명한다. 이어지는 5장에서 본 연구에서 제시하는 악성코드 유사도 분석 과정을, 그리고 6장에서 제시한 방법의 실제 실험 결과를 보였다. 최종적으로 7장에서 결론을 제시하였다.

2. 관련 연구

행위 분석을 통한 악성코드 분석에 관련된 연구는 다음과 같다. 기본적으로 각 관련 연구들은 아래와 같이 기본적인 과정을 통해 악성코드를 분석하였다. 우선 행위 분석을 통하여 악성코드가 호출한 API의 목록을 얻는다. 이어서 API를 이용하여 악성코드를 분류 혹은 탐지하는데 쓸 수 있는 척도를 얻는다. 이렇게 만들어진 척도를 통해 데이터마이닝 등의 기법을 적용, 최종적인 악성코드 분석 결과를 도출하였다.

논문에 따라 API를 어떻게 이용하는지는 조금씩 차이가 있었다. 예를 들어 API 호출 리스트를 통해 frequent itemset을 만들어 사용하거나[2], API를 통해 악성코드의 '행위'를 정의한 뒤 분류(classfication)를 수행하는 방법[3] 등이 연구되었다. 또한 API의 특성에 따라 분류를 수행하거나[4] 그래프를 정의하여 변종 악성코드를 탐지하는 연구[5]도 수행되었다.

추가적으로 정적으로 악성코드를 분류하는 연구가 있었다. 악성코드 바이너리 파일의 내부 정보를 이용하여 분류하는 연구[6], 악성코드가 사용하는 명령어의 빈도수를 이용하여 분류하는 연구[7-9]가 있었다. 또한 블룸 필터를 이용한 악성코드 분석 연구[10] 역시 수행되었다.

3. 서열 정렬(Sequence Alignment)

서열 정렬 기법은 바이오인포매틱스 분야에서 쓰이는 기법이다. 바이오인포매틱스 분야에서는 DNA, RNA, 단백질 등의 서열을 분석 대상으로 한다. 이러한 서열들을 분석하여, 두 개 혹은 그 이상의 서열들의 유사한 부분을 찾아내고 그 관계성을 찾아내는 것이 바이오인포매틱스 분야의 큰 목표 중 하나이다. 이 때 사용되는 기법 중 대표적인 것이 서열 정렬 기법이다.

DNA의 경우 그 서열은 A, G, C, T의 4가지 문자로 이루어진 문자열로 볼 수 있다. 이 단위가 되는 각 문자를 '토큰'이라 정의한다. 두 문자열에 포함된 각 토큰에 대하여 하나씩 비교하여 일치하는 부분이 가장 많아지도록 문자열을 정렬하는 것이 서열 정렬 기법에 쓰이는 알고리즘의 목표이다.

서열 정렬 알고리즘은 다음과 같은 과정을 수행한다. 두 문자열 a, b에 대하여 각 길이를 m, n으로 정의한다. 이 때 아래 수식과 같이 행렬 H를 정의할 수 있다. 행렬 H의 각 원소는 아래의 4가지 경우 중 최대의 값을 가지며, 각 경우의 값은 '점수'라고 정의한다.

$$H(i,0)=0,\ 0\le i\le m$$
$$H(0,j)=0,\ 0\le j\le n$$
$$H(i,j)=\max\begin{cases}0\\ H(i-1,j-1)+s(a_i,b_j)\\ \max_{k\ge 1}\{H(i-k,j)+W_k\}\\ \max_{l\ge 1}\{H(i-k,j)+W_k\}\end{cases}$$
$$(1\le i\le m,\ 1\le j\le n)$$

a, b는 각각 "ACACACTA", "AGCACACA"라고 가정하며, $s(a_i, b_j)$는 a_i, b_j가 같을 때 2, 다를 때 -1이다. Wi는 정렬을 수행하는데 있어 일부러 공백(gap)을 삽입했을 때 더욱 높은 유사성이 얻어지는 경우가 있는데, 이렇게 공백을 삽입했을 경우에 적용되는 값이다. -i로 정의된다. 최종적으로 행렬 H는 아래와 같이 형성된다.

