Multi N-gram을 이용한 악성코드 분류 시스템

권희준[1], 김선우[2], 임을규[3]

An Malware Classification System using Multi N-gram

Heejun Kwon[1], Sunwoo Kim[2], Eul Gyu Im[3]

요 약

악성코드(Malware)란, 일반 사용자의 컴퓨터를 감염시켜 악성 행위를 하기위한 목적으로 만들어진 바이너리 파일을 일컫는다. 아이디나 암호와 같은 개인정보를 유출하는 것에서부터 주요기관에 대한 DDoS공격 까지, 다양한 종류의 악성 행위를 하며, 보통 이들의 행위를 바탕으로 트로이 목마, 바이러스, 웜, 디도스 등으로 구분된다. 이러한 악성코드의 탐지는 여러 가지 많은 분석방법을 통해 만들어진 시그니처에 의해 주로 수행된다. 하지만 하루에도 수 백개의 변종 악성코드들이 간단한 실행압축 등의 기술을 통해서 제작되며 이를 시그니처를 기반으로 탐지하는 경우, 변종 분석 및 데이터베이스의 업데이트 등 오랜 소요 시간을 필요로 한다. 본 논문에서는 많은 탐지방법 중 하나인 N-gram 기법을 설명하고, 이를 악성코드 분류에 적합하게 개선하여 동종 및 변종 악성코드를 분류하는 방법을 제시하고 일부 악성코드들을 대상으로 제시한 기법을 적용함으로써 그 정확성과 효율성을 기술한다.

핵심어 : 악성코드 분류, 바이너리 분석

Abstract

Malware means binary file infecting user computers which aim to does malicious actions. Not only exposing personal informations such as ID and password, but also attacking major organization using DDoS, they do actions in various ways. In common, they are classified by their behavior, for example, Trojan, virus, worn and DDoS. Detection of malwares are usually conducted by signatures obtained by various analysis methods. However, because hundreds of mutant malwares are produced in a simple packing technique in a day, existing detection technique costs long processing time, for analyzing mutants and updating databases. In this paper, we explain N-gram technique and improve it to fitted for classifying malwares whether same kind or various ones. Furthermore, verifies accuracy and efficiency of this novel technique by applying it to the real-world malwares.

Keywords : Malware classification, Binary analysis

접수일(2012년10월16일), 심사의뢰일(2012년10월17일), 심사완료일(1차:2012년10월30일, 2차:2012년11월15일)

게재일(2012년12월31일)

[1]133-791 서울시 성동구 행당1동 17 한양대학교 전자컴퓨터통신공학과.
email : heejuni@hanyang.ac.kr

[2]133-791 서울시 성동구 행당1동 17 한양대학교 전자컴퓨터통신공학과.
email : bitjaru@hanyang.ac.kr

[3](교신저자)133-791 서울시 성동구 행당1동 17 한양대학교 컴퓨터공학부.
email: imeg@hanyang.ac.kr

* 본 연구는 문화체육관광부 및 한국저작권위원회의 2012년도 저작권기술개발사업의 연구결과로 수행되었음.

1. 서론

최근 소프트웨어 시장의 규모가 기하급수적으로 증가하게 되면서, 하루에도 수십만 건의 바이너리(Binary)가 제작된다. 하지만 단순히 바이너리 뿐만 아니라, 악성 행위를 수행하는 바이너리를 의미하는 악성코드의 수 또한 폭발적으로 증가하고 있다. 불법적인 방법을 통해 사용자들의 개인정보를 탈취하는 목적에서부터 공격자의 명령에 따라 다른 컴퓨터를 공격하기 위한 목적, 그리고 금전적인 목적까지 다양한 형태로 작성된 악성코드들은 하루에도 수 백 또는 수 천개 이상이 발견되고 있다. 이러한 악성코드들은 자동 생성 및 변경 도구들을 통해 다양한 변종들을 손쉽게 생성시킬 수 있기 때문에, 신속한 분류를 통한 대응이 필요하다. 하지만 악성코드 탐지에서부터 시그니처(Signature)의 생성까지는 많은 시간이 소요된다. 실제로 분석하는 시간이 악성코드의 악성행위 발발 시점보다 오래 소요되어 제 시간 안에 악성행위를 방지할 수 없는 경우 또한 존재한다.

