Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.5, Oct. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.5.897

효율적인 악성코드 분류를 위한 최적의 API 시퀀스 길이 및 조합 도출에 관한 연구*

최 지 연,[1,2†] 김 희 석,[1] 김 규 일,[1] 박 학 수,[1] 송 중 석[1,2‡]
[1]한국과학기술정보연구원, [2]과학기술연합대학원대학교

A study on extraction of optimized API sequence length and combination for efficient malware classification*

Ji-yeon Choi,[1,2†] HeeSeok Kim,[1] Kyu-il Kim,[1] Hark-soo Park,[1] Jung-suk Song[1,2‡]
[1]Korea Institute of Science and Technology Information,
[2]Korea University of Science & Technology

요 약

인터넷이 지속적으로 발달하면서 이에 따른 부작용으로 사이버 해킹 공격 또한 지능적인 공격으로 진화하고 있다. 해킹 공격의 도구로 사용되는 악성코드는 공격자들이 자동 제작 툴을 이용해 손쉽게 악성코드를 생성할 수 있기 때문에 악성코드의 수가 급증하고 있다. 그러나 수많은 악성코드를 모두 분석하기에는 많은 시간과 노력이 요구됨에 따라 신·변종 악성코드에 대한 별도의 분류가 필요한 상황이다. 이에 따라 신·변종 악성코드를 분류하는 다양한 연구들이 등장하고 있으며, 해당 연구들은 악성코드 분석을 통해 악성 행위를 나타내는 다양한 정보를 추출하고 이를 악성코드를 대표하는 특징으로 정의하여 악성코드를 분류한다. 그 중, 대부분이 API 함수와 API 함수로부터 추출한 특정 길이의 API 시퀀스를 이용하여 악성코드를 분류하고 있다. 그러나 API 시퀀스의 길이는 분류의 정확성에 영향을 미치기 때문에 적합한 API 시퀀스의 길이를 선택하는 것이 매우 중요하다. 따라서 본 논문은 특정 길이에 한정하지 않고, 다양한 길이의 API 시퀀스를 생성 및 조합하여 악성코드 분류의 정확성을 향상시키기 위한 최적의 API 시퀀스 및 조합을 찾는 방법론을 제안한다.

ABSTRACT

With the development of the Internet, the number of cyber threats is continuously increasing and their techniques are also evolving for the purpose of attacking our crucial systems. Since attackers are able to easily make exploit codes, i.e., malware, using dedicated generation tools, the number of malware is rapidly increasing. However, it is not easy to analyze all of malware due to an extremely large number of malware. Because of this, many researchers have proposed the malware classification methods that aim to identify unforeseen malware from the well-known malware. The existing malware classification methods used malicious information obtained from the static and the dynamic malware analysis as the criterion of calculating the similarity between malwares. Also, most of them used API functions and their sequences that are divided into a certain length. Thus, the accuracy of the malware classification heavily depends on the length of divided API sequences. In this paper, we propose an extraction method of optimized API sequence length and combination that can be used for improving the performance of the malware classification.

Keywords: Malware classification, API sequence, Length, Combination

접수일(2014년 9월 2일), 수정일(2014년 9월 24일), 게재확정일(2014년 9월 26일)

* 본 연구는 2014년도 한국과학기술정보연구원의 「대용량 보안 이벤트 자동검증 고도화 기술 연구사업」의 지원을 받아 수행된 연구임 (K-14-L06-C15)

† 주저자, ji935@kisti.re.kr

‡ 교신저자, song@kisti.re.kr(Corresponding author)

I. 서 론

인터넷의 지속적인 발달에 따라 전 세계적으로 새로운 서비스가 제공되고 다양한 기술이 등장하고 있다. 그러나 이에 따른 부작용으로 사이버 해킹 공격이 증가하고 있으며, 동시에 지능적인 공격으로 진화하고 있다. 이러한 사이버 공격의 도구로 사용되고 있는 악성코드는 Neon Exploit System, Eliesta 등의 자동 제작 프로그램을 사용하여 손쉽게 생성할 수 있을 뿐만 아니라 누구나 어렵지 않게 제작 툴을 구할 수 있기 때문에, 그로 인한 악성코드의 수가 매년 기하급수적으로 증가하고 있다[1].

급격하게 증가하는 악성코드에 대응하기 위해 대부분의 백신 업체가 악성코드를 분석하는데 많은 인력과 시간을 투자하고 있지만, 급증하는 악성코드를 모두 분석하기에는 역부족인 상황이다. 이러한 문제를 해결하고자 기존에 발견된 악성코드보다는 신·변종 악성코드에 대한 심층 분석을 수행하기 위해 악성코드를 기존에 알려진 것과 신·변종으로 분류하기 위한 다양한 연구들이 등장하고 있다.

악성코드 분류 연구를 수행하기 위해서는 악성코드를 분석하여 악성코드의 행위를 내포하고 있는 정보들을 추출해야 하며, 주로 API 함수, String, 해시, 네트워크 연결에 관한 로그 등이 이에 해당된다. 따라서 대부분의 분류 연구들은 앞서 추출한 정보를 악성코드의 대표 특징으로 정의하고, 이를 이용해 악성코드 간 유사도를 계산하여 신·변종 악성코드를 분류하는 방법론을 제안하고 있다.

그 중 많은 연구들이 악성코드가 실행되면서 호출하는 API 함수의 호출 빈도나 API 함수의 시퀀스 정보(API 함수를 순서에 따라 특정 길이로 분할한 정보)를 악성코드의 분류를 위한 대표 특징으로 선택하고 있다. 그러나 유사한 악성코드라 할지라도 API 함수 호출 순서가 동일하지 않은 경우가 많기 때문에 API 시퀀스를 어떤 길이로 분할하는가에 따라 악성코드 분류의 정확성이 좌우될 수 있다.

따라서 본 논문은 특정 길이에 한정하지 않고, 다양한 길이의 API 시퀀스를 생성 및 조합하여 악성코드 분류의 정확성을 향상시키기 위한 최적의 API 시퀀스 길이 및 조합을 찾는 방법론을 제안한다. 이를 위해 동적 분석을 수행하여 API 함수를 추출하고 이로부터 각각의 길이에 따른 API 시퀀스를 생성 및 조합하여 악성코드의 특징으로 선택한다. 또한 앞서 선택한 악성코드의 특징을 바탕으로 악성코드 간의 유사도를 측정하고, 측정한 유사도를 기반으로 악성코드 분류의 오류율을 계산함으로써 분류의 정확성을 판단한다. 이를 바탕으로 본 논문의 실험을 진행한 결과, 다양한 길이의 API 시퀀스를 조합한 경우보다 API 함수를 2개씩 결합한 길이의 API 시퀀스가 더 높은 분류 정확성을 보임에 따라 이를 최적의 API 시퀀스로 도출한다. 따라서, 이를 통해 도출한 결과를 향후 API 시퀀스에 기반을 둔 악성코드 분류 연구에 활용할 경우, 분류의 정확도를 극대화하는데 많은 기여를 할 수 있을 것으로 판단된다.

