情報保護學會論文誌
第18卷 第6(A)號, 2008. 12

Native API 빈도 기반의 퍼지 군집화를 이용한 악성코드 재그룹화 기법연구*

권 오 철†, 배 성 재, 조 재 익, 문 종 섭‡
고려대학교, 정보경영공학전문대학원

Malicious Codes Re-grouping Methods using Fuzzy Clustering based on Native API Frequency*

O-chul Kwon†, Seong-jae Bae, Jae-ik Cho, Jung-sub Moon‡
Graduate School of Information Management and Security, Korea University

요 약

Native API(Application Programming Interfaces)는 관리자 권한에서 수행되는 system call의 일종으로 관리자 권한을 획득하여 공격하는 다양한 종류의 악성코드를 탐지하는데 사용된다. 이에 따라 Native API의 특징을 기반으로한 탐지방법들이 제안되고 있으며 다수의 탐지방법이 교사학습(supervised learning) 방법의 기계학습(machine learning)을 사용하고 있다. 하지만 Anti-Virus 업체의 분류기준은 Native API의 특징점을 반영하지 않았기 때문에 교사학습을 이용한 탐지에 적합한 학습 집합을 제공하지 못한다. 따라서 Native API를 이용한 탐지에 적합한 분류기준에 대한 연구가 필요하다. 본 논문에서는 정량적으로 악성코드를 분류하기 위해 Native API를 기준으로 악성코드를 퍼지 군집화하여 재그룹화하는 방법을 제시한다. 제시하는 재그룹화 방법의 적합성은 기계학습을 이용한 탐지성능의 차이를 기존 분류방법을 결과와 비교하여 검증한다.

ABSTRACT

The Native API is a system call which can only be accessed with the authentication of the administrator. It can be used to detect a variety of malicious codes which can only be executed with the administrator's authority. Therefore, much research is being done on detection methods using the characteristics of the Native API. Most of these researches are being done by using supervised learning methods of machine learning. However, the classification standards of Anti-Virus companies do not reflect the characteristics of the Native API. As a result the population data used in the supervised learning methods are not accurate. Therefore, more research is needed on the topic of classification standards using the Native API for detection. This paper proposes a method for re-grouping malicious codes using fuzzy clustering methods with the Native API standard. The accuracy of the proposed re-grouping method uses machine learning to compare detection rates with previous classifying methods for evaluation.

Keywords : Native API, Fuzzy Clustering, Malicious Codes Re-grouping, Machine Learning, Data Mining

접수일: 2008년 9월 11일; 수정일: 2008년 10월 15일;
채택일: 2008년 12월 08일
* 이 연구에 참여한 연구자(의 일부)는 '2단계BK21사업'의 지원비를 받았음
† 주저자, kwonoch@korea.ac.kr
‡ 교신저자, jsmoon@korea.ac.kr

Ⅰ. 서 론

국가사이버안전센터의 발표 자료에 따르면 2007년도

에 발생한 해킹사고 피해운영체제 중 Windows계열이 88%로 가장 많이 발생한 것으로 나타났다. 반면 리눅스 운영체제는 전체의 10%로 2006년 대비 3% 감소한 것으로 나타났다[1]. 이는 대다수의 피해사례가 윈도우를 사용하는 개인 PC를 대상으로 하는 침해사고가 많았기 때문이다. 또한 침해사고의 81%는 웜, 바이러스와 같은 악성코드의 감염으로 인해 발생하였으므로 이에 대한 예방이 필요하다.

이러한 피해를 예방하기 위하여 악성코드 탐지 및 해킹방어에 대한 연구가 활발하게 이루어지고 있다. 공격탐지를 위해 많은 종류의 공격 속성과 다양한 탐지 알고리즘이 사용되고 있으며, 그 중 system call을 이용한 탐지방법에 대한 연구가 활발하게 진행되고 있다[2][3][9][11][22].

Windows의 운영체제의 경우, system call을 분석하는데 있어서 Native API가 사용된다[10][12]. Native API는 윈도우즈 커널 모드에서 작동하는 API(Application Programming Interfaces)를 의미한다[17][18]. Native API를 기반으로 생산된 특징점들을 이용하는 비정상행위 탐지방법은 의사결정트리(Decision Tree), Hidden Marcov Model (HMM), Support Vector Machine (SVM)등 다양한 패턴인식 기법들을 응용하여 연구되고 있다[8][16]. 이러한 탐지방법은 기존 Anti-Virus 업체들이 지정한 공격분류방법을 적용하여 탐지를 하고 있으나 업체들이 서로 다른 악성 코드 분류 기준을 사용하고 있고 Native API의 특징을 잘 반영하고 있지 않아 악성코드를 분류하는데 적합하지 않은 문제점이 있다.

본 논문에서는 Native API의 호출 빈도를 특징점으로 하는 악성코드의 군집화(Clustering)방법을 제시한다. 군집화 방법은 집단의 소속정도를 부여하는 퍼지 군집화 방식을 사용하고, 주요 퍼지 군집화에 사용되는 여러 알고리즘을 비교 분석하여 가장 효과적인 알고리즘을 제시한다. 가장 효과적인 군집화 방법에 따라 기존의 악성코드를 재그룹화 하여 새로운 레이블(Label)을 부여한다. 또한 군집화에 따른 재그룹화가 기존 Anti-Virus 업체의 분류에 비해 탐지 성능을 향상시키는 역할을 하게 됨을 검증한다.

본 논문의 구성은 2장에서 관련연구로서 기존의 악성코드 분류 방법과 문제점, Native API 수집방법 및 퍼지 군집화에 대하여 살펴보고, 3장에서는 악성코드를 군집화하여 재그룹화하는 방법을 제안하고 유효성을 검증하는 방법을 설명한다. 4장에서는 실험의 세부과정을 설명하고 실험결과를 분석한다. 마지막으로 5장에서 결론을 맺으며 논문을 마무리한다.

Ⅱ. 관련 연구

2.1 기존의 악성코드 분류 방법들

Anti-Virus 업체들은 악성코드를 감염대상, 자기복제 방법, 전파방법, 사용하는 취약점, Payload의 성격, 변종여부 등에 따라 분류한다. 예를 들어, Kaspersky lab.의 악성코드 분류방법은 크게 Network worms, Classic viruses, Trojan programs로 나누며 다시 16개의 소분류로 나누는 방식이다[19]. 하지만 업체별 구분기준은 표준화되지 않아 업체별로 상이한 분류방법을 사용한다. 이에 따라 동일한 악성코드를 업체별로 서로 다른 악성코드로 분류하는 경우가 발생한다. M. Bailey et al.은 각 업체별 분류가 악성코드별로 20~49%정도 상이하게 나타난다고 밝혔다[6].

[표 1]은 Storm worm(CME-711)에 대한 각 Anti-Virus 업체들의 분류를 나타낸다[25]. Storm worm은 다량의 스팸메일을 발송하고 메일수신자가 원격지 링크를 통해 악성코드를 다운로드하도록 하고 P2P 봇넷을 형성하도록 하는 악성코드이다.