$$H=\begin{bmatrix} & -A & C & A & C & A & C & T & A\\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2\\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1\\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1\\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4\\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 8\\ A & 0 & 2 & 3 & 6 & 6 & A & 8 & 7 & A\\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9\\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12\end{bmatrix}$$

행렬 H의 최댓값을 갖는 원소를 기준점으로 하여, 대각선과 가로, 세로 방향 중 한 방향을 선택하여 진행한다. 만약 값이 같을 경우, 대각선 방향이 우선시 되며 가로, 세로 방향은 동일한 순위를 갖는다. 최종적으로 값이 0인 원소에 도달할 때까지 진행하며 경로를 형성한다.

경로가 형성되면 형성할 때의 반대 순서로 진행하며 실제 정렬을 수행한다. 위 행렬에서 가로 방향으로 진행하는 경우 문자열 b에 공백이 삽입되며, 세로 방향의 경우는 a에 공백이 삽입된다. 대각선의 경우는 공백이 삽입되지 않는다. 이렇게 수행된 정렬 결과는 아래와 같다.

a: **A-CACACTA**
b: **AGCACAC-A**

서열 정렬 알고리즘은 크게 두 가지 종류로 나뉜다. 먼저 두 문자열의 공통부분을 찾는데 그 의의를 두는 '국부 정렬(Local Alignment)' 알고리즘이 있다. 반대로 전체적인 관점에서 두 문자열이 가장 유사하도록 하는 '포괄 정렬(Global Alignment)' 알고리즘이 있다. 전자의 경우 두 문자열의 길이가 다르거나, 대체적으로 다른 문자열을 정렬하는데 쓰인다. 대표적인 알고리즘으로 Smith-Warterman 알고리즘이 있다[11]. 후자의 경우는 두 문자열이 길이가 비슷하고 전체적으로 비슷한 경우에 쓰인다. Needlman-Wunsch 알고리즘이 대표적이다[12]. 두 알고리즘은 정렬 과정은 동일하나, 점수 적용에서 차이가 있다.

본 연구에서는 변종 악성코드들의 API 호출 리스트의 길이가 대체로 다양하며, 호출되는 API의 종류 역시 다양하다. 따라서 전체적인 유사도보다는 부분적인 유사도를 얻는 것이 좋을 것으로 판단하였으며, 따라서 본 논문에서는 Smith-Waterman 알고리즘을 이용하였다.

4. 서열 정렬 기법의 성능 개선

4.1 서열 정렬 기법의 성능 문제

서열 정렬 기법의 성능은 기본적으로 분석 대상으로 하는 서열의 길이에 의존적이다. 3장에서 설명한 행렬을 구성하는 경우의 시간 복잡도는 m*n이다[13]. 이 때 m, n은 각각 분석 대상이 되는 두 서열의 길이를 나타낸다. 만약 두 서열 중 한 서열의 길이가 상당히 길어지는 경우(약 2만개 이상의 문자로 이루어진 문자열 등), 행렬 자체의 크기가 매우 커질 뿐 아니라 행렬을 구성하는데 필요한 시간이 지나치게 길어진다. 서론에서 언급한 바와 같이 악성코드는 그 수적으로 큰 증가량을 가지므로, 성능 문제는 정확도만큼이나 중요한 고려사항이다.

동일한 길이를 같는 두 서열의 길이에 따른 서열 정렬 수행 시간은 아래 그림 1과 같다. 그래프의 경향으로 확인할 수 있듯이, 서열 길이가 증가함에 따라 시간 복잡도는 지수적으로 증가한다. 이는 앞서 언급한 바와 같이 성능 문제에 대한 고려가 반드시 필요하다는 것을 보여준다.