본 논문의 구성은 다음과 같다. 2장에서는 악성코드 분석 방법에 따른 분류 연구 및 API 함수를 이용한 분류 연구에 대하여 기술하고, 3장에서는 최적의 API 시퀀스 길이 및 조합을 찾기 위한 방법을 제안한다. 4장에서는 제안한 방법을 바탕으로 실험을 수행함으로써 결과를 도출하며, 마지막으로 5장에서는 결론을 제시한다.

II. 관련 연구

2.1 정적 분석 기반 분류 연구

악성코드 분류와 관련된 많은 연구[2]들이 악성코드의 행위에 관한 특징 정보들을 추출하기 위해 주로 정적 분석과 동적 분석을 수행하고, 이를 통해 추출한 정보를 악성코드의 특징으로 사용하고 있다. 먼저 정적 분석은 악성코드를 실행하지 않은 상태에서 악성코드의 바이너리 실행파일에 포함된 문자열을 추출하고 추출한 문자열을 분석함으로써 악성코드의 구조와 특성을 파악할 수 있다.

보통 정적 분석은 악성코드의 PE(Portable Executable) 구조를 분석하는 것이 일반적이다. PE 구조란 Microsoft의 운영체제 Windows 3.1부터 지원되는 실행파일의 구조를 의미하며, PE 구조를 사용하는 파일의 확장자는 cpl, exe, dll, ocx, vxd, sys, scr, drv가 있다[3].

연구[4]는 악성코드의 PE 구조를 기반으로 퍼지해시 기법을 이용한 악성코드 간의 유사도 비교기법을 제안하였는데, 제안한 방법은 악성코드의 특징으로 PE구조의 .data섹션을 추출하고, 이를 기반으로 퍼지해시를 생성하여 악성코드를 비교 및 분류하였다.

또한 악성코드 정적 분석을 위한 다양한 프로그램들이 개발되었는데, 대표적으로 OllyDbg[5], Imm

unity Debugger[6], IDA Pro[7] 등의 디스어셈블 도구가 이에 해당한다. 연구[8]은 이 중 IDA Pro 툴을 이용하여 악성코드 분석을 수행해 악성코드 내 함수의 길이를 추출하고, 이를 악성코드 분류를 위한 특징으로 사용하였다.

악성코드 자동분류 시스템을 제안한 연구[9] 역시 IDA Pro를 사용하여 악성코드를 분석함으로써 의미 있는 문자열을 추출하고, 이를 악성코드 특징으로 사용하여 다른 악성코드와의 유사성을 비교한다.

2.2 동적 분석 기반 분류 연구

정적 분석은 악성코드를 실행하지 않고 바이너리 실행파일을 분석하기 때문에, 탐지 기법을 우회하기 위해 난독화 또는 은닉화 된 악성코드를 분석하는 것이 불가능하다. 이러한 문제점을 극복하기 위해 동적 분석이 수행되는데, 동적 분석은 악성코드를 실행시켜 실제 악성코드의 행위를 분석하는 방법이다. 따라서 악성코드를 모니터링 함으로써 실제 악성코드의 행위를 포함하고 있는 정보들을 추출하고, 이를 유사성 비교를 위한 특징으로 사용하여 악성코드를 분류할 수 있다.

이를 위해 연구[10]에서는 악성코드의 행동을 모니터링 하여 악성코드의 행위 정보를 자동으로 추출할 수 있는 API Capture툴을 제안하였는데, API Capture는 시스템 콜 인자 및 악성코드의 주요 속성을 자동으로 기록할 수 있다.

악성코드의 시스템 콜 상태 변화를 이용하여 악성코드를 자동으로 분류하기 위한 방법을 제안한 연구[11]는 가상머신에서 악성코드를 실행시켜 네트워크에서 이루어지는 악성코드의 행위를 관찰한다. 이를 통해 추출한 시스템 콜의 지속적인 상태 변화를 특징으로 정의하여 악성코드를 그룹별로 분류하는데 이용한다.

또한, 악성코드의 행동관련 정보를 이용한 분류 연구[12]도 제안되었다. 악성코드를 클러스터링 하는데 사용되는 행동관련 정보(behavior profiles)에는 악성코드의 오염정보(taint tracking)를 추적한 내용과 네트워크 행위 등이 포함되어 있다.

2.3 API 함수를 이용한 악성코드 분류 연구

2.1 및 2.2에서 언급한 바와 같이 악성코드 분석을 통해 추출한 다양한 정보(API 함수, String, 해시, 네트워크 연결에 관한 로그 등)가 악성코드 분류를 위한 특징으로 사용되고 있다. 그 중, API 함수를 이용한 분류 연구가 가장 활발하게 진행되고 있는데 API(Application Programming Interface)란 응용 프로그램에서 사용할 수 있도록 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 의미한다[13].

API 함수는 악성코드가 실행되면서 파일을 생성하거나 다운로드 하는 등의 악의적인 행위에 대한 정보를 모두 포함하고 있다. 또한, 악성코드로부터 API 함수를 추출하는 방법은 정적 분석을 통해 API 함수가 저장되어 있는 악성코드 PE파일의 IAT(Import Address Table)를 추출하는 방법과, 동적 분석을 수행하여 악성프로그램 실행 중 호출되는 API를 후킹(hooking)하는 방법이 존재한다.

정적 분석을 수행하여 악성코드 API를 추출하는 연구들[14][15]은 PE파일 내 IAT로부터 API 함수들을 추출하고, Fig.1.과 같이 악성코드가 실행되면서 호출하는 API 함수의 순서를 이용하여 API 시퀀스를 생성하였다. 또한 이를 악성코드 특징으로 사용하여 유사도를 계산함으로써 악성코드를 분류하는 연구를 제안하였다.

동적 분석을 통해 API 함수를 추출하는 연구들[16][17]은 가상 머신에서 악성코드를 실행시켜 악성코드가 실제로 호출하는 API 함수들을 추출한다. 이들은 추출된 API 함수로부터 호출 빈도를 측정하여 CAPI(Critical API)를 생성하거나, 호출 순서를 기록하고 이를 그래프로 표현하여 유사도를 비교함으로써 악성코드를 분류한다.