[표 1] Storm worm에 대한 Anti-virus 업체별 분류

업체명	분류
ClamAV	Trojan.Downloader-647
F-Secure	Small.DAM
Kaspersky	Trojan-Downloader.Win32. Small.dam
McAfee	W32/Nuwar@MM
Microsoft	Win32/Nuwar.N@MM!CME-711
Sophos	Troj/DwnLdr-FYD
Symantec	Trojan.Peacomm
Trend Micro	TROJ_SMALL.EDW

동일한 악성코드에 대해 업체별로 Downloader 또는 Mass-Mailer로 다르게 분류한 것을 알 수 있다. 이러한 분류의 혼란을 최소화하기 위하여 글로벌 악성코드 정보공유 프로젝트인 CME(Common Malware Enumeration)가 동일 악성코드의 상이한 분류를 통일시키는 작업을

하고 있으나, 치명적인 악성코드에 한하여 제한적으로 수행하고 있으므로 전반적인 악성코드 분류 표준화에는 큰 영향을 주지 못하고 있다[25]. 이러한 상이한 악성코드 분류는 데이터 마이닝을 통한 위협 탐지의 사전 학습단계에서 부정확한 정보를 제공하게 되기 때문에 평가단계에서 잘못된 결론을 내릴 수가 있다[4][7]. 따라서 데이터를 특성에 맞게 군집화하고 특징점에 맞는 레이블(label)을 지정하는 분류방법의 연구가 필요하다.

2.2 Native API Call

2.2.1 개요

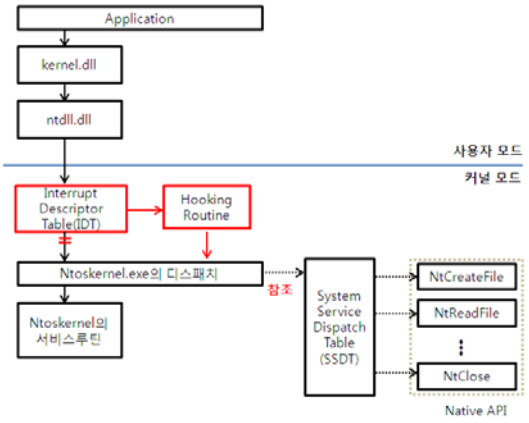
API(Application Programming Interfaces)는 동적 라이브러리(dll)에 포함된 프로그래밍 가능한 함수들이다. 이는 사용자 모드와 커널 모드로 작동한다. Native API는 윈도우즈 커널 모드에서 작동하는 API를 의미한다[17][18]. 커널 모드에서는 사용자 레벨의 응용프로그램이 접근하지 못하도록 하여 중요한 시스템 레벨의 프로그램을 보호한다. 이러한 원리로 일반 사용자의 시스템의 오용이나 남용으로 인한 시스템 저해를 막음으로써 전반적인 시스템의 안정성을 보장한다[15].

2.2.2 Native API 수집 방법

시스템에서 발생되는 Native API에 대한 정보를 수집하기 위해서는 커널을 가로채는(hooking) 기법이 필요하다. 이 기법에는 System Service Dispatch Table (SSDT)를 이용하는 기법과 Interrupt Descriptor Table(IDT)을 이용하는 기법, 그리고 디바이스 드라이버 오브젝트의 Major Function hooking 기법이 있다[5].

SSDT를 이용하는 커널 후킹 기법은 SSDT를 조작하여 특정한 Native API 정보를 얻는 방법이다. Windows는 Win32. POSIX, OS/2 서브시스템을 지원한다. 이 서브시스템이 제공하는 서비스의 주소는 SSDT라고 하는 커널 구조체에 의해 관리 된다. 이 테이블은 system call 번호 순서대로 해당 함수의 메모리 주소를 가지고 있다. 즉, 해당 함수의 메모리 주소를 변조하여 Native API 정보를 얻는 기법이다.

IDT 커널 후킹 기법은 SSDT 기법과 마찬가지로 IDT를 변조하여 Native API 정보를 얻는 기법이다. IDT는 인터럽트를 처리하는데 사용되는 테이블이다. 인터럽트는 소프트웨어로부터 또는 하드웨어로부터 발생한다. IDT에는 사용자가 키보드입력을 하였거나 페이지 폴트가 발생했을 때, 또는 SSDT의 시스템 서비스 함수가 호출 되는 경우에 각각 어떤 처리가 이뤄져야 하는지에 대한 정보가 저장되어 있다. IDT 커널을 가로채면 SSDT의 커널 함수가 호출되기 전에 먼저 후킹 함수가 호출된다.



[그림 1] IDT 커널 후킹 구조도

마지막으로 디바이스 드라이버 오브젝트의 Major Function 후킹 기법은 디바이스 드라이버의 Major Function 테이블을 변조하는 것이다. 디바이스 드라이버는 로드되면서 여러 형태의 I/O Request Packet (IRP) 를 처리하기 위한 함수 리스트를 초기화한다. 각 IRP 의 종류에 따라서 IRP를 처리하는 함수를 Major Function 이라고 하고 이 Major Function 들의 주소를 관리하는 것을 Major Function 테이블이라고 한다. 즉 커널 레벨에서 동작하는 디바이스 드라이버의 Major Function Table 을 조작하는 방법이다.

2.3 퍼지 군집화(Fuzzy Clustering)

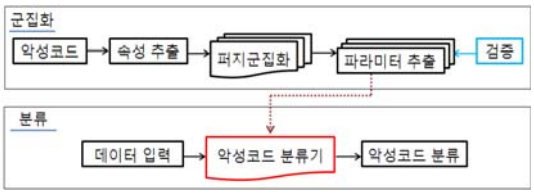
군집화는 주어진 데이터의 집합을 유사한 성질(similarity)을 가지는 몇 개의 그룹(Cluster, Partition)으로 나누는 방법을 뜻한다. 유사성은 흔히 데이터간의 거리(distance norm)를 기준으로 한다. 클러스터 분할법은 크게 하드(Hard)와 퍼지(Fuzzy)로 구분된다. 하드

군집화는 고전적인 그룹이론에 근거한 이론으로 각 데이터는 어느 한 곳의 클러스터에 속해야한다. 퍼지 군집화는 목적 데이터가 여러 클러스터에 비율적으로 속하게 하는 이론으로 그룹의 경계가 명확하지 않는 경우가 더 많은 실생활에 적합하다고 볼 수 있다[24]. 악성코드는 다양한 공격방법이 혼합되어있는 구조를 많이 가지므로 소속 정도를 부여하는 퍼지 군집화 방식의 분류가 적합하다.

Ⅲ. 제안하는 방법

3.1 개요

악성코드들을 군집화하고 분류하는 방법은 그림 2와 같다.