현재 대부분의 백신 프로그램 내에서 악성코드 탐지 및 치료는 동적(dynamic) 혹은 정적(static) 분석방법을 통해서 얻어진 시그니처를 사용하여 동작한다. 동적 분석의 경우, 제한된 시간 및 독립된 공간 내에서 바이너리를 직접 실행시키고, 실질적으로 일어나는 행위들을 관찰하는 방법이다. 하지만, 악성행위가 제한된 시간동안 발생하지 않는다면 이를 검출해 낼 수 없으며, 즉 바이너리의 모든 행위를 분석할 수 없다는 단점이 있다. 정적 분석의 경우, 바이너리를 어셈블리어 레벨에서 관찰한 후, 그 특징을 기반으로 시그니처를 생성하는 분석방법이다. 모든 발생가능한 행위를 검출할 수 있다는 장점이 있지만, 이러한 분석을 위한 선처리(pre-processing)과정이 필요하며, 분석 시간이 오래 걸린다는 단점이 있다.

본 논문에서는 선처리 과정을 필요로 하지 않는 정적 분석방법인 N-gram 기법을 통한 시그니처 생성 및 탐지에 초점을 맞추었다. 기존의 기법의 한계점을 제시하고, N-gram 적용 대상 범위 text 섹션으로 제한하여 그 정확성과 신속성을 높인다. 대부분의 악성코드들이 안티 바이러스를 회피하기 위해 실행 압축(Packing)을 수행하며, 섹션 구분이 어려워지게 된다. 이를 해결하기 위해 섹션 별 엔트로피 값을 이용한다. 또한, 멀티 N-gram 매트릭스(Multi N-gram Matrix) 기법을 적용하여 한번의 N-gram 수행에 1-gram, 2-gram, 3-gram을 동시에 수행함으로써 효율성을 높이는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 N-gram 기법들과, 제안하는 . text 섹션 추출에 사용되는 엔트로피에 대해 설명한다. 3장에서는 본 논문에서 제안하는 프레임워크의 전체적인 구조를 설명한다. 4장에서는 제안한 기법을 바탕으로 실제 악성코드 및 정상 프로그램을 분류해내는 실험을 수행함으로써 그 우수성을 증명한다. 마지막으로 5장에서는 결론과 향후 연구 방향에 대하여 제시한다.

2. 관련 연구

2.1 N-gram

N-gram이란, 바이너리 파일 전체를 길이 N의 서브스트링(Sub-string)으로 나누는 방법을 의미한다. N의 단위는 1 bit, 1 hex, 1 byte 등 기준에 따라 자유롭게 정할 수 있다. 이 기법은 대표적인 확률적 언어 모델의 하나로써, n개의 단어의 연쇄를 확률적으로 표현하여 다음 단어를 확률적으로 예측하는 데 사용된다. 예를 들어 "SIGNATURE"라는 문자열이 있다고 가정했을 경우, 1byte를 기준으로 5-gram 기법을 적용한다면, "SIGNA", "IGNAT", "GNATU", "NATUR", "ATURE" 라는 5가지 하위 문자열들이 각각 빈도수 1로 생성된다. N-gram을 이용한 악성코드 탐지기법은 이러한 하위 문자열의 빈도수를 프로그램의 시그니처로 사용하여 탐지하는 방법이다. 프로그램 전체를 이진 바이너리 파일로 표현한 후, n에 따라 크기 n의 하위 바이너리 문자열을 추출한 것이 시그니처로 사용된다.

[1, 2]에서는 KNN-알고리즘을 사용하여 N-gram 시그니처를 비교하여 악성코드를 탐지하는 방법을 연구하였다. 추출된 N-gram 중 빈도수 상위 K개의 N-gram만을 유사도 계산에 사용하였으며, 탐지를 위해 악성코드들과 정상 프로그램들의 N-gram을 추출하여 시그니처로 제작하였다. 알 수 없는 파일이 등장하면 악성 행위 여부를 판단하기 위해 두 시그니처와 유사도를 계산한다. 실험 결과를 통해 N = 4, K = 17일 경우 가장 적은 오탐율을 보인다고 밝혔다.

[3]에서는 한번의 N-gram 추출에 2에서 6 사이의 크기의 2-gram 윈도우(Window)를 사용함으로써 N-gram의 효율을 늘렸다. 추출된 처음과 마지막 2-gram의 값을 사용하여 다음 추출할 윈도우의 크기를 유동적으로 변경하는 방식을 통해 불필요한 부분으로 생각되면 추출 자체를 하지 않는 방법이다. 추출된 2-gram을 레퍼런스 벡터(Reference vector)로 표현하고, 이를 클러스터링(Clustering)함으로써 악성 여부를 판단한다.

