情報保護學會論文誌
第 22 卷 第 4 號, 2012. 8

Native API 의 효과적인 전처리 방법을 이용한 악성 코드 탐지 방법에 관한 연구

배 성 재,[1†] 조 재 익,[1] 손 태 식,[2] 문 종 섭[1‡]
[1]고려대학교, [2]아주대학교

Malicious Code Detection using the Effective Preprocessing Method Based on Native API

Seong-jae Bae,[1†] Jae-ik Cho,[1] Tae-shik Shon,[2] Jong-sub Moon[1‡]
[1]Korea University, [2]Ajou University

요 약

본 논문에서는 악성코드의 시스템 콜 