[그림 2] 군집화 및 분류 순서도

군집화단계는 악성코드의 Native API의 발생빈도를 바탕으로 주성분의 API를 속성(feature)으로 추출하여 이를 퍼지 군집화하고, 군집화 결과값을 파라미터(parameter)로 추출하여 가장 적합한 퍼지 군집화 결과를 검증하는 단계이다.

분류단계는 최적화된 파라미터를 기반으로 악성코드와 정상프로그램을 분류하는 단계이다. 악성코드 분류기(Classifier)의 학습단계에는 군집화단계의 파라미터가 사용되며 이를 통해 생성된 분류모델을 바탕으로 입력된 데이터를 분류하고 악성코드일 경우 세부 분류까지 판별한다.

3.2 군집화

3.2.1 악성코드의 속성 추출

악성코드 실행에 따른 Native API의 발생 빈도는 IDT 커널 후킹 기법을 이용하여 수집하게 된다. 수집된 데이터는 군집화에 적합한 속성으로 추출되게 되며 속성 추출 절차는 다음과 같다.

먼저 군집화에 사용되는 j번째 악성코드를 X_j라 하자. 사용된 전체 악성코드는 N개이며 $x=(x_1, x_2, \cdots, x_N)$으로 표기된다. 전체 Native API의 개수는 n개이며 이 중 주성분 요소 s개를 선정하게 된다. 따라서 각 x_j는 s개의 속성으로 이루어지게 되며 $x_j=(x_{j1}, x_{j2}, \cdots, x_{js})$으로 표기된다.

T_j : 악성코드 x_j가 수행될 때 발생한 Native API의 총 개수
Q_{jk} : j번째 악성코드의 k번째 Native API의 발생 개수
P_{jk} : j번째 악성코드의 k번째 Native API의 발생 빈도
R_k : 모든 코드에서, k번째 Native API의 평균빈도

$$P_{jk} = Q_{jk}/T_{j,} \quad 1 \le j \le N, \quad 1 \le k \le n \tag{1}$$

$$R_k = \frac{1}{N}\sum_{j=1}^{N} P_{jk} \tag{2}$$

따라서 모든 악성코드는 (1)을 속성으로 가지는 vector로 표시된다. 즉, $x_j=(P_{j1}, P_{j2}, \cdots, P_{js})$이다.

3.2.2 악성코드 퍼지 군집화

3.2.2.1 Fuzzy C-means(FCM) 알고리즘

Fuzzy C-means(FCM)는 가장 널리 사용되어지고 있는 알고리즘으로 구형의 클러스터에 적합한 알고리즘이 있다. FCM 알고리즘은 각 데이터와 각 클러스터 중심과의 거리를 고려한 유사도 측정에 기초한 목적 함수의 최적화 방식을 사용하며, 목적함수는 다음과 같다[24].

$$J(X;U,V)=\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^m\|x_j-v_i\|_A^2 \tag{3}$$

여기서 N은 데이터의 개수, c는 클러스터의 개수이고, 주어진 입력 데이터의 집합 $X=\{x_{1,}x_2,\cdots x_N\}$에 대한 퍼지 c 분할을 $c\times N$ 행렬 U로 나타낼 때 μ_{ij} 는 데이터 x_j 가 클러스터 i에 속하는 소속 정도를 나타낸다. 또한 $\|\cdot\|$은 유클리디안 놈(Euclidean norm)이고 v_i는 i번째 클러스터의 중심을 나타내며, $m\in\{1,\infty\}$은 퍼지 정도를 나타내는 매개변수이다. 본 논문에서는 m=2를 사용한다. FCM 알고리즘의 수행절차는 다음과 같다.

단계 1 : 클러스터의 개수 $c(1 < c < N)$ 값, 가중지수 $m(1 \le m \le \infty)$값, 종료 조건($\epsilon > 0$)을 결정한다.

단계 2 : 다음 조건을 만족하는 분할 행렬 $U^{(0)}$를 초기화 한다.

$$\sum_{i=1}^{c} \mu_{ij} = 1,\ 0 < \sum_{j=1}^{N} < N, \quad u_{ij} \in [0,1],\ 1 \le i \le c,\ 1 \le j \le N \tag{4}$$

단계 3 : 각 클러스터의 중심값을 계산한다.(l=0,1,2..)

$$vi^{(l)} = \frac{\sum_{j=1}^{N} (\mu_{ij}^{(l-1)})^m x_j}{\sum_{j=1}^{N} (\mu_{ij}^{(l-1)})^m},\ 1 \le j \le N \tag{5}$$

단계 4 : 거리를 계산한다. A는 놈-유도(norm- inducing) 행렬이다.

$$D_{ijA}^2 = (x_j - v_i)^T A (x_j - v_i), \quad 1 \le i \le c,\ 1 \le j \le N \tag{6}$$

단계 5 : 분할 행렬 U를 갱신한다.

$$\mu_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{c} (D_{ijA}/D_{kjA})^{2/(m-1)}} \tag{7}$$

단계 6 : 다음 종료 조건이 만족될 때까지 단계 3~5를 반복한다. l은 반복 횟수를 의미한다.

$$\| U^{(l)} - U^{(l-1)} \| < \epsilon \tag{8}$$

FCM의 수행절차는 Gustafson-Kessel, Gath-Geva 알고리즘에서도 일정부분 사용되는 기본적인 퍼지 군집화 수행방법이다.

3.2.2.2 Gustafson-Kessel(GK) 알고리즘

Gustafson-Kessel(GK) 알고리즘은 데이터 집합에서 서로 다른 기하학적인 형태의 클러스터를 검출하기 위하여 유클리디언 거리측정과 함께 공분산 행렬로부터 유도된 적응적인 거리 측정(Adaptive distance norm)을 적용한 표준 FCM의 확장 형태이다[23]. 이를 통해 타원형의 클러스터 형성이 가능해짐에 따라 FCM보다 융통성 있는 군집화를 가능하게 한다. GK 알고리즘의 목적함수는 다음과 같이 정의한다.

$$J(X; U, V, A) = \sum_{i=1}^{c} \sum_{j=1}^{N} (\mu_{ij})^m D_{ijA_i}^2 \tag{9}$$

$$D_{ijA}^2 = (x_j - v_i)^T A_i (x_j - v_i), \quad 1 \le i \le c,\ 1 \le j \le N \tag{10}$$

거리 놈(norm) D_{ijA}^2는 데이터 집합에서 서로 다른 기하학적인 형태의 클러스터들에 대하여 계산한 결과이다. 여기서 행렬 A_i는 c-means 함수에서 최적화 변수로 이용된다. A_i를 유도된 행렬 $A_i = (A_1, A_2, \cdots, A_c)$의 c개의 조합으로 기술한다. A_i는 공분산 행렬 F_i에 대한 놈-유도(norm-inducing) 행렬로써 다음과 같이 계산된다.

$$A_i = [\rho_i \det(F_i)]^{1/n} F_i^{-1} \tag{11}$$

F_i는 i번째 클러스터의 퍼지 공분산 행렬로 다음 식과 같이 정의 된다.

$$F_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m (x_j - v_i)(x_j - v_i)^T}{\sum_{j=1}^{N} (\mu_{ij})^m} \tag{12}$$

여기서 공분산은 U에서 소속정도에 의해 가중처리된다.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} (D_{ijA_i}(x_j, v_i)/D_{kj}(x_j, v_k))^{2/(m-1)}}, \quad (1 \le i \le c,\ 1 \le j \le N) \tag{13}$$

GK 알고리즘은 FCM의 수행철차와 유사하며, 3단계와 4단계 사이에 공분산 F_i와 A_i를 구하는 절차가 추가된다.