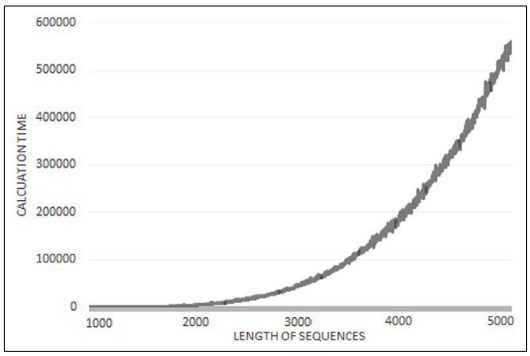

그림 1 서열 길이에 따른 서열 정렬 시간
Fig. 1 Running time of sequence alignment for the length of sequences

이러한 행렬 구성의 시간 복잡도 문제는 곧 행렬 구성에 걸리는 시간 비용을 줄이는 것으로 해결할 수 있다. 널리 쓰이는 서열 정렬 알고리즘이자 프로그램인 BLAST의 경우, 유사한 두 서열을 정렬할 때는 행렬의 주대각선 근처 부분만을 구성하게 된다는 점을 이용하여 행렬 구성에 필요한 시간을 단축하였다[14]. 그러나 본 논문에서 분석하고자 하는 악성코드의 API 호출 리스트의 경우 3절에서 언급한 바와 같이 전체적인 유사도가 낮으므로 앞서 설명한 BLAST와 같은 방법을 적용할 수 없다. 따라서 다른 방법을 이용하여 행렬 구성에 걸리는 시간을 단축하여야 한다.

4.2 반복 패턴 제거를 통한 성능 개선

본 논문에서는 행렬 구성에 걸리는 시간 단축을 위해 행렬의 크기 자체를 축소하는 방법을 시도하였다. 이를 위해 행렬 크기와 대응되는 분석 대상 서열들의 길이를 단축하고자 하였다. 이 때 본래 서열이 가지던 정보가 손실될 수 있으나, 서열의 길이를 줄이더라도 분석 결과로 얻어지는 두 서열 간 유사도의 정확성은 최대한 낮아지지 않아야 한다. 이러한 서열 길이 단축을 위하여 본 논문에서는 반복 패턴 제거를 적용하였다.

본 논문에서는 하나의 서열 안에서 2번 이상 연속되어 나타나는 부분을 가리켜 '반복 패턴'이라고 정의하였다. 예를 들어 서열 "ACACACTA"의 경우, "AC"가 3번 연속으로 반복되어 나타난다. 이러한 반복 패턴을 제거하면 서열 "ACTA"를 얻는다.

이와 같이 반복 패턴 제거를 통한 접근은 두 서열 간의 유사한 부분 중 반복되는 부분을 제거하여도 유사도 측면에서는 큰 손실이 없다는 것에서 비롯되었다. 또한 프로그램이 반복문을 수행하는 경우가 빈번하기 때문에

API 호출 리스트 역시 반복되는 부분이 많으며, 이러한 반복 패턴을 제거함으로써 매우 높은 성능 향상을 기대할 수 있다.

5. 제시하는 악성코드 유사도 분석 과정

본 논문에서 제시하는 악성코드 유사도 분석 기법은 그림 2와 같이 도식화 할 수 있다.