앞서 소개된 기존 연구들은 악성코드 특징으로 API 함수의 호출 빈도나 특정 길이의 API 시퀀스를 이용해 악성코드 분류를 위한 방법론을 제안하고 있다. 그러나 API 시퀀스의 길이에 따라 악성코드 분류의 정확성이 좌우될 수 있기 때문에 본 논문에서는 다양한 길이의 API 시퀀스를 생성 및 조합하여

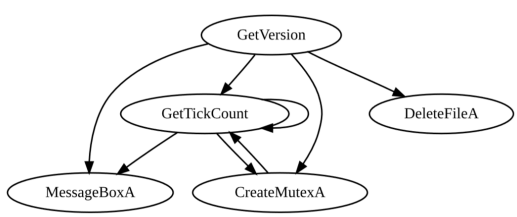

Fig. 1. Example of API Sequence

악성코드 분류의 정확성을 향상시키기 위한 최적의 API 시퀀스 길이 및 조합을 찾는 방법을 제시한다.

III. 방법론

본 논문에서는 API 함수를 이용한 악성코드 분류의 정확성을 향상시키기 위해 최적의 API 시퀀스 길이 및 조합을 도출하기 위한 방법론을 제안한다.

제안하는 방법에서는 악성코드에 대해 동적 분석을 수행하여 API 함수를 추출하고, 이를 바탕으로 다양한 길이의 API 시퀀스를 생성 및 조합한다(3.1 참조). 또한 3.1에서 생성 및 조합한 API 시퀀스에 해당하는 데이터들을 정규화 및 압축한다(3.2참조). 이어서 3.2에서 정규화 및 압축한 데이터를 바탕으로 악성코드 간 유사도를 계산한다(3.3참조). 유사도 계산이 완료된 후, 유사도 계산 결과를 기반으로 악성코드 분류 오류율을 측정함으로써(3.4참조) 최적의 API 시퀀스 길이 및 조합을 도출한다.

3.1 특징 추출

악성코드 분류를 수행하기 위한 최적의 API 시퀀스 길이 및 조합을 찾기 위하여 악성코드에 대한 동적 분석을 통해 추출한 API 함수를 바탕으로 악성코드 특징을 추출한다. 이를 위해, 가상환경에서 실제 악성프로그램을 동작시켜 수행되는 악성 행위를 모니터링 함으로써 프로그램 실행 중 호출되는 API 함수를 모두 추출한다. 이때, 본 논문에서는 서로 유사한 악성코드로부터 추출한 API 함수명 및 호출 순서는 전반적으로 동일하다는 가정 하에, 악성코드의 연속적 행위를 비교할 수 있도록 API의 호출 순서대로 API 함수를 추출한다.

본 단계는 추출한 API 함수를 바탕으로 두 단계를 수행하여 최적의 API 시퀀스 길이 및 조합을 찾기 위한 데이터를 생성하는데 먼저 3.1.1은 API 함수들을 특정 길이의 블록(block)형태로 분할하여 호출 순서를 포함하는 API 시퀀스를 생성하며, 3.1.2는 특정 길이에 따라 분할된 API 시퀀스를 서로 조합하는 과정이다.

3.1.1 API 시퀀스 생성

각 악성코드에 대한 동적 분석을 통해 추출한 API 함수로부터 API 시퀀스를 추출하기 위해 본 논문에서는 *N*-gram[18] 기법을 이용한다.

N-gram이란 문자열을 *N*값의 서브스트링(sub-string)으로 나누는 것을 의미하는데, 본 단계는 *N*-gram 기법을 사용함으로써 추출한 API 함수들을 일정한 길이인 *N*에 따라 블록 형태로 분할하고 분할된 블록들을 API 시퀀스로 정의한다.

API 시퀀스 생성 방법은 Fig.2.와 같다. 예를 들어, 두 개의 악성코드(malware1, malware2)를 가정하고 이로부터 2-gram의 API 시퀀스를 생성하고자 한다. API 시퀀스 생성 방법은 API 함수로부터 순서대로 하나씩 오른쪽으로 이동하며 2개의 블록으로 분할하여 API 시퀀스를 생성한다. malware 1의 API 함수는 A, B, C, D이며 malware 2의 API 함수는 A, C, D, E일 때 API 시퀀스를 생성한 결과, malware 1은 AB, BC, CD가 생성되며 malware 2는 AC, CD, DE가 생성된다. 또한 이로부터 생성된 각 악성코드의 API 시퀀스에 해당하는 각각의 호출 빈도수를 측정한다.

호출 빈도수는 모두 벡터(vector) 형태로 존재하며 이는 3.3에서 수행되는 악성코드 간 유사도 계산을 위해 사용된다. 이때, 벡터 간의 계산을 위해서는 차원을 일치시켜야하므로 모든 API 시퀀스의 합집합을 취하여 서로 다른 악성코드를 모두 동일한 API 시퀀스 및 순서로 일치시켜준다. 만약 malware 1에 대해 기존에는 없었던 AC라는 API 시퀀스가 추가된 경우에는, 해당 API 시퀀스의 호출 빈도수를 0으로 채워준다.

본 단계에서는 악성코드 분류의 정확도를 극대화하기 위한 최적의 API 시퀀스 길이를 도출하기 위해, 길이가 1인 API 시퀀스부터 차례로 길이를 1씩 증가시키며 다양한 길이의 API 시퀀스 후보를 생성한다.

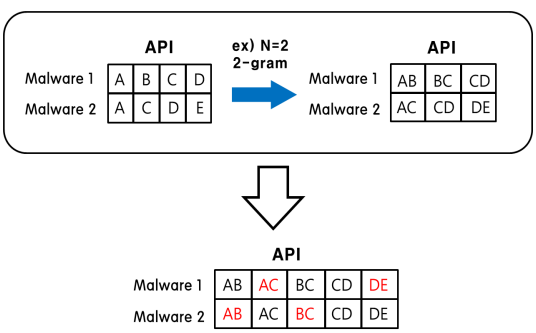

Fig. 2. Example of generating API sequence

3.1.2 API 시퀀스 조합

기존의 많은 분류 연구들은 이전 단계에서 생성한 특정 길이의 API 시퀀스를 단독으로 사용하여 악성코드를 분류하는데 사용한다. 그러나 본 논문에서는 추가적으로 각 길이의 API 시퀀스들에 대한 조합이 분류의 정확성에 어떠한 영향을 미치는지를 평가하기 위해 3.1.1에서 생성한 다양한 길이의 API 시퀀스를 조합한다.

만약 이전 단계에서 API 시퀀스 1-gram과 2-gram을 추출한 경우, 이에 대한 조합 방법은 Fig.3.과 같다. 앞서 생성한 API 시퀀스 1-gram 및 2-gram에 대한 조합을 수행하는 방법은 각각의 악성코드에 대한 1-gram의 API 시퀀스 A, B, C, D, E와 2-gram의 API 시퀀스 AB, BC, CD, DE를 모두 결합하는 방식이다. 따라서 해당 예에서는 1-gram의 API 시퀀스, 2-gram의 API 시퀀스와 이들을 조합한 1-gram& 2-gram의 API 시퀀스, 이렇게 총 3개의 API 시퀀스 및 조합 결과를 추출할 수 있다.