[4]에서는 1971개의 정상 프로그램과 1651개의 악성코드를 대상으로 N-gram을 추출하여 인포메이션 게인(Information Gain)을 측정하여 신뢰도가 높은 N-gram을 시그니처로 사용하였다. 인포메이션 게인이란 해당 N-gram의 출현 확률을 기반으로 해당 N-gram이 가지고 있는 정보의 가치를 측정하는 기법이다.

N-gram을 단순히 바이트 레벨에만 적용하는 것이 아니라, 디스어셈블링을 통해 명령어를 추출한 다음 opcode 레벨의 N-gram을 적용하는 연구[5] 또한 존재한다. 다양한 악성코드 및 정상 프로그램들의 opcode N-gram을 추출하여 머신 러닝(Machine learning)을 통해 악성 여부를 판단한다.

[5]는 언어 회화 표현을 빠르게 검사하기 위해 N-gram을 트리형태로 구현하여 적용하였고, 이를 통해 N-gram의 확장된 연산 처리를 가능하게 함으로써, 단어 N-gram들의 연계성을 표현하는 데

사용하였다. 본 논문에서 제안하는 Multi N-gram Matrix방법은 [5]에서 제시한 기법을 배열수준으로 구현하여 시그니처 제작에 응용함으로써 보다 효율적인 악성코드 분류를 수행한다.

2.2 Entropy

엔트로피란 정보의 양을 나타내는 척도로 사용되며, Shannon's Entropy라고도 불려진다. 이를 수식으로 표현하면 다음과 같다.

$$H(x) = -\sum_{i=1}^{n} P(i) \times \log_2 P(i)$$

$P(i)$는 i가 일어날 확률을 의미하며, 엔트로피 값 $H(x)$는 $P(i)$와 $\log_2 P(i)$의 곱의 합이다. 엔트로피 값이 높다는 것은 해당 정보가 나타나는 확률이 낮다는 것이며, 그만큼 정보의 가치가 높다는 것을 의미한다.

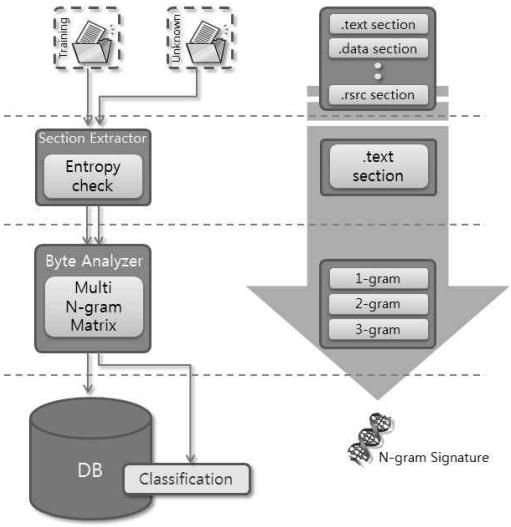
[7, 8]에서는 파일의 엔트로피를 분석하여 패킹(Packing)이 되었는지 되지 않았는지를 탐지할 수 있음을 밝혔다. 이는 정상적인 파일의 경우 섹션이 나누어져 있기 때문에 데이터가 부분적으로 집중되어있음을 이용한 것이다. 패킹을 한 경우 일반적으로 데이터가 균등하게 파일 전체에 고루 분배되기 때문에 엔트로피가 높게 나타난다. 이러한 엔트로피의 차이를 통해 분류할 수 있다.

[9]에서는 암호화 된 프로그램 또한 엔트로피 분석을 통해 구분할 수 있음을 제시하였다. 일반적인 프로그램의 경우 6.08 – 6.36의 엔트로피 값을 가지며, 패킹이 된 프로그램의 경우 7.19 - 7.26의 높은 값을 가졌다. 마지막으로 암호화 된 프로그램의 경우 7.2 – 7.3의 값을 가졌다.

3. 제안하는 방법

3.1 개요

본 논문에서는 N-gram을 사용하여 악성코드를 분석한다. 제안하는 시스템의 구조는 [그림 1]과 같다. 제안하는 시스템은 악성코드의 효율적이고 정확한 악성코드 분석을 위해 N-gram 적용 범주를 .text 섹션에 집중한다. 섹션들을 분류하기 위해 본 논문에서는 섹션 별 엔트로피 값을 계산하여 비교한다. 추출된 .text섹션에는 Multi N-gram matrix기법을 사용하며, 추출한 N-gram 가운데 빈도수가 가장 높은 K개의 N-gram들을 그 시그니처로 정의한다. 모든 항목은 악성코드의 분류를 위해 데이터베이스에 저장된다. 새로운 악성코드가 입력되면 동일한 방법으로 N-gram 및 String을 추출하여, 기존의 시그니처들과 비교하여 동종 유사도 혹은 이종 여부를 계산해 낸다.