3.2.2.3 Gath-Geva(GG) 알고리즘

Gath-Geva(GG) 알고리즘은 데이터의 밀도뿐 아니라 넓이도 고려하여, 밀도는 좀 작지만 넓게 분포한 클러스터도 군집화가 가능하도록 GK 알고리즘을 개선한 방법이다. 또한 군집화가 어려운 서로 중첩된 클러스터도 찾아 낼 수 있으며, 거리 계산식에 지수함수가 포함되어 거리에 따른 소속정도의 차이가 커지기 때문에 노이즈에 강한 장점을 가진다[13].

목적함수는 GK와 동일하며 공분산 행렬을 F_i대신 F_{wi}를 사용하고 가중치 w=2를 사용한다는 점과 사전확률(priori probability) α_i를 사용한다는 차이점이 있다. 사전확률은 거리계산 공식에서 사용되며 지수계산

으로 인해 초기값에 민감한 결과를 출력한다. 본 논문에서는 GK알고리즘의 결과를 사전확률로 사용하였다.

$$D_{ij}(x_j, v_i) = \frac{\sqrt{\det(F_{wi})}}{\alpha_i} exp(\frac{1}{2}(x_j - v_i^{(l)})^T F_{wi}^{-1}(x_j - v_i^{(l)})) \quad (14)$$

$$F_{wi} = \frac{\sum_{j=1}^{N}(\mu_{ij})^w (x_j - v_i)(x_j - v_i)^T}{\sum_{j=1}^{N}(\mu_{ij})^w} \quad (15)$$

$$\alpha_i = \frac{1}{N}\sum_{j=1}^{N}\mu_{ij} \quad (16)$$

3.2.3 파라미터 추출

퍼지 군집화를 통해 각 클러스터의 대푯값과 데이터의 클러스터 소속정도(U)를 파라미터로 추출한다. 클러스터의 대푯값은 대체로 클러스터의 특성을 나타내는 중심값(V)이 된다. 데이터의 클러스터 소속정도는 군집화의 적절성을 검증하는 지표로 사용되며, 소속정도가 높은 클러스터로 데이터를 레이블하는데 사용된다.

3.2.4 군집화 유효성 검증

퍼지 군집화는 사전에 클러스터의 개수를 지정하고 수행되게 되므로 지정한 클러스터의 개수가 데이터에 적합한지를 검증해야한다. 가장 많이 사용하는 검증 방법으로 Partition Coefficient(PC)와 Partition Entropy coefficient(PE)가 있다[24]. PC는 클러스터 내의 데이터의 상관관계를 보는 것으로서 얼마나 조밀하게 모여 있는지를 나타낸다. PE는 PC와 유사하나 클러스터 내의 데이터가 얼마나 확산되어있는지를 나타낸다. PC는 1에 가까울수록 PE는 0에 가까울수록 분류가 잘 되었음을 증명한다.

$$PC(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^2 \quad (17)$$

$$PE(c) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}\mu_{ij}\log(\mu_{ij}) \quad (18)$$

또한 클러스터별 PC와 PE의 값이 확연한 변화를 보이는 지점(knee of increase or decrease)의 클러스터의 개수가 데이터에 적합한 개수로 판단되게 된다. Native API에 적합한 클러스터의 개수를 사전에 확인할 수 없으므로 PC와 PE를 이용하여 적합한 클러스터의 개수를 검증한다.

3.3 분류

3.3.1 데이터 입력

입력되는 데이터는 악성코드와 일반프로그램의 Native API 빈도가 레이블이 없이 합쳐진 테이블이 된다. 이렇게 입력된 데이터는 악성코드와 일반프로그램으로 분류(Classification)되게 되며 분류의 정확도가 탐지율이 된다.

3.3.2 데이터 마이닝을 이용한 악성코드 분류기

악성코드 분류기(Classifier)는 크게 학습단계와 평가단계로 이루어진다. 학습단계(Training Step)는 최적화된 군집화 결과에 따른 레이블을 이용하여 분류모델을 생성하는 교사학습(supervised learning) 단계이다. 평가단계(Test Step)는 분류모델을 바탕으로 레이블이 없는 입력 데이터를 분류하는 단계이며 사용하는 알고리즘과 학습단계의 정확도가 결과에 영향을 미친다.

사용될 분류기는 데이터 마이닝 기법에서 많이 사용되는 의사결정나무(Decision Tree) 분류, 베이지안(Bayesian) 분류, KNN(K-Nearest Neighbor) 분류 알고리즘이다.

3.3.2.1 의사결정나무(Decision Tree) 분류

의사결정나무는 노드와 링크로 구성된 일반적인 구조를 가지며, 각각의 노드는 데이터들이 표현된 특징(feature)중의 하나가 되며, 링크는 그 특징이 가질 수 있는 값이 된다. 의사결정나무에 의한 규칙 생성방법은 데이터로부터 그 데이터들이 암시적으로 포함하고 있는 개념을 추출하여 의사결정나무의 형태로 일반화하고 이를 이용하여 새로운 데이터를 분류하게 된다[26].

3.3.2.2 베이지안(Bayesian) 분류

베이지안 분류는 베이즈의 정리를 이용한 의사결정

방식이다. 베이즈 정리는 사전확률(prior probability)과 사후확률(posterior probability)간의 관계를 조건부 확률로 정리한 내용으로 어떤 실험결과에서 얻은 정보를 이용하여 다음 사건의 처음 확률을 개선시키는데 사용할 수 있으며, 처음 확률을 사전확률이라하고 개선된 확률을 사후확률이라고 한다. 베이지안 분류는 이 정리를 이용하여 특징값이 주어진 상태에서 어느 범주로 분류될지에 대한 확률을 계산하는 분류방식이다[27].

3.3.2.3 KNN(K-Nearest Neighbor) 분류

KNN 분류 방법은 분석 데이터 셋 자체를 분류 모델로 가지기 때문에 별도의 분류 모델을 생성하지 않는다. 다만 분류해야 할 데이터와 가장 유사한 K개의 기존 데이터를 대상으로 분류 가능성이 가장 높은 범주로 분류한다. 분류시 유사한 정도를 계산하기 위해 유클리드 거리를 이용한다. 즉 n차원 공간에서 자신과 가장 가깝게 위치한 K개의 다른 표본들의 클래스 중에서 가장 많은 것으로 분류되는 것이 KNN 알고리즘이다[28].