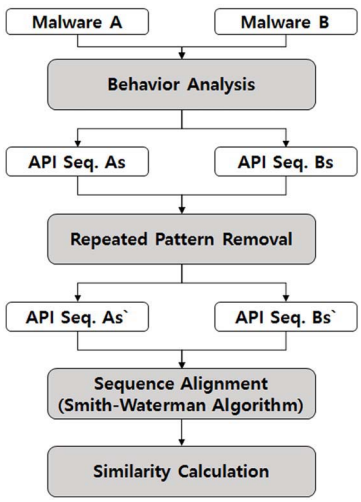

그림 2 제시하는 악성코드 유사도 분석 과정

Fig. 2 The process of the proposed malware similarity analysis

5.1 악성코드 행위 분석

악성코드 간 유사도 분석을 위해 3절에서 언급한 바와 같이 Smith-Waterman 알고리즘을 이용하고자 하였다. 알고리즘을 적용하는 대상은 악성코드의 API 호출 리스트이다. API 호출 리스트를 얻기 위해서는 우선 악성코드의 행위 분석 결과가 필요하였으며, 행위 분석은 오픈소스 샌드박스인 Cuckoo Sandbox[15]를 이용하였다.

Cuckoo Sandbox는 악성코드에 대해 기본적인 시그니처 분석과 더불어 행위 분석까지 수행한 뒤 그 결과를 보고서로 정리하여 출력한다. 본 연구에서는 악성코드의 API 호출 리스트가 필요한 것이므로, 먼저 보고서에서 API 호출에 관련된 부분, 즉 행위 분석 결과 부분만 따로 추출한다. 이렇게 일차적으로 추출한 API 호출 리스트는 호출된 시간 순서대로 API의 이름을 나열한 것이다.

Smith-Waterman 알고리즘을 이용하기 위해서는 단위체로 이루어진 문자열이 필요하다. 따라서 API 호출 리스트에 포함된 각 API 이름에 대하여 정의된 코드의 리스트가 필요하였다. API 코드는 다음과 같이 정의하였다. 첫째, 코드는 크게 2부분으로 나뉜다. API가 속하는 카테고리 부분과 해당 카테고리 내에서 API 이름 부분이다. 둘째, 각 부분은 카테고리 부분이 4비트, API 이름 부분이 8비트의 길이를 갖으며, 16진수로 변환하여 각 1, 2개의 문자(0~F)로 표현된다. 예를 들어, 3번째 카테고리에 속하는 5번째 API는 305이다. 표 1에서 본 연구를 위해 정의한 카테고리 목록과 각 카테고리에 해당하는 코드를 나타내었다.

표 1 카테고리 별 코드

Table 1 The codes for the categories

category	code	category	code
registry	0	system	7
filesystem	1	device	8
process	2	threading	9
service	3	hooking	A
network	4	misc	B
socket	5	windows	C
synchronization	6	-	-

5.2 반복 패턴 제거

4절에서 설명한 반복 패턴 제거를 수행한다. 반복 패턴 제거에 있어 고려할 사항 중 하나는 제거 대상이 되는 반복 패턴의 길이이다. 즉 얼마만큼의 길이를 기준으로 반복 여부를 탐색할지를 결정해야 하는 것이다.

그림 3은 반복 패턴의 길이에 대한 서열 길이의 변화를 나타낸 것이다. 이 그래프를 통해 5개 단위로 이루어진 반복 패턴을 탐색하여 제거하는 경우 서열 길이가 크게 단축되는 것을 확인할 수 있다. 또한 7개 단위의 반복 패턴을 제거하는 경우 모든 서열은 최소의 길이를 가진다. 따라서 반복 패턴의 길이는 7개 단위로 정의하였다.

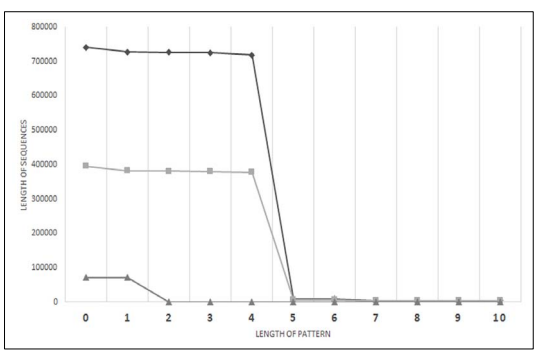

그림 3 반복 패턴 길이에 따른 서열 길이 변화

Fig. 3 change of sequences` lengths for repeated pattern`s length

5.3 서열 정렬 기법 적용 및 유사도 산출

API 코드 리스트에 대한 반복 패턴 제거 이후 최종

적으로 Smith-Waterman 알고리즘을 적용하여 유사도 분석을 수행한다. 알고리즘을 통해 얻어지는 것은 일치하는 부분들에 대한 정보이다. 본 연구에서는 유사도를 다음과 같이 정의하였다.