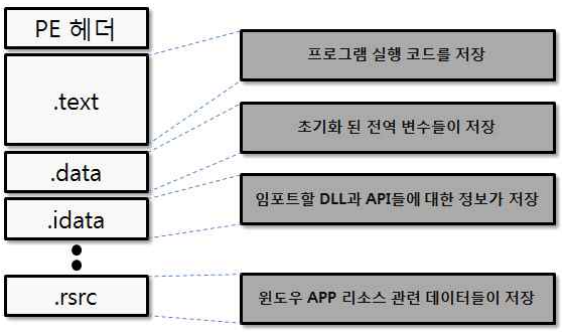


[그림 1] 전체 시스템 구조도

[Fig. 1] Whole System structure

3.2 Section Extraction using Entropy

기존의 N-gram 기법은 바이너리 전체를 대상으로 적용되었다. 모든 섹션 및 헤더(Header)에서 N-gram을 추출하여 그 빈도수를 계산하기 때문에, 빈도수가 낮은 N-gram은 유사도 계산에 영향을 끼치지 않는다. 또한, 'FF' 혹은 '00' 과 같이 섹션을 구분하고 조정(Align)하기 위해 사용되는 값들이 유사도 계산에 사용된다. 이러한 잘못된 입력 잡음(Input Noise) 때문에 악성코드 분류 시, 그 정확도가 떨어지는 결과를 초래한다. 따라서 보다 정확한 시그니처를 만들기 위해 본 논문에서는 .text 섹션에만 N-gram 기법을 적용한다. 이를 위해서는 섹션의 분류가 우선적으로 이루어져야 한다.



[그림 2] PE 구조 및 섹션 설명

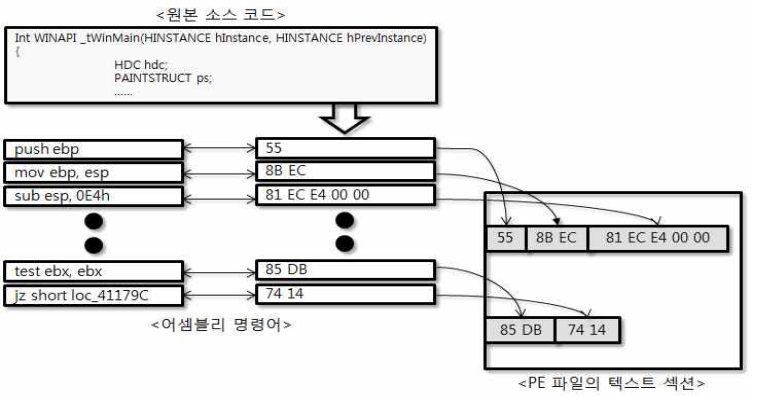
[Fig. 2] Description of PE structure and Section

윈도우 실행파일인 PE(Portable Executable)의 구조는 [그림 2]와 같으며, 헤더 정보 중 섹션 헤

더를 통해 섹션을 추출해 낼 수 있다. 하지만, 최근 악성코드들은 대부분 실행 압축 기술이 적용되어있거나, 섹션 명을 사용자 임의로 변환해 놓는다. 이는 단순히 Hex Editor를 통해서 손쉽게 변환 가능하다. 이러한 기술들이 적용되는 이유는 바이너리의 해쉬(Hash)값 등을 시그니처로 사용하는 안티 바이러스 기술을 우회하거나, 분석을 어렵게 하기 위함이다. 따라서, 분석을 위해 실행 압축 해제(Unpacking)을 한 후 섹션을 분류해 낼 수 있어야 한다. 본 논문에서는 .text 섹션 추출을 위해 섹션 별 엔트로피(Entropy)값을 이용한다. 섹션마다 내제되어 있는 정보의 종류와 그 방식이 다르므로 엔트로피의 값 또한 다르게 나타난다. 본 논문에서는 이러한 섹션 엔트로피 차이를 이용하여 섹션들을 분류한다.

3.3 N-gram on .text Section

.text section은 코드섹션으로도 불리며, 뜻 그대로 실행 명령어들을 담고 있는 섹션이다. 즉 작성된 바이너리의 모든 명령어들이 [그림 3]과 같은 형태로 모두 섹션 안에 존재한다. 따라서 본 섹션에 1-gram을 적용하는 경우, 단일 바이트 opcode의 빈도수를 근사하게 계산해 낼 수 있으며, 명령어들이 정렬(Align)되는 단위가 1byte 이므로 operand 들의 빈도수 또한 함께 계산 해 낼 수 있다. N 값으로 2 혹은 3 등을 적용하는 경우 2byte opcode, 3 byte opcode와 더불어 명령어의 빈도수 또한 근사하게 계산 해 낼 수 있다.