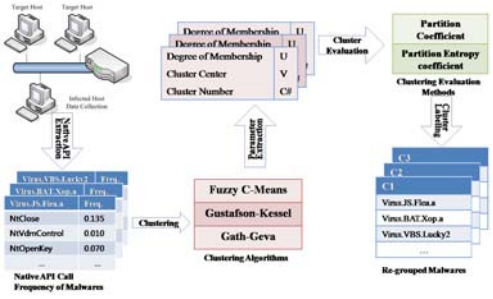
3.3.3 악성코드 분류

분류기의 분류 결과에 따라 정상 프로그램과 악성코드의 분류하게 되며, 악성코드의 세부그룹까지 구분하게 된다. 세부그룹까지 구분하는 가장 큰 이유는 악성코드의 빠른 분석에 도움이 되기 위함이다. 따라서 탐지의 정확도는 세부그룹 분류 정확도를 포함하게 된다.

Ⅳ. 실험 결과

4.1 실험 방법

4.1.1 군집화 단계



[그림 3] 군집화 순서도

그림 3은 악성코드의 군집화 순서도를 나타낸다.

악성코드의 Native API를 수집하기 위하여 Window XP SP2가 설치된 PC 3대로 구성된 소규모 네트워크를 구성하였다. 이 중 1대의 PC가 감염 PC가 되며 2대의 PC는 악성코드 전파 대상 PC가 된다. 감염 PC에서 총 242개의 악성코드를 실행하였으며 실험에 사용된 악성코드는 Offensive Computing에서 제공하는 악성코드를 사용하였다[20]. 악성코드의 선정은 대조군인 Kaspersky lab.의 분류기준이 적용된 악성코드 내에서 이루어졌다.

Native API 수집은 IDT 후킹 방법을 사용하여 악성코드가 실행되기 전 부터 실행 후 10~30분간 전체 프로세스를 대상으로 수행되었다. 여기서 악성코드의 수행으로 생성된 Native API를 악성코드의 PID를 기준으로 추출하였다. 242개의 악성코드에서 사용한 Native API의 종류는 전체 Native API 949개 중 380개였으며 이 중 발생 빈도(R_j)가 높은 13개의 API를 선별하여 퍼지 군집화를 수행하였다. 즉, 각 코드별 속성 개수(s)는 13이다. 따라서 P는 242×13인 행렬이 된다.

퍼지 군집화에는 Fuzzy C-means(FCM), Gustafson-Kessel(GK), Gath-Geva(GG) 알고리즘을 사용하였으며 matlab 7.2로 프로그래밍하였다. 악성코드의 분류 초기값으로 $N=242$, $s=13$, $\epsilon=0.0001$로 설정된다. 데이터에 적합한 클러스터의 개수는 사전에 알 수 없으므로 클러스터 개수(c)를 2~6개로 바꿔가며 군집화를 수행한다. 입력 데이터를 정리하면 [표 2]와 같다.

[표 2] 입력 데이터

수식	행렬 차원	설 명
P_{jk}	$N \times s$	j번째 악성코드의 k번째 Native API의 발생 빈도 행렬
c	상수	군집화에 사용되는 클러스터의 개수 (2~6)
N	상수	입력된 악성코드의 개수 (242)
s	상수	주요 Native API Call의 개수 (13)
ϵ	상수	목적함수 종료조건 (Threshold, 0.0001)

군집화 결과는 [표 3]과 같이 나오게 된다.

각 군집화의 결과 중 분할행렬 U가 PC와 PE에서 검증데이터로 사용된다. PC와 PE 검증을 통해 가장 효과적으로 군집화한 알고리즘과 클러스터 개수(c)를 확인

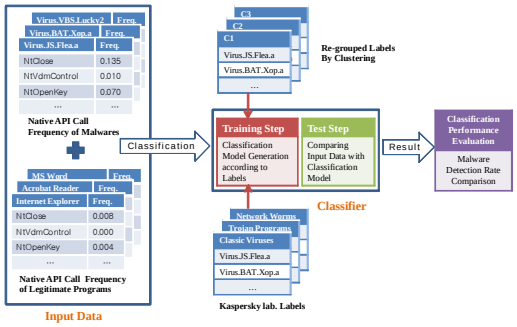
[표 3] 군집화 결과 데이터

수식	행렬 차원	설 명
U_{ij}	$c \times N$	j번째 악성코드가 i번째 클러스터에 소속된 정도를 나타내는 행렬
D_{ij}^2	$c \times N$	클러스터 중심과 데이터간의 거리 행렬
V_i	$c \times n$	i번째 클러스터의 중심값
C_i	상수	클러스터 번호$(1 \le i \le c)$

한다. 가장 적합하게 군집화된 결과를 바탕으로 악성코드를 클러스터 소속정도가 가장 높은 클러스터로 재그룹화하여 레이블을 지정한다.

4.1.2 분류 성능 검증단계

[그림 4]는 분류 순서도를 상세하게 나타낸다.



[그림 4] 분류 순서도

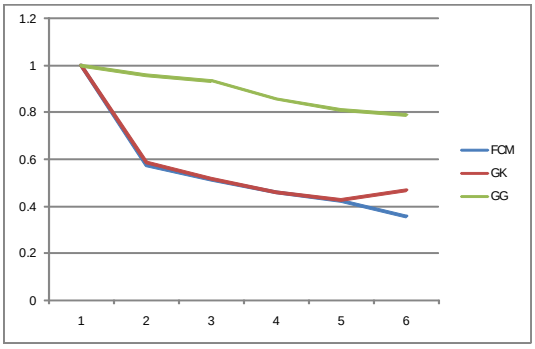
분류 성능 검증단계는 군집화를 바탕으로 분류된 데이터를 일반적으로 사용하는 악성코드 탐지 알고리즘에 적용하여 탐지성능을 분석하는 단계이다. 먼저 탐지와 분류를 병행하기위하여 입력 데이터에 Internet explorer, acrobat, Ms Word등 정상프로그램에서 추출한 정상데이터 27개를 추가하였다.

분류기는 입력데이터를 학습단계와 평가단계로 나누어 처리하게 된다. 이때 학습단계에서 사용하는 레이블을 군집화에 의한 결과와 Kaspersky lab.의 기준으로 나누어 각각 학습하게 한다. 학습단계에서는 전체 데이터의 66%를 사용하였으며 나머지를 평가집합으로 사용하였다. 사용된 탐지 알고리즘은 의사결정나무 분류 방식 중 KDD99Cup 평가에서 가장 우수하다고 알려진 C4.5 방법, 베이지안 분류 방법 중 우수한 성능을 보이는 BayesNet 방법, 그리고 KNN 분류 방법을 사용하였다[9]. 각 알고리즘은 Weca 3.4에서 제공하는 라이브러리를 사용하였다[21]. 두 레이블링 기준의 적합성은 각 분류기에 따른 악성코드 분류 정확도를 비교하여 검증하게 된다.