$$similarlity = \frac{L}{(m+n)/2}$$

위 수식에서 L은 최대 길이를 갖는 부분 문자열의 길이이며, m, n은 각각 두 문자열의 길이를 의미한다. 위와 같이 정의된 유사도는 0과 1 사이의 값을 갖게 되며, 1에 가까울수록 두 악성코드의 유사한 부분이 많은 것이다.

6. 실 험

앞서 제시한 방법을 이용하여 실제로 10가지 패밀리에 속하는 변종 악성코드에 대하여 실험을 수행하였다. 각 패밀리마다 15개의 샘플을 선정하여 총 150개의 샘플에 대해 실험을 수행하였다. 실험 결과는 그림 4와 같이 막대그래프를 통해 시각화하였다.

그림 4의 그래프에서 가로축은 각 패밀리의 이름을 나타내며, 세로축은 각 패밀리별로 동일한 패밀리와 다른 패밀리에 대해 얻어진 유사도 값의 평균이다. 각 패밀리에 대해 왼쪽 막대는 동일 패밀리에 대한 유사도, 오른쪽 막대는 다른 패밀리들에 대한 평균 유사도이다.

같은 패밀리에 속하더라도 API 호출 순서가 일치하는 부분이 많지 않아 유사도 자체는 낮게 얻어지기도 하였다. 그러나 그래프를 통해 확인할 수 있듯이, 특정 패밀리에 속하는 악성코드 샘플 간의 유사도는 같은 패밀리에 속하는 악성코드 샘플들에 대하여 상대적으로 높게 나타났다. 유사도 분석 결과의 경향성을 고려할 때 본 연구에서 제시하는 유사도 분석 방법을 변종 악성코드의 분류에 적용할 수 있을 것이다.

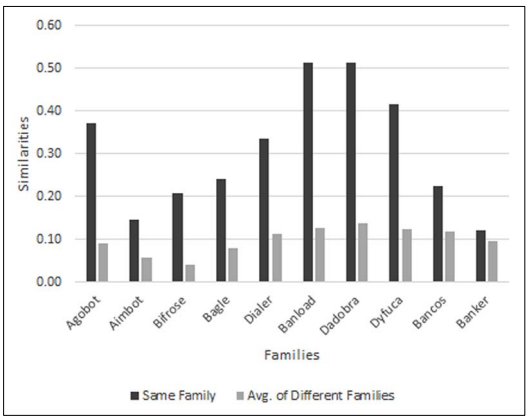

그림 4 유사도 분석 실험 결과

Fig. 4 Result of the similarity analysis experiments

7. 결 론

본 논문에서는 악성코드의 행위 분석을 통해 API 호출 정보를 추출하였고, 추출한 API 호출 정보에 대해 서열 정렬 알고리즘을 적용하여 악성코드 간 유사도를 계산하였다. 또한 알고리즘 적용 이전에 API 호출 정보 내의 반복 패턴을 제거함으로써 알고리즘의 성능 향상 가능성을 제시하였다. 또한 제안하는 유사도 분석 방법의 실제 분석 결과를 실험 결과로서 제시하였다.

본 논문에서 설명한 연구의 한계점과 그에 대한 향후 연구 방향은 다음과 같다. 우선 악성코드가 아닌 정상 프로그램에 대한 추가적 실험이 필요하다. 이는 단순 악성코드 분류가 아닌 악성코드 탐지 척도로서의 이용 가능성을 확인하기 위함이다. 또한 반복 패턴 제거 뿐 아닌 알고리즘 자체의 성능 개선 방법에 대한 지속적인 연구를 시도할 계획이다.