[그림 3] 텍스트 섹션의 생성

[Fig. 3] Creation of .text section

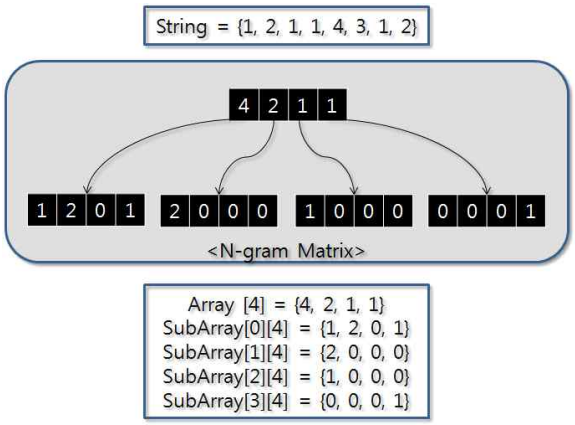
기존의 N-gram 기법에서는 byte를 대상으로 N-gram을 추출해 내는 데 약 99%의 시간이 걸리며, 파일 크기(n)가 선형적으로 증가함에 따라 수행 시간이 O(n2)의 형태로 증가한다. 이는, 매 N-gram 추출 시, 기존의 N-gram List에 존재하는지를 탐색하는데 시간이 소요되기 때문이다. 또한 한 번의 N-gram 실험 시 N의 값은 한 가지만 사용할 수 있다. 수행시간의 더딤은 정확한 시그니처를 작성하기 위해 적게는 수십에서 많게는 수백개 까지의 샘플에 다양한 N값을 정하여 N-gram을 수행하여야 하는 N-gram 기법에 치명적이다. 본 논문에서는 이러한 문제점을 해결하기

위해 기존의 리스트를 사용한 방법 대신 N-gram Matrix를 사용하였다. 나아가 Matrix를 여러 레벨(level)로 확장하여, 한번의 N-gram 실험으로 N을 다양하게 추출할 수 있는 Multi N-gram Matrix을 적용하였다.

3.3.1 Multi N-gram Matrix

기존의 N-gram 기법의 경우 N값 하나를 정해 이를 바이너리에 적용한다. N-gram은 선처리 과정이 필요하지 않다는 장점이 있지만, N 값이 높아질수록 list 내 비교·탐색 시간이 높아진다. 또한 분류의 정확도를 높이기 위해서는 N의 값을 다양하게 적용시키는 것이 필요하므로 N값에 따른 반복된 계산이 필요하다. 악성코드의 정확한 분류를 위해서는 그 시그니처를 만들기 위해 샘플을 적게는 수십개에서 많게는 수백개를 사용하여야 한다. 따라서 본 논문에서는 M. Sui 등[5]이 제안한 Variable N-Gram 기법을 이용하여 N-gram Matrix라는 기법을 구현하였다. 이 방법에서는 한 번의 N-gram 수행으로 1-gram, 2-gram, 3-gram list들을 모두 추출하여 저장한다.

이러한 다중 N-gram은 배열을 사용하여 구현되며, 배열의 각 인덱스(index)는 1바이트의 1-gram값을 의미한다. 1 바이트는 8bit로써 28개의 종류가 존재하므로, 하나의 배열에는 256개의 인덱스가 존재한다. 하나의 인덱스 배열 공간에는 해당 1-gram의 빈도수가 저장된다. 또한, 각각의 인덱스 배열에는 256개의 하위 배열이 존재한다. 한 개의 1-gram이 추출되는 경우 바로 다음에 추출되는 1-gram은 해당 인덱스의 하위 배열에 저장되게 된다. [그림 4]는 2bit 로 이루어진 스트링에 bit를 기준으로 N-bit-gram을 적용한 예를 나타낸다. 2 bit이므로, 하나의 배열 당 인덱스 개수는 22 = 4가 사용된다. 1-bit-gram을 사용하고자 하면 Array들 만을 비교하면 된다. Array[i]의 값은 스트링 내의 1-bit-gram ‘i’의 빈도수를 의미한다. 마찬가지로, Array[i][j]의 값은 스트링 내의 ‘ij’의 빈도수가 되며, 이는 스트링 내의 2-bit-gram ‘ij’의 빈도수를 의미한다. 본 논문에서는 이를 byte 수준으로 확장시켜 Multi N-gram Matrix를 작성한다.