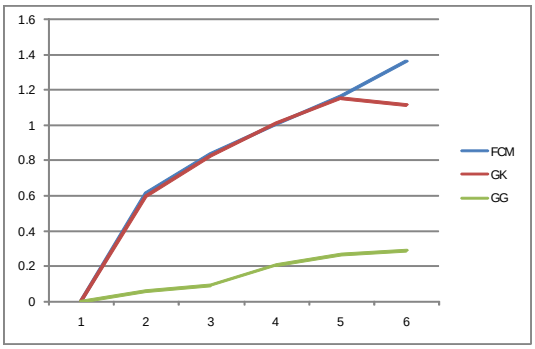
4.2 실험 결과

4.2.1 퍼지 군집화 결과

[그림 5]와 [그림 6]은 군집화 결과 데이터를 PC와 PE로 검증한 결과를 나타낸다. 군집화 결과 FCM과 GK 군집화는 PC에서 0.5~0.3대의 낮은 클러스터 밀집도를 보이고 있다. PE에서도 0.6~1.3으로 높은 엔트로피를 보이고 있어 클러스터가 넓게 분산되어있음을 나타낸다. 반면에 GG 군집화의 경우 PC는 0.9~0.7대의 높은 클러스터 밀집도를 보이고 PE는 0.05~0.3의 낮은 엔트로피를 보여 클러스터가 조밀하게 분산되어있음을 나타낸다. 이는 Native API 군집화에서 거리계산식에



[그림 5] 각 알고리즘별 PC 결과



[그림 6] 각 알고리즘별 PE 결과

지수함수를 포함하여 거리에 따른 소속차이를 크게 한 GG 군집화가 더 효과적임을 나타낸다.

또한 GG 클러스터는 3에서, GK 클러스터는 5에서 그래프의 기울기가 변동이 있음을 보인다. 이는 데이터에 적합한 클러스터의 개수가 각각 3과 5라는 것을 나타낸다. FCM의 경우 2~6 간의 기울기가 일정하여 적합한 클러스터의 개수를 선택하기 어려운 결과값을 보인다. 군집화의 검증 결과 클러스터값을 3으로 설정한 GG 군집화 방식이 Native API 퍼지 군집화에 적합하다는 결과를 도출한다.

4.2.2 파라미터 분석

퍼지 군집화 결과 추출된 파라미터, 데이터의 클러스터 소속정도(U)와 각 클러스터의 대푯값인 중심값(V)을 분석하도록 하겠다. 기존 분류와의 차이점을 비교하기 위해 데이터의 클러스터 소속정도를 대표적인 Anti-Virus 업체인 Kaspersky lab.의 분류 방식과 비교하도록 하겠다. 그리고 클러스터의 중심값을 분석하여 군집화된 클러스터의 성격을 분석하도록 하겠다. [표 4]는 Kaspersky lab.의 분류 방식의 소규모 그룹별 악성코드들이 GG알고리즘으로 분류된 클러스터에 어느 정도 소속(Degree of Membership)하였는지 나타내며, 표 5는 클러스터 대푯값을 나타낸다. U_i는 소분류의 악성코드들이 C_i 클러스터에 소속한 정도를 나타내며, V_i는 C_i 클러스터 중심값을 나타낸다.

[표 4] Kaspersky lab. 분류 방식의 소분류별 클러스터 소속비율 비교(GG 알고리즘, c=3)

대분류	소분류	클러스터 소속 비율		
		U_1	U_2	U_3
Network Worms	Email Worms	33.6%	45.9%	20.5%
	Instant Messaging Worms	1.7%	56.2%	41.9%
	Internet Worms	15%	74.9%	10.1%
	IRC Worms	71.4%	23.2%	5.4%
	File-sharing Networks Worms	10.3%	65.5%	24.1%
Classic Viruses	File and Boot Viruses	90.9%	0%	9.1%
	Macro Viruses	92.5%	7.5%	0%
	Script Viruses	83.9%	16.1%	0%
Trojan Programs	Backdoors	0%	61.2%	38.8%
	PSW Trojans	0%	68.8%	31.2%
	Trojan Clickers	0%	57.1%	42.9%
	Trojan Downloaders	0%	75.0%	25%
	Trojan Droppers	0%	75.0%	25%
	Trojan Proxies	0%	94.4%	5.6%
	Trojan Spies	14.3%	0%	85.7%
	Trojan Notifiers	0%	61.1%	38.9%

C_1은 Kaspersky lab.의 분류방식 중 Classic Viruses의 소속비율이 90%가량으로 높은 유사성을 나타냈다. 클러스터의 중심값에서는 Classic Viruses에서 가상 Dos 머신으로 명령을 전송하는 NtVdmControl과 파일에 데이터를 쓰는 NtWriteFile, 그리고 IRC worm에서 다중객체 동기화를 위해 사용하는 NtWaitForMultiple Objects의 사용비중이 높게 나왔다.

C_2는 다수의 Network Worms와 Trojan을 구성원으로 내포하였다. 클러스터의 중심값에서는 객체에게 핸들을 반환하는 NtClose, 레지스트리 키 객체를 호출하는 NtOpenKey, 그리고 파일의 특정한 부분을 메모리에 mapping시키는 NtMapView OfSection의 사용비중이 높게 나왔다.

C_3는 키 로그나 프로그램 로그를 수집하여 전송하는

[표 5] 클러스터 중심값(GG 알고리즘, c=3)

Native API	V_1	V_2	V_3
NtClose	0.18623	0.33621	0.06619
NtVdmControl	0.22612	0.00573	0.00804
NtUserValidate HandleSecure	0.00158	0.03670	0.53538
NtOpenKey	0.12440	0.19244	0.06514
NtQueryValueKey	0.12744	0.14199	0.02864
NtWriteFile	0.27316	0.01062	0.01457
NtQueryPerformance Counter	0.20175	0.01792	0.00081
NtWaitForMultiple Objects	0.16583	0.01413	0.02350
NtAllocateVirtual Memory	0.10430	0.07533	0.01191
NtMapViewOfSection	0.04644	0.08302	0.00892
NtCreateFile	0.02318	0.01682	0.03705
NtRequestWaitReplyPort	0.09364	0.09211	0.00712
NtQueryInformationFile	0.05556	0.01554	0.05330

Trojan Spies에 밀집한 형태를 나타낸다. 중심값에서도 사용자의 핸들의 유효성을 검증하는 NtUserValidateHandle Secure의 사용비중이 높게 나왔다.

위 군집화 결과를 종합하여 분석하면, GG 군집화의 결과는 Kaspersky lab.의 분류기준과 비교하였을 때 Classic Viruses에 대해서는 유사한 분류를 하였으나 Network Worms와 Trojan은 중첩하여 분류하였으며 Trojan Spies에 대해서는 별도의 클러스터로 분류하는 차이점을 보였다.