References

[1] M. Egele, T Scholte, E. Kirda, and C. Kruegel, "A Survey on Automated Dynamic Malware-Analysis Techniques and Tools," *ACM Computing Surveys (CSUR)*, Vol. 44, No. 2, pp. 1-42, Feb. 2012.

[2] Y. Qiao, Y. Yang, and L. Ji, J. He, "Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences," *2013 12th IEEE International Conference on. IEEE*, pp. 265-270, 2013.

[3] D. Veerwal and P. Menaria, "Ensemble of Soft Computing Techniques for Malware detection," *International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)*, Vol. 6, No. 2, pp. 159-167, Sep.-Nov. 2013.

[4] K.S. Han, I. K. Kim, and E. G. Im, "Malware Classification Methods Using API Sequence Characteristics," *Proc. of the International Conference on IT Convergence and Security 2011 Lecture Notes in Electrical Engineering*, Vol. 120, pp. 613-626, 2012.

[5] K. S. Han, I. K. Kim, and E. G. Im, "Detection Methods for Malware Variant Using API Call Related Graphs," *Proc. of the International Conference on IT Convergence and Security 2011 Lecture Notes in Electrical Engineering*, Vol. 120, pp. 607-611, 2012.

[6] B. Kang, T. Kim, H. Kwon, Y. Choi, and E. G. Im, "Malware Classification Method via Binary Content Comparison," *Proc. RACS '12 Proceedings of the 2012 ACM Research in Applied Computation Symposium*, pp. 316-321, 2012.

[7] K. S. Han, B. Kang, and E. G. Im, "Malware Classification using Instruction Frequencies," *Proc. RACS '11 Proceedings of the 2011 ACM Symposium on Research in Applied Computation*, pp. 298-

300, 2011.
[8] K. S. Han, B. Kang, H. Kwon, B. Li, and E. G. Im, "Virus Classification via Instruction Frequency," *Proc. of the Fourth Joint Workshop between HYU and BUPT*, Oct. 2011.
[9] K. S. Han, S. Kim, and E. G. Im, "Instruction Frequency-based Malware Classification Method," *International Information Institute(Tokyo) Information*, Vol. 15, No. 7, 2012.
[10] B. Kang, H. S. Kim, T. Kim, H. Kwon, and E. G. Im, "Fast Malware Classification using Counting Bloom Filter," *INFORMATION-An International Interdisciplinary Journal*, Vol. 15, No. 7, Jul. 2012.
[11] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, Vol. 147, No. 1, pp. 195-197, Mar. 1981.
[12] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, Vol. 48, No. 3, pp. 443-453, Mar. 1970.
[13] J, Cohen, "Bioinformatics-an introduction for computer scientists," *ACM Computing Surveys (CSUR)*, Vol. 36, No. 2, pp. 122-158, Jun. 2004.
[14] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, Vol. 215, No. 3, pp. 403-410, Oct. 1990.
[15] Cuckoo Sandbox. [Online] Available: http://cuckoosandbox.org/

조 인 겸
2014년 한양대학교 컴퓨터공학부 졸업(학사). 2014년~현재 한양대학교 컴퓨터소프트웨어 전공 석사과정. 관심분야는 컴퓨터보안, 악성코드 분석

임 을 규
1992년 서울대학교 컴퓨터공학과 졸업(학사). 1994년 서울대학교 컴퓨터공학과 졸업(석사). 2002년 University of Southern California Computer Science Dept. 졸업(박사). 2005년~현재 한양대학교 컴퓨터공학부 부교수. 관심분야는 제어시스템 보안, 악성코드, 정보보호, 소프트웨어 취약 점검