[그림 4] Multi N-gram Matrix 기법

[Fig. 4] Multi N-gram Methodology

4. 실험 및 결과

4.1 실험 데이터

본 논문에서 제안한 방법의 실험을 위해 VX Heaven[12]으로부터 악성코드 샘플을 수집하였다. 수집한 악성코드는 트로이 목마(Trojan)으로 분류되며, DDoS(Distributed Denial of Service) 공격을 수행한다. 각 악성코드 샘플의 진단명은 Kaspersky 안티바이러스프로그램의 진단명을 사용하였다. 분별 가능성을 증명하기 위해 정상 바이너리 또한 실험에 사용하였으며, 일반적으로 윈도우 OS 상에서 사용되는 메모장(notepad.exe), 계산기(calc.exe), 익스플로러(Explorer.exe) 등을 사용하였다.

4.2 엔트로피 실험 결과

[표 2] [표 3]는 Boxed 패밀리 및 정상 바이너리들의 섹션별 엔트로피를 계산해 낸 결과이다. 대상 정상 바이너리들의 경우, 프로그램 내에서 사용된 API, DLL의 정보가 담겨있는 import table이 .idata section이 존재하지 않지만 .text 섹션에서 테이블이 명시되어 있었다. 따라서 .text 섹션의 엔트로피를 계산하는 과정에서 이를 제외하였으며, import table의 엔트로피를 .idata section의 엔트로피로 계산하였다. 결과적으로, .text 섹션이 섹션들 중 가장 높은 엔트로피를 보였으며, .idata, data 섹션 순으로 높게 나타났다.

[표 2] 정상 바이너리 파일의 섹션별 엔트로피 계산

[Table 2] Calculated Entropy of Benign binaries per section

분류	파일명	Section name		
		.text	.data	import table
Benign	notpad.exe	6.81	0.84	6.22
	ipconfig	6.9	3.63	6.14
	calc.exe	6.93	1.39	5.97
	iexplorer.exe	7.34	5.92	6.21

[표 3] Delf 패밀리의 섹션별 엔트로피 계산

[Table 3] Calculated Entropy of Delf families per section

분류		파일명	Section name		
			CODE	DATA	idata
Trojan Malware	Delf Family	Delf.af	6.87	3.22	4.7
		Delf.b	6.99	2.19	4.91
		Delf.c	7.02	4.54	5.72

[표 4]는 Boxed 패밀리의 엔트로피를 계산한 것으로, 특이점은 UPX 패킹이 이루어 졌다는 점이다. 분석을 하기 전 언패킹을 마쳤지만, 섹션 이름은 UPX0, UPX1, UPX2, .dswlab등으로 나뉘어 졌음을 발견하였다. 이를 직접적으로 헥사 에디터를 사용하여 확인해 본 결과, 가장 엔트로피가 높게 나타났던 UPX0 섹션은 명령어를 담고 있는 .text 섹션의 구조를 가지고 있었으며, .dswlab 섹

션은 .idata 섹션과 동일한 형태를 담고 있는 것을 확인할 수 있었다. 따라서, 기존의 안티 바이러스 기술을 회피하기 위한 섹션 명 변경 기술이 적용된 바이너리라도, 엔트로피를 계산해 내어 섹션 구분을 하는 것이 가능하다는 것을 알 수 있다.

[표 4] Boxed 패밀리의 섹션별 엔트로피 계산
[Table 4] Calculated Entropy of Boxed families per section

분류		파일명	Section name		
			UPX0	UPX1	dswlab
Trojan Malware	Boxed Family	Boxed.a	6.78	0.87	5.1
		Boxed.b	6.12	2.39	2.89
		Boxed.i	3.71	2.89	5.15

4.3 .text 섹션 N-gram 실험 결과

본 논문에서 제시하는 .text 섹션 N-gram을 구현하여 실험해본 결과는 [표 5]와 같다. 악성코드 샘플로는 Boxed 패밀리와 Delf 패밀리를 사용하였으며, K = 17, N = 4을 사용하였다. 트레이닝 셋(Training Set)은 Boxed.b 및 Boxed.d를 사용하였다. 따라서 계산된 유사도는 b와 d에 N-gram를 적용하여 얻어진 Boxed Familiy의 시그니처와 얼마나 유사한지를 나타낸다. 논문에서 제시한 섹션 N-gram의 대조군으로는 Santos[1]가 제시한 KNN-Ngram을 사용하였다. 실험 결과, 기존의 N-gram과 비교했을 경우 .text 섹션 N-gram의 수행시간이 현저히 적고, 동일 패밀리 별 유사도는 한층 증가됨을 확인 할 수 있었다. 반면에, 타 패밀리 및 정상 바이너리와의 유사도는 감소하였다. 따라서 .text 섹션에만 N-gram을 적용할 경우 더 정확하고 효율적인 시그니처 제작이 가능하다는 것을 알 수 있다.