4.2.3 분류 성능 비교

[표 6~8]과 [그림 7~10]은 데이터 마이닝을 이용한 3가지 탐지방법에서 Kaspersky lab.의 분류기준에 따라 레이블된 집합을 학습집합으로 사용한 탐지 결과와 군집화 결과에 따라 레이블된 집합을 학습집합으로 사용한 탐지 결과를 비교하여 나타낸다.

3가지 탐지 알고리즘 모두에서 기존 분류 기준은 Native API기반의 탐지에 있어서 낮은 탐지율(True-Positive)과 높은 오탐율(False-Positive)을 보인다. Classic Viruses는 클래스 내의 유사성이 비교적 높은 편이어서 타 클래스에 비해 높은 탐지율과 낮은 오탐율을 나타냈다. 하지만 Network Worms와 Trojan Programs는 유사한 성격을 지닌 소속 악성코드들로 인하여 클래스의 경계가 중첩됨에 따라 높은 오탐율을 보였다. 이러한 클래스간의 중첩은 정상 프로그램에 대해서도 잘못된 탐지를 유도하게 되었다.

[표 6] 의사결정나무를 적용한 탐지 결과

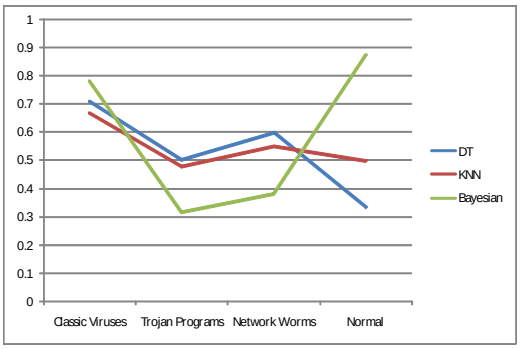
Class	TP Rate	FP Rate	Class	TP Rate	FP Rate
Classic Viruses	0.708	0.151	C_1	0.938	0.050
Trojan Programs	0.500	0.089	C_2	0.941	0
Network Worms	0.598	0.324	C_3	0.889	0.027
정상 프로그램	0.333	0.066	정상 프로그램	0.875	0.024

[표 7] 베이지안 분류를 적용한 탐지 결과

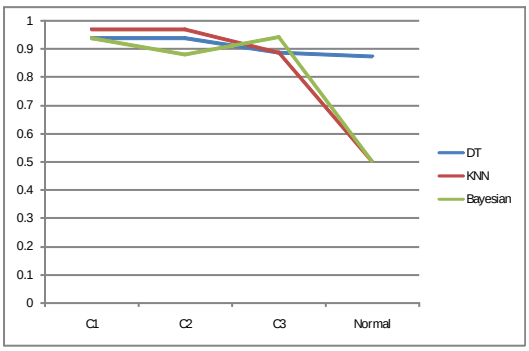
Class	TP Rate	FP Rate	Class	TP Rate	FP Rate
Classic Viruses	0.783	0.174	C_1	0.938	0.083
Trojan Programs	0.318	0.057	C_2	0.882	0
Network Worms	0.385	0.358	C_3	0.944	0.081
정상 프로그램	0.875	0.119	정상 프로그램	0.500	0

[표 8] KNN을 적용한 탐지 결과

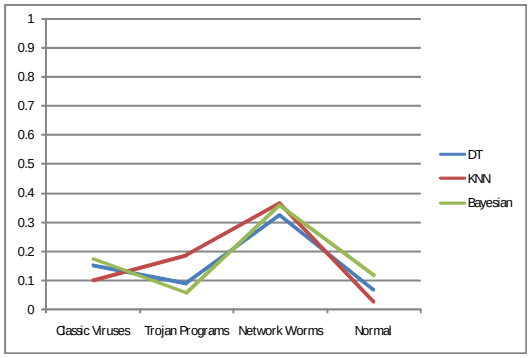
Class	TP Rate	FP Rate	Class	TP Rate	FP Rate
Classic Viruses	0.667	0.099	C_1	0.969	0.050
Trojan Programs	0.478	0.188	C_2	0.971	0.017
Network Worms	0.550	0.365	C_3	0.889	0.041
정상 프로그램	0.500	0.024	정상 프로그램	0.500	0.012



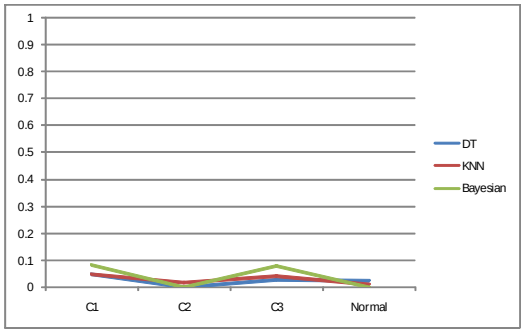
[그림 7] 카스퍼스키 기준 분류의 탐지율



[그림 8] 군집화 기준 분류의 탐지율



[그림 9] 카스퍼스키 기준 분류의 오탐율



[그림 10] 군집화 기준 분류의 오탐율

제안된 군집화 방법을 적용한 탐지는 기존방식에 비해 높은 탐지율과 낮은 오탐율을 나타냈다. C_1은 Classic Viruses와 유사한 클러스터로 타 클러스터에 비해 높은 탐지율을 보였으나 다소 높은 오탐율을 나타냈다. C_2 는 Network Worms와 Trojan Programs의 특성을 포괄하는 클러스터로 기존 분류의 중첩성격을 반영하여 가장 낮은 오탐율을 나타냈다. C_3는 Trojan Spies에 밀집된 클러스터의 성격을 보이면서 타 클러스터에 비해 낮은 탐지율을 보였다.

결과적으로 Kaspersky lab.의 분류기준은 Native API기반의 탐지에서 낮은 탐지율과 높은 오탐율을 나타냄에 따라 적합지 않은 방법이었으며, 제안된 군집화 방법에 따른 분류방식은 높은 탐지성능을 나타내 Native API기반의 탐지에 적합한 방식임을 증명한다.

Ⅴ. 결 론

본 논문에서는 Windows 시스템에서 사용하고 있는 system call의 일종인 Native API를 이용하여 악성코드를 퍼지 군집화하여 재그룹화하는 방법을 연구하였다. 소규모로 구성된 네트워크에서 IDT 후킹 기법을 이용하여 Native API를 추출하였고 이를 FCM, GK, GG 알고리즘을 이용하여 퍼지 군집화하였다. 분류된 결과는 PC와 PE를 통하여 군집화와 클러스터의 개수의 적합성를 검증하였다.