본 단계에서 각 길이의 API 시퀀스를 조합한 개수는 수식 1과 같으며, 수식 1의 n은 3.1.1에서 마지막으로 생성된 N-gram의 API 시퀀스, 즉 N의 값에 해당된다. 또한 r의 초기 값은 2로, 조합이 수행될 때마다 1씩 증가시켜주어 r이 n의 값과 일치할 때까지 반복적으로 조합을 수행한다.

$$ {}_nC_r = \frac{n!}{r!(n-r)!} \quad (1) $$

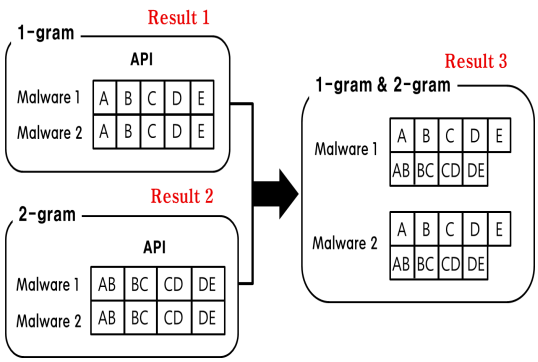

Fig. 3. Example of combining API sequence

3.2 정규화 및 압축

3.1.2에서 생성된 API 시퀀스 및 조합의 값에 대한 정규화를 수행한다. 앞서 생성한 각 길이의 API 시퀀스 및 조합에 대한 데이터는 모두 벡터 형태이며, 각 API 시퀀스에 해당하는 호출 빈도수를 포함하고 있다. 또한 호출 빈도수의 값은 서로 다른 단위의 크고 작은 값들로 이루어져 있으며 매우 넓은 범위를 형성하고 있다. 따라서 이러한 값들로 유사도를 계산할 경우, 유사도 결과가 큰 단위의 값에 의존되기 때문에 이를 방지하고자 값의 범위를 일치시키는 정규화 작업을 수행한다.

정규화 방법은 Fig.4.와 같은 과정으로 수행되며, 특정 악성코드로부터 생성한 API 시퀀스 호출 빈도수로 이루어진 벡터에 대해 (각 원소가 갖는 값)÷(각 원소가 갖는 값의 전체 합)의 식에 따라 정규화를 수행한다. 예를 들어, malware n에 대한 API 시퀀스 호출빈도 수가 A, B, C, D, E라고 하면, 이에 대한 정규화 후의 값은 차례로 A÷(A+B+C+D+E), B÷(A+B+C+D+E), C÷(A+B+C+D+E), D÷(A+B+C+D+E), E÷(A+B+C+D+E)가 된다.

이러한 작업은 모든 조합 결과에 대해 동일하게 수행되며, 정규화를 수행한 데이터들은 압축과정을 통해 유사도 계산에 필요한 최소의 데이터를 추출하게 된다. 본 연구에서는 압축을 위해 수행되는 여러 기법 중 PCA(Principle Component Analysis)[19]를 사용하여 데이터를 압축한다.

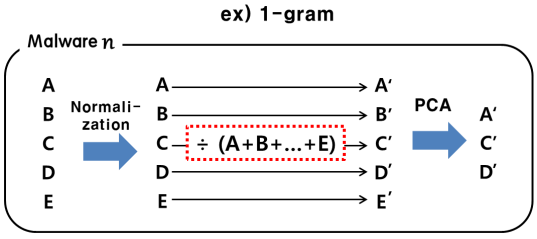

Fig. 4. Process of normalization and compression

3.3 유사도 계산

본 단계에서는 정규화 및 압축된 데이터로부터 악성코드의 유사도를 계산한다. 계산은 3.1.2에서 조합된 데이터 각각에 대해 개별적으로 수행되기 때문

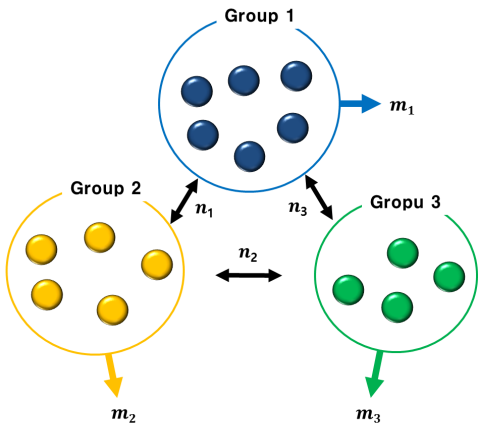

Fig. 5. Process of calculating similarity

에 총 ${}_nC_r$번의 계산을 진행하게 된다.

계산 방법은 Fig.5.와 같이 먼저 동일 그룹 내에 포함되어있는 악성코드들끼리의 유사도를 계산하여 이를 X_1이라 정의하고, 타 그룹에 속하는 악성코드와의 유사도를 모두 측정하고 이를 X_2로 정의한다.

각 악성코드의 유사도 계산은 피어슨 상관계수 분석 기법을 이용하며 이를 수식으로 표현하면 수식2와 같다.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \quad (2)$$

두 개의 악성코드 X와 Y에 대한 유사도는 X와 Y의 공분산($cov(X,Y)$)을 X와 Y에 대한 각각의 표준편차($\sigma_X\sigma_Y$)로 나누어줌으로써 계산할 수 있다.

3.4 최적의 API 시퀀스 길이 및 조합 도출

최적의 API 시퀀스 길이 및 조합을 도출하기 위해 악성코드 분류에 대한 오류율을 측정한다. 본 단계를 진행하기 위해 3.3에서는 각 길이의 API 시퀀스 및 조합 별로 악성코드 그룹 간의 유사도 계산을 수행하였다.

먼저 분류 오류율은 각 길이의 API 시퀀스 또는 API 시퀀스의 조합이 악성코드 간의 유사성을 효과적으로 비교할 수 있는지, 또한 악성코드 분류를 얼마나 정확하게 수행할 수 있는 특징인지를 판단하기 위한 지표이다.