[표 5] 섹션 N-gram 및 KNN-Ngram 의 실험 결과 비교
[Table 5] Comparison of experiments between Section N-gram and KNN-Ngram

분류		파일명	Ngram		.text Section Ngram	
			유사도	수행시간	유사도	수행시간
Trojan Malware	Boxed Family	Boxed.a	0.882	19.89	0.824	6.04
		Boxed.b	0.824	11.81	0.941	5.73
		Boxed.i	0.706	45.56	0.706	13.22
		Boxed.m	0.706	46.69	0.824	15.41
		Boxed.u	0.882	45.03	1	29.82
	Delf Family	Delf.af	0.471	12.43	0.353	9.33
		Delf.b	0.293	2.44	0.118	2.08
		Delf.c	0.353	1667	0.294	1280
Benign		notpad.exe	0.353	558.26	0.294	7.49
		ipconfig	0.471	3.57	0.294	2.70
		calc.exe	0.412	2116.95	0.294	688.66
		iexplorer.exe	0.176	3349.36	0.412	2018

4.3.1 Multi N-gram Matrix 결과

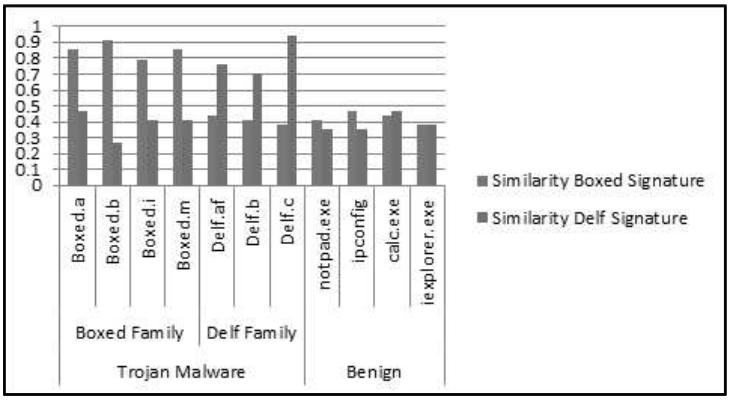
[표 6]는 기존 Santos[1]의 N-gram을 구현 및 실험해 본 결과, 각 단계별 수행 시간 및 비중을 나타낸 것이며, 앞서 언급한 바와 같이 대부분의 수행시간이 N-gram을 추출해 내는 데 소요되는 것을 알 수 있다.

[표 6] KNN-Ngram 및 Multi N-gram Matrix 수행 시 소요 시간 및 비중
[Table 6] Execution time of KNN-Ngram and Multi N-gram Matrix

	KNN-Ngram		Multi N-gram Matrix	
수행 동작	수행 시간	비중(%)	수행 시간	비중(%)
N-gram Extraction	77.891	99.8	0.71	96
KNN Algorithm Selection	0.003	0.01	0.01	1.4
Similarity Calculation	0.035	0.04	0.02	2.7
TOTAL	78.019	100	0.74	100

기존의 리스트를 사용한 N-gram의 경우 1-gram, 2-gram, 3-gram을 모두 적용하여 시그니처를 만드는 경우 77.891초가 소요되었다. 본 논문에서 제시한 Multi N-gram Matrix의 경우 0.74초이며, 기존 기법에 비해 1/100의 시간을 줄일 수 있었다.

각기 다른 N값에 대한 N-gram 시그니처를 사용하여 유사도를 계산 해 낼 수 있었고, 이를 통해 계산해 낸 통합 유사도는 [그림 5]에 나타나 있다. 전체적으로 유사도는 2-gram만을 적용한 섹션 N-gram 결과와 유사한 경향을 보였다. 일반적으로 동일 패밀리의 악성코드의 경우 0.7 ~ 0.95에 달하는 높은 유사도를 보였지만, 이종 패밀리 및 정상 프로그램과의 유사도는 0.4 이하의 낮은 값이 측정되었다. 따라서, 임계값(Threshold)을 0.7로 하면 오탐 없는 분류가 가능하다.