이 중 GG알고리즘을 이용한 군집화 방법이 악성코드의 Native API를 분류하는데 적합함을 확인하였으며, Kaspersky lab.의 분류기준과의 유사성과 차이점을 비교하였다. 실험에서 얻어진 군집화 결과는 Native API의 고유한 성질을 반영한 분류 방법으로 Native API를 이용한 악성코드탐지 알고리즘에 적용하였을 때 기존의 분류방법에 비해 향상된 탐지 성능을 유도함을 증명한다.

향후 연구로는 여러 Anti-Virus 업체를 대조군으로하여 악성코드의 Native API를 대규모로 수집하고 클러스터를 더욱 세분화하는 연구와 군집화된 그룹을 Native API의 특징점에 맞는 명명법을 개발하고 이를 바탕으로 새로운 공격양상을 자동으로 파악하는 연구가 있겠다.

참고문헌

[1] 국가사이버안전센터, “2008 국가정보보호백서”, 2008.

[2] R. Sadoddin, A. Ghorbani, “A Comparative Study of Unsupervised Machine Learning and Data Mining Techniques for Intrusion Detection.” *Springer Berlin, LNCS 4571*, pp. 404-418, 2007.

[3] SB. Cho, HJ. Park, “Efficient anomaly detection by modeling privilege flows using hidden Markov model.” *Computer & Security*, Vol 22, No 1, pp 45-55, 2003.

[4] 김미희, 오하영, 채기준, “데이터 마이닝을 이용한 공격 탐지 메커니즘의 실험적 비교연구”, *한국통신학회논문지 '06-2*, Vol 31, No 2C, pp. 208-218, Feb. 2006.

[5] Greg Hoglund, James Butler, “Rootkits: Subverting the Windows Kernel.” *Pearson Education*, Inc, 2006.

[6] Michael Bailey, Jon Oberheide, Jon Andersen, Z. Morley Mao, Farnam Jahanian, and Jose Nazario, “Automated classification and analysis

of internet malware." *RAID '07, LNCS 4637*, pp. 178-197, 2007.

[7] N Idika, AP Mathur, "A Survey of Malware Detection Techniques." *cs.purdue.edu*, 2007.

[8] 강태우, 조재익, 정만현, 문종섭, "API call의 단계별 복합분석을 통한 악성코드 탐지", *정보보호학회 논문지 제17권 제6호*, pp. 89-98, Dec. 2007.

[9] Seung-Hyun Paek, Yoon-Keun Oh, JooBeom Yun, Do-hoon Lee, "The Architecture of Host-based Intrusion Detection Model Generation System for the Frequency Per System Call", *ICHIT*06, 2(2), pp.277-283, 2006.

[10] A.A. Abimbola, J.M. munoz, W.J. Buchanan, "NetHost-Sensor: Monitoring a target host's application via system call." *Information Security Technical Report, Elsevier*, pp. 166-175, Oct. 2006.

[11] NK Boora, C Bhattacharyya, K Gopinath, "Efficient Algorithms for Intrusion Detection." *ICDCIT 2004, LNCS 3347*, pp. 346-352, 2004.

[12] R Battistoni, E Gabrielli, LV Mancini, "A Host Intrusion Prevention System for Windows Operating System." *ESORICS 2004, LNCS 3193*, pp. 352 – 368, 2004.

[13] 박한샘, 유시호, 조성배, "Gath-Geva 알고리즘을 이용한 유전자 발현 데이터의 분석", *한국정보과학회 2004년도 봄 학술발표논문집 제31권 제1호(B)*, pp. 253-255, Apr. 2004.

[14] B. Balasko, J. Abonyi and B. Feil, "Fuzzy clustering and data analysis toolbox." http://www.fmt.vein. hu/*softcomp/fclusttoolbox/*

[15] 배성재, 권오철, 문종섭, 임종인, "Windows 악성코드 탐지를 위한 Native API 빈도에 기반한 접근 방법에 관한 연구", *2008 한국컴퓨터종합학술대회 논문집*, Vol.35, No.1(A), pp 132-133, 2008.

[16] M Wang, C Zhang, J YU, "Native API Based Windows Anomaly Intrusion Detection Method Using SVM." *SUTC* '06, 2006.

[17] Gary Nebbett, *Windows NT/2000 Native API Reference*, Macmillan Technical Publishing (MTP), February 15, 2000.

[18] Mark Russinovich, "INSIDE THE NATIVE API." http://www.spies. *informatik.tu-muenchen.de/lehre/praktika/SS02/bsprakt/inside_the_native_api.html*

[19] Kaspersky lab., http://www.viruslist.*com*

[20] Offensive Computing, http://www.*offensivecomputing.net*

[21] Weka, http://www.cs.waikato.ac.nz/ml/*weka/*

[22] N Ye, X Li, Q Chen, SM Emran, M Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data." *Man and Cybernetics, Part A, IEEE* Transactions on, Vol 31, No 4, pp. 266-274, Jul 2001.

[23] 전영준, 김진일, "퍼지 알고리즘의 융합에 의한 다중분광 영상의 패턴분류", *한국정보과학회 논문지 : 소프트웨어 및 응용 제32권 제7호*, pp. 674-682, Jul. 2005.

[24] Sergios Theodoridis, Konstantinos Koutroumbas, *Pattern Recognition 3rd Edition*, Academic Press, 2006.

[25] CME(Common Malware Enumeration), *http://cme.mitre.org/*

[26] Ross Quinlan, C4.5: *Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[27] Sukhan Lee, Shunichi Shimoji, "BAYESNET: Bayesian classification network based on biased randomcompetition using Gaussian kernels", *Neural Networks, IEEE International Conference on*, Vol.3, pp.1354-1359,1993.

[28] Yihua Liao and V. Rao Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection." *Computers & Security*, Vol 21, Issue 5, pp. 439-448, Oct 2002.

<著 者 紹 介>

권 오 철 (O-chul Kwon) 학생회원
2002년 3월 : 공군사관학교 컴퓨터공학과 학사
2007년 3월~현재 : 고려대학교 정보경영공학전문대학원 석사과정
<관심분야> 시스템 보안, 네트워크 보안, 데이터 마이닝

배 성 재 (Seong-jae Bae) 학생회원
2005년 8월 : 서울시립대학교 컴퓨터통계학과 졸업
2007년 9월~현재 : 고려대학교 정보경영공학과 석사과정
<관심분야> 침입탐지, 악성 코드, 데이터 마이닝

조 재 익 (Jae-ik Cho) 학생회원
2005년 2월 : 동국대학교 컴퓨터학과 학사
2008년 2월 : 고려대학교 정보경영공학전문대학원 석사
2008년 3월~현재 : 고려대학교 정보경영공학전문대학원 박사과정
<관심분야> 네트워크 모델링, 패턴인식

문 종 섭 (Jong-sub Moon) 정회원
1981년~1985년 : 금성 통신 연구소 연구원
1991년 : Illinois Institute of technology 졸업(전산학 박사)
1993년~현재 : 고려대학교 전자 및 정보공학부 교수
<관심분야> 생체인식, 침입탐지, 운영체제