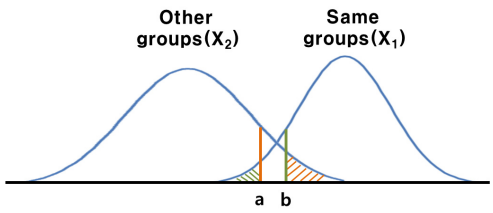

Fig. 6. Cumulative probability distribution

따라서 오류율은 Fig.6.과 같이 누적확률분포를 적용하여 계산하며, 빗금으로 표현된 범위에 대한 누적 확률을 구하고자 한다. 이를 수식으로 표현하면 수식 3과 같다.

$$ER = P(X_1 < a) + P(X_2 > b) \quad (3)$$

수식 3에서 X_1은 동일 그룹 내에 속하는 악성코드 사이의 유사도를 그룹 구분 없이 전부 결합한 값이며, X_2는 타 그룹에 속하는 악성코드와의 유사도에 대한 전체의 값을 의미한다. 또한 a는 X_2의 최대값이며, b는 X_1의 최소값을 의미하고 X_1의 평균은 m_1, 분산은 σ_1로 정의한다. 마찬가지로 X_2의 평균은 m_2, 분산은 σ_2와 같다. 따라서 수식3은 다시 수식4로 표현할 수 있다.

$$ER = P(Z < \frac{a-m_1}{\sigma_1}) + P(Z > \frac{b-m_2}{\sigma_2}) \quad (4)$$

본 단계에서는 수식4를 바탕으로 각 조합에 대한 오류율을 모두 구하고, 가장 적은 오류율을 갖는 길이의 API 시퀀스 또는 API 시퀀스의 조합을 효율적인 악성코드 분류를 위한 최적의 특징으로 도출한다.

IV. 실험 및 결과

4.1 실험환경 및 데이터

제안한 방법의 성능을 평가하고, 악성코드 분류를 위한 최적의 API 시퀀스 길이 및 조합을 도출하기 위해 과학기술사이버안전센터(S&T SEC)[20]의 허니팟(honeypot)으로부터 수집한 악성코드 샘플 143개를 이용하여 실험을 수행하였다. 수집한 악성코드는 Backdoor, Email worm, Network

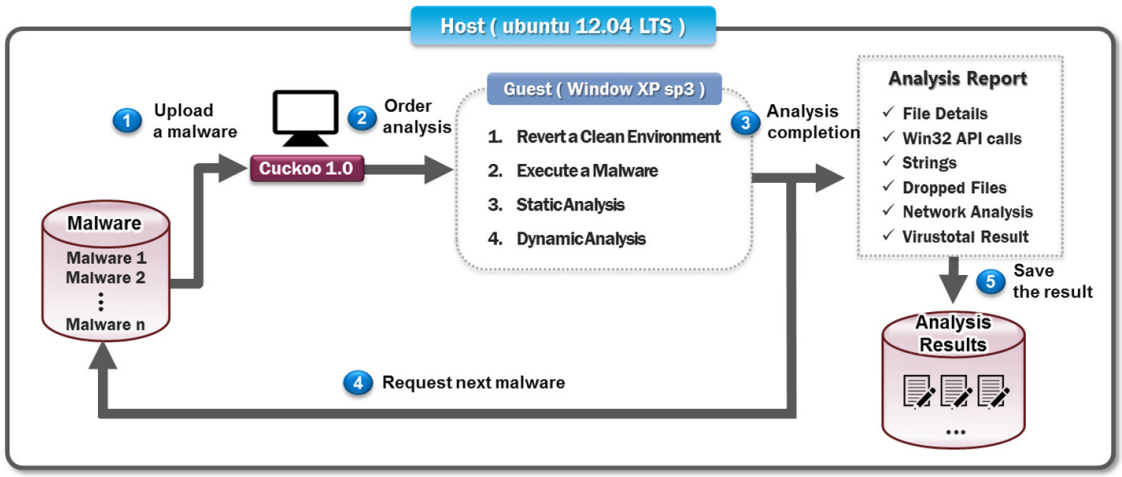

Fig. 7. Experimental environment

worm, Virus, Trojan 등의 공격에 포함된다.

본 논문에서 제안한 악성코드 분류 방법의 정확도 평가하기 위해서는 악성코드 그룹에 대한 정답 데이터가 필요하다. 이를 위해 VirusTotal[21]에 143개의 악성코드 샘플을 업로드 하여 각 악성코드의 진단명을 추출하였다. 각 악성코드 샘플에 대한 진단명은 Microsoft, Avast, AVG 안티바이러스의 진단명을 사용하였으며[22], 동일한 진단명을 갖는 악성코드 샘플들의 경우 유사 행위를 하는 악성코드로 판단하여 이를 같은 그룹으로 분류하였다. 그 결과, Table 1에서 보는 바와 같이 5개의 그룹을 생성하였으며, 각 그룹에 해당되는 97개의 악성코드 샘플을 재추출하였다. 따라서 동일한 진단명을 바탕으로 Trojan의 하위 진단명을 갖는 R1, R3, R4와 Email worm의 하위 진단명을 갖는 R2와 R5를 생성하여 실험을 진행하였다.

본격적인 실험을 위해, 악성코드 분석 시스템인 쿠쿠(Cuckoo)[23] 샌드박스를 이용하여 앞서 추출한 97개의 악성코드로부터 API 함수를 수집한다. 쿠쿠는 가상환경에서 실제 악성프로그램을 동작시킴으로써 악성코드 실행 시 수행되는 악성행위를 모니터링 및 분석하는 시스템이다.

악성코드 분석 결과 및 API 함수를 수집하기 위한 실험 환경은 Fig.7.과 같으며, Ubuntu 12.04 LTS가 설치된 호스트와 호스트 내부에 Windows XP sp3가 설치된 게스트를 구성하였다. 게스트는 외부의 환경에 영향을 받지 않는 가상 환경 시스템으로 설정되어있다. 또한 쿠쿠 시스템은 게스트에서 악성프로그램을 실행 및 모니터링 함으로써 해당 프로그램의 분석 결과를 제공하며, 상세 동작 과정은 다음과 같은 순서로 진행된다.

① 분석을 원하는 악성코드 샘플을 쿠쿠 시스템에 업로드 한다.
② 쿠쿠 시스템은 게스트를 악성코드 실행 전 환경으로 복구하고, 업로드 된 악성코드를 게스트 내에서 실행시켜 악성코드의 행위를 모니터링 한다.

Table 1. The number of elements of malware groups

	Microsoft	Avast	AVG	# (Rx)
R1	PWS: Win32/ OnLine Games. xx	Win32: Crypt-IGU [Trj]	Win32/ Patched. FZ	36
R2	Virus: Win32/ Sality.G	Win32: Sality-U	Win32/ Sality	27
R3	Trojan: Win32/ Yoddos.A	Win32: Trojan -gen	Dropper. Agent. RCT	13
R4	Trojan Dropper:Win32/ Oficla.x	Win32:Malware -gen	Genericxx .xxxx	12
R5	Virus: Win32/ Virut.xx	Win32:Mydoom-x [Wrm]	Win32/ Virut.A	9

또한 모니터링 결과를 바탕으로 정적 분석과 동적 분석을 수행한다.

③ 분석이 완료되면, 악성코드 행위에 대한 다양한 정보를 포함한 분석 보고서가 생성된다.