[그림 5] Multi N-gram Matrix를 통한 유사도 계산 결과
[Fig. 5] Similarity results of Multi N-gram Matrix

5. 결론 및 향후 연구

본 논문에서는 엔트로피 계산을 통해 섹션을 분할하여 .text 섹션에 N-gram을 적용하여 악성코드를 분류할 수 있는 방법을 제안하였다. Multi N-gram Matrix 기법을 통해 시그니처를 제작함으로써, 기존의 N-gram의 큰 약점인 수행시간을 1/100로 단축시킬 수 있었으며, 정확도를 높일 수 있었다. 그러나 모든 악성코드 샘플에 대하여 실험을 진행하지 않고, PE 파일만을 대상으로 진행하였기 때문에, 향후 다양한 악성코드 풀(Pool)을 구성하여 실험하여야 한다. 또한 본 논문에서 다루지 않은 다양한 섹션들에 대해서 분류에 그치지 않고, 알맞은 기법을 적용하여 시그니처를 제작하는 연구를 진행함으로써, 보다 정확하고 효율적인 분류 기법에 대한 연구를 지속하여야 한다.

참고문헌

[1] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, N-grams-based file signatures for malware detection, in Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS), Volume AIDSS, pp. 317–320, 2009

[2] Lai, Y. X., "A Feature Selection for Malicious Detection" Proc. of the 9th International Association for Computer and Information Science Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel./Distributed Computing, Phuket, Thailand, pp.365-370, 2008

[3] Parvin, B_Minaeil, et al, A New N-gram Feature Extraction-Selection Method for Malicious Code, in 2011 Springer-Verlag Berlin Heidelberg, ICANNGA, pp. 98 – 107, 2011

[4] Kolter, J.Z., Maloof, M.A.: Learning to detect malicious executables in the wild. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–478. ACM Press, New York, 2004

[5] M.-H. Siu and M. Ostendorf, Variable n-gram and extensions for conversational speech language modeling, IEEE Trans. Speech Audio Processing, vol. 8, pp. 63–75, Jan. 2000.

[6] Moskovitch R, Feher, Tzachar N, Berger E, Gitelman M, Dolev S, et al. Unknown malcode detection using OPCODE representation. In: European conference on intelligence and security informatics 2008 (EuroISI08), Esbjerg, Denmark; 2008.

[7] Robert Lyda and James Hamrock. Using Entropy Analysis to Find Encrypted and Packed Malware. IEEE Security & Privacy, 5(2):40{45, March 2007.

[8] Min-Jae Kim, et al. "Design and Performance Evaluation of Binary Code Packing for Protecting Embedded Software against Reverse Engineering" in 13th IEEE International Symposium, (ISORC), pp. 80 – 86, 2010.

[9] Han, S., Lee, K., Lee, S.: Packed PE File Detection for Malware Forensics. In: 2nd International Conference on Computer Science and its Applications, CSA, Korea, 2009.

[10] 권오철, 배성재, 조재익, 문종섭, Native API 빈도 기반의 퍼지 군집화를 이용한 악서코드 재그룹화 기

법 연구, 정보보호학회 논문지, 제 18권, 제 6호, pp. 115 – 127, 2008.

[11] 한경수, 김인경, 임을규, API 순차적 특징을 이용한 악성코드 변종 분류 기법, 보안공학연구논문지 (Journal of Security Engineering), 제 8권 제 2호, pp. 319 – 335, 2011.

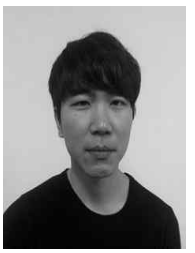
[12] http://vx.netlux.org/

[13] Boyun Zhang, Dingxing Zhang, Jianping et al, New Malicious Code Detection Based on N-Gram Analysis and Rough Set Theory, Springer-Verlag Berlin Heidelberg, 2007.

저자 소개

권희준 (Hee-jun Kwon)

2011년 2월 : 한양대학교 컴퓨터전공 학사
2011년~현재 : 한양대학교 전자컴퓨터통신공학과 석사과정
관심분야 : 악성코드 분석, 네트워크 보안, 정보보호



김선우 (Sunwoo Kim)

2012년 2월 : 한양대학교 컴퓨터전공 학사
2012년~현재 : 한양대학교 전자컴퓨터통신공학과 석사과정
관심분야 : 네트워크 보안, 악성 프로그램 분석, 정보보호

임을규 (Eul Gyu Im)

1992년 : 서울대학교 컴퓨터공학과 학사
1994년 : 서울대학교 컴퓨터공학과 석사
2002년 : University of Southern California 컴퓨터과학과 박사
2002년~현재 : 한양대학교 컴퓨터공학부 부교수
관심분야 : 네트워크 보안, 악성 프로그램 분석, RFID 보안, SCADA 보안