④ 쿠쿠 시스템은 ①단계로 이동하여 다음 악성코드에 대해 동일한 작업을 반복 수행한다.

⑤ 악성코드 분석 보고서는 호스트에 파일로 저장되며, 사용자에게 제공된다.

4.2 API 시퀀스 생성 및 조합 결과

본 단계에서는 4.3에서 수행되는 악성코드 유사도 계산을 위해 악성코드 분석 결과로부터 API 함수를 추출하고, API 시퀀스를 생성 및 조합한다.

먼저 Table 2는 API 시퀀스를 생성한 결과를 보여준다. 본 실험에서는 1-gram의 길이부터 4-gram의 API 시퀀스를 생성하였으며, 97개 악성코드 샘플로부터 각각의 길이에 해당하는 API 시퀀스를 모두 추출하였다. 그 결과, 1-gram의 경우 97개 악성코드 샘플로부터 추출한 API 시퀀스는 101개였으며 API 시퀀스의 길이가 길어질수록 그에 해당하는 전체 API 시퀀스의 개수가 증가하는 것을 확인할 수 있다.

Fig.8.은 하나의 악성코드로부터 추출한 API 함수를 2-gram의 길이로 분할하여 API 시퀀스를 생성한 결과의 예를 보여준다. 좌측은 API 함수가 2개씩 분할되어 있는 것을 확인할 수 있으며, 우측은 결합된 해당 API 시퀀스의 호출 빈도수를 측정하여 기록한 내용이다. 특히 좌측에서 유사한 API 함수명 및 동일한 기능을 수행하는 API 함수가 반복되는 모습을 볼 수 있는데, 본 논문에서는 서로 유사한 악성코드로부터 추출한 API 함수명 및 호출 순서는 전반적으로 동일하다는 가정 하에 반복되는 동일 기능의 API 함수를 별도로 제외하지 않고 전부 기록하였다.

또한 수집한 악성코드 샘플 97개 역시 모두 위와 같은 동일한 형태로 데이터를 포함하고 있다.

Table 2. Results of generating API sequence

Length of API sequence	The number of API sequence
1-gram	101
2-gram	1496
3-gram	6314
4-gram	15921

```
LdrGetProcedureAddress_LdrGetProcedureAddress  94
RegOpenKeyExW_RegOpenKeyExW                    90
RegQueryValueExW_RegCloseKey                   79
NtOpenKey_NtOpenKey                            70
RegCloseKey_RegOpenKeyExW                      58
RegOpenKeyExW_RegQueryValueExW                 55
RegOpenKeyExA_RegOpenKeyExA                    54
NtReadFile_NtReadFile                          49
RegCloseKey_RegCloseKey                        44
NtOpenKey_NtQueryValueKey                      36
NtQueryValueKey_NtQueryValueKey                34
NtQueryInformationFile_NtSetInformationFile    28
RegEnumValueA_RegEnumValueA                    27
RegQueryValueExW_RegQueryValueExW              26
RegOpenKeyExW_RegCloseKey                      23
FindFirstFileExW_FindFirstFileExW              23
LdrLoadDll_LdrGetProcedureAddress              21
NtSetInformationFile_NtQueryInformationFile    21
NtEnumerateKey_NtOpenKey                       16
RegOpenKeyExA_RegQueryValueExW                 15
LdrGetProcedureAddress_RegQueryValueExW        15
```

Fig. 8. Example of API sequence of 2-gram

Table 2에 해당하는 각 길이의 API 시퀀스를 조합한 결과, Table 3과 같이 총 11개의 조합 결과를 추출할 수 있었다. 또한 전체 API 시퀀스의 개수는 각 길이의 API 시퀀스에 해당하는 전체 API 시퀀스의 개수를 더한 값이다. 따라서 1-gram의 API 시퀀스 개수는 101개이며, 2-gram의 시퀀스 개수는 1496개이므로, 1-gram과 2-gram을 조합한 경우, 이들의 API 시퀀스 개수는 101개와 1496개를 더한 1597개이다. 이 때, 악성코드 샘플 97개는 1597개의 API 시퀀스와 이에 해당하는 호출 빈도수를 포함하고 있으며 이러한 호출 빈도수는 4.3단계

Table 3. Results of combining API sequence

Combination	Combined API sequence	The number of API sequence
2	1&2-gram	1597
	1&3-gram	6415
	1&4-gram	16022
	2&3-gram	7810
	2&4-gram	17417
	3&4-gram	22235
3	1&2&3-gram	7911
	1&2&4-gram	17518
	1&3&4-gram	22336
	2&3&4-gram	23731
4	1&2&3&4-gram	23832

에서 악성코드 유사도를 계산하기 위한 데이터로 사용된다.

4.3 악성코드 유사도 계산 결과

본 단계는 4.2단계에서 생성 및 조합한 악성코드별 API 시퀀스 및 조합에 대하여 유사도를 계산한다. 계산은 Table 3에 해당하는 모든 데이터에 대해 수행하였으며, 각 그룹별로 같은 그룹 내에 포함되어 있는 악성코드 간의 유사도를 계산하고 타 그룹과의 유사도를 계산하였다.

- **(1-gram 유사도 결과)** Table 4는 1-gram의 API 시퀀스에 대한 악성코드의 유사도를 계산한 결과이다. 동일 그룹 내에서는 악성코드들의 유사도가 1에 가까운 값이 나타났으며, 타 그룹과의 계산 결과 역시 일부 그룹을 제외하고는 0.5 이상의 높은 유사도를 갖는 것을 확인할 수 있다.
- **(2-gram 유사도 결과)** Table 5는 2-gram의 API 시퀀스에 대한 유사도 계산 결과를 보여준다. Table 4와 비교했을 때, 2-gram의 경우 타 그룹과의 유사도가 큰 폭으로 낮아진 것을 볼 수 있다. 특히, R3 그룹의 경우 Table 4에서 타 그룹과의 유사도가 0.5 이상으로 나타났으나 Table 5에서는 0.2 이하로 나타나는 것을 확인할 수 있다.
- **(3-gram 유사도 결과)** Table 6은 3-gram의 API 시퀀스를 사용하여 악성코드 그룹들 간의 유사도를 계산한 결과이다. Table 5에 비해 동일 그룹 및 타 그룹간의 유사도가 전체적으로 증가한 것을 확인할 수 있다.
- **(4-gram 유사도 결과)** Table 7은 4-gram의 API시퀀스를 사용하여 유사도를 계산한 결과이다. Table 6의 계산 결과와 마찬가지로 Table 5에 비해 동일 그룹과 타 그룹 간의 유사도가 전반적으로 증가하였다. 따라서 5-gram의 API 시퀀

Table 4. Similarity of 1-gram by groups

	R1	R2	R3	R4	R5
R1	**0.999**	-0.178	**0.617**	**0.554**	0.220
R2	-0.178	**0.996**	-0.095	-0.127	-0.234
R3	**0.617**	-0.095	**0.992**	**0.818**	**0.584**
R4	**0.554**	-0.127	**0.818**	**0.974**	**0.776**
R5	0.220	-0.234	**0.584**	**0.776**	**0.999**

Table 5. Similarity of 2-gram by groups

	R1	R2	R3	R4	R5
R1	**0.999**	-0.082	**-0.05**	**0.285**	0.203
R2	-0.082	**0.993**	-0.330	0.023	-0.026
R3	**-0.05**	-0.330	**0.978**	**0.154**	**0.092**
R4	**0.285**	0.023	**0.154**	**0.948**	**0.741**
R5	0.203	-0.026	**0.092**	**0.741**	1

Table 6. Similarity of 3-gram by groups

	R1	R2	R3	R4	R5
R1	**0.999**	-0.207	**0.009**	**0.384**	0.302
R2	-0.207	**0.991**	-0.279	0.09	0.099
R3	**0.009**	-0.279	**0.989**	**0.114**	**0.107**
R4	**0.384**	0.09	**0.114**	**0.949**	**0.765**
R5	0.302	0.099	0.107	**0.765**	1

Table 7. Similarity of 4-gram by groups

	R1	R2	R3	R4	R5
R1	**0.999**	-0.45	**-0.21**	**0.548**	0.435
R2	-0.45	**0.986**	-0.114	0.222	0.078
R3	**-0.21**	-0.114	**0.882**	**-0.03**	**-0.02**
R4	**0.548**	0.222	**-0.03**	**0.997**	**0.835**
R5	0.435	0.078	-0.019	**0.835**	**0.999**

Table 8. Similarity of API sequence combination

Combined API sequence	Same groups	Other groups
1-gram	**0.997**	0.165
2-gram	0.993	**0.025**
3-gram	0.993	0.039
4-gram	0.987	-0.004
1&2-gram	0.992	0.034
1&3-gram	0.995	0.037
1&4-gram	0.994	-0.008
2&3-gram	0.994	-0.003
2&4-gram	**0.992**	**0.008**
3&4-gram	0.989	0.034
1&2&3-gram	0.995	0.055
1&2&4-gram	0.994	0.071
1&3&4-gram	0.994	0.056
2&3&4-gram	0.992	-0.053
1&2&3&4-gram	0.992	-0.052

스를 이용하는 것은 Table 5보다 좋은 결과를 기대할 수 없으므로 본 단계에서는 더 이상의 시퀀스를 생성 및 실험하지 않는다.

- **(조합 API 시퀀스 유사도 결과)** Table 8은 각 길이의 API 시퀀스 및 조합 결과를 기반으로 악성코드 그룹 간의 평균 유사도를 계산한 결과이다. 각 길이의 시퀀스를 단독으로 사용하였을 때, 동일 그룹에 포함되는 악성코드 샘플들 간의 평균 유사도는 시퀀스 1-gram을 이용할 때 제일 높은 유사도를 갖는 것을 볼 수 있다. 그러나 타 그룹에 포함되는 악성코드 간의 평균 유사도는 2-gram의 API 시퀀스를 사용하였을 때 가장 낮은 유사도를 보이며, 조합된 API 시퀀스의 경우, 2-gram과 4-gram의 시퀀스를 조합한 경우가 동일 그룹 간 유사도와 타 그룹 간 유사도의 차이가 가장 큰 것으로 나타났다.

4.4 최적의 API 시퀀스 및 조합 도출 결과

최적의 API 시퀀스 및 조합을 도출하기 위해 4.3의 유사도 계산 결과를 바탕으로 분류에 대한 오류율을 측정하였으며, Table 9는 오류율 계산 결과를 보여준다.

모든 API 시퀀스의 조합이 대체로 낮은 분류 오류율을 보이지만, 2-gram의 API 시퀀스를 단독으로 이용한 경우가 가장 적은 오류율을 갖는 것을 확인할 수 있다. 또한 Fig.9.에서도 다른 API 시퀀스 조합의 오류율에 비해 2-gram의 API 시퀀스 오류율은 확연하게 낮은 것을 볼 수 있다.

따라서 2-gram의 API 시퀀스를 사용하는 것이 가장 높은 분류 정확성을 갖는 것으로 판단되며, 악

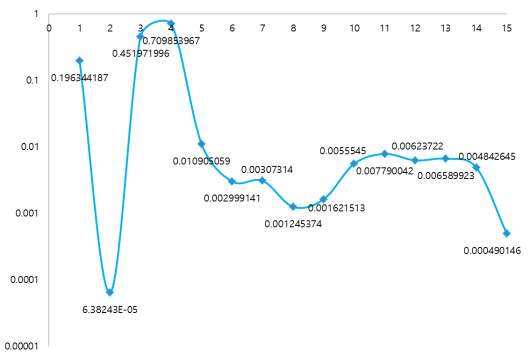

Fig. 9. Classification error rate by combined API sequence

Table 9. Classification error rate of API sequence combination

	Combined API sequence	Error rate
1	1-gram	0.196
2	2-gram	**6.38E-05**
3	3-gram	0.452
4	4-gram	0.71
5	1&2-gram	0.011
6	1&3-gram	0.0031
7	1&4-gram	0.003
8	2&3-gram	0.0012
9	2&4-gram	0.002
10	3&4-gram	0.006
11	1&2&3-gram	0.0078
12	1&2&4-gram	0.006
13	1&3&4-gram	0.0066
14	2&3&4-gram	0.0048
15	1&2&3&4-gram	0.00049

성코드 API 함수로부터 추출할 수 있는 최적의 API 시퀀스 길이 및 조합은 2-gram의 API 시퀀스인 것으로 도출되었다.

V. 결 론

본 논문에서는 악성코드 분류의 정확성을 향상시키기 위한 최적의 API 시퀀스 길이 및 조합을 찾는 방법을 제안하고, 동시에 최적의 값을 도출하였다. 제안 방법은 악성코드를 동적 분석하여 다양한 길이의 API 시퀀스를 생성 및 조합하고, 이를 바탕으로 유사도 및 분류 오류율을 구하여 최적의 API 시퀀스 길이 및 조합을 찾는다.

따라서 제안한 방법을 평가하고 결과를 도출하기 위해 수집한 악성코드 샘플에 대하여 실험을 진행하였고, 그 결과 2-gram의 API 시퀀스를 이용하여 악성코드를 분류했을 때 오류율이 가장 낮은 수치를 보임으로써 이를 최적의 특징으로 판단할 수 있었다. 그러나 비교적 적은 악성코드 샘플에 대상으로 실험이 진행되었기 때문에 향후 대량의 악성코드 샘플에 대한 실험이 이루어져야 하며, 다양한 유사도 계산 기법을 적용하여 실험 결과에 대한 신뢰성을 높이는 연구를 지속하여야 한다.

References

[1] Chaetae Im, JooHyung Oh, and Hyuncheol Jeong, "Study of Technical Trends and Analysis Method of Recent Malware," Journal of The Korea Information Science Society, 28(11), pp. 117-126, Nov. 2010

[2] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat, "Malware analysis and classification a survey," Journal of Information Security, vol. 5, no. 2, pp. 56-64, Apr. 2014

[3] API, Wikipedia, available at http://ko.wikipedia.org/wiki/API [Accessed: 1th September 2014]

[4] Changwook Park, Hyunji Chung, Kwangseok Seo and Sangjin Lee "Research on the Classification Model of Similarity Malware using Fuzzy Hash," Journal of The Korea Institute of Information Security & Cryptology, 22(6), pp. 132 5-1336, Dec. 2012

[5] OllyDbg, available at http://www.dllydbg.de/ [Accessed: 1th September 2014]

[6] Immunity Debugger, available at http://www.immunityinc.com/ [Accessed: 1th September 2014]

[7] IDA Pro, available at https://www.hex-rays.com [Accessed: 1th September 2014]

[8] R.Tian, L.M.Batten, and S.C.Versteeg, "Function length as a tool for malware classification," Proceedings of the 3rd International Conference on Malware 2008, pp. 69-76, Oct. 2008.

[9] Ronghua Tian, Lynn Batten, Rafiqul Islam, and Steve Versteeg, "An automated classification system based on the strings of trojan and virus families," 4th International Conference on Malicious and Unwanted Software 2009, pp. 23-30, Oct. 2009.

[10] Qi-Guang Miao, Yun-Wang, and Ying-Cao, "APICapture - a tool for monitoring the behavior of malware," 2010 3rd International Conference on Advanced Computer Theory and Engineering, pp. 390-394, Aug. 2010.

[11] M.Biley, J.Oberheid, J.Andersen, and Z.Morley Mao, F.Jahanian, and J.Nazario, "Automated classification and analysis of Internet malware," Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection, LNCS 4637, pp. 178-197, 2007.

[12] U.Bayer, P.M.Comparetti, C.Hlau schek, and C.Kruegel, (2009) "Scalable, behavior-based malware clustering," Proceedings of the 16th Annual Network and Distributed System Security Symposium 2009, Feb. 2009.

[13] Portable Executable, Wikipedia, available at http://ko.wikipedia.org/wiki/PE_%ED%8F%AC%EB%A7%B7 [Accessed: 1th September 2014]

[14] Kyoung-Soo Han, In-Kyoung Kim, and Eul-Gyu Im, "Malware Family Classification Method using API Sequential Characteristic," Journal of Security Engineering, 8(2), pp. 319-335, Apr. 2011

[15] Kazuki Iwamoto and Katsumi Wasaki, "Malware classification based on extracted API Sequences using static analysis," 12 Proceedings of the Asian Internet Engineering Conference, pp. 31-38, Nov. 2012.

[16] Vinod P, H.Jain, Y.K.Golecha, M.S. Gaur, and V.Laxmi, "MEDUSA: Metamorphic malware dynamic analysis using signature from API," Proceedings of the 3rd International Conference on Security of Information and Networks, pp. 263-269, Sep. 2010.

[17] Younghee Park, Douglas Reeves, Vik ram Mulukutla, and Balaji Sunda ravel, "Fast

malware classification by automated behavioral graph matching," Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research, Apr. 2010.
〔18〕 N-gram, Wikipedia, available at http://en.wikipedia.org/wiki/N-gram 〔Accessed: 1th September 2014〕
〔19〕 I.Jolliffe, Principal component analysis, 2nd Ed., Springer, 488 p, 2002
〔20〕 Science&Technology Security Center, available at http://www.sntsec.or.kr/ 〔Accessed: 1th September 2014〕
〔21〕 Virustotal homepage, available at https://www.virustotal.com/ko/ 〔Accessed: 1th September 2014〕
〔22〕 Antivirus and Threat Report: January 2014, available at http://www.opswat.com/about/media/reports/antivirus-january-2014 〔Accessed: 1th September 2014〕
〔23〕 Cuckoosandbox homepage, available at http://www.cuckoosandbox.org/ 〔Accessed: 1th September 2014〕

〈저자소개〉

최 지 연 (Ji-yeon Choi) 학생회원
2013년 2월: 한남대학교 경영정보학과 졸업
2013년 3월~현재: 과학기술연합대학원대학교 그리드 및 슈퍼컴퓨팅 석사과정
〈관심분야〉 정보보호, 악성코드 분석, 네트워크 보안

김 희 석 (HeeSeok Kim) 정회원
2006년 2월: 연세대학교 수학과 졸업
2008년 2월: 고려대학교 정보보호대학원 석사
2011년 8월: 고려대학교 정보보호대학원 박사
2011년 9월~2012년 12월: Bristol University 박사 후 연구원
2013년 2월~현재: 한국과학기술정보연구원 과학기술정보보호실 선임연구원
〈관심분야〉 부채널 공격, 암호시스템 안전성 분석 및 고속구현, 보안관제, 네트워크 보안

김 규 일 (Kyu-il Kim) 정회원
2005년 2월: 성균관대학교 컴퓨터공학과 석사
2010년 2월: 성균관대학교 컴퓨터공학과 박사
2010년 6월~현재: 한국과학기술정보연구원 과학기술정보보호실 선임연구원
〈관심분야〉 보안관제, 침해사고대응, 악성코드 분석

박 학 수 (Hark-soo Park) 정회원
1989년 2월: 한남대학교 전자계산학과 졸업
1991년 2월: 한남대학교 컴퓨터공학과 석사
2003년 2월: 한남대학교 컴퓨터공학 박사
1991년 3월~현재: 한국과학기술정보연구원 책임연구원
〈관심분야〉 정보보호, 보안관제, 침해사고대응

송 중 석 (Jung-suk Song) 정회원
2003년 2월: 한국항공대학교 통신정보공학 졸업
2005년 2월: 한국항공대학교 정보공학 석사
2009년 3월: 교토대학교(일본) 지능정보학 박사
2009년 4월~2010년 9월: 일본정보통신연구원 정보통신 보안 연구소 전문연구원
2010년 10월~2011년 9월: 일본정보통신연구원 네트워크 보안 연구소 선임연구원
2011년 10월~현재: 한국과학기술정보연구원 과학기술정보보호실 선임연구원
2012년 9월~현재: 과학기술연합대학원대학교 그리드 및 슈퍼컴퓨팅 겸임교수
〈관심분야〉 보안관제, 침해사고대응, 악성코드 분석, 네트워크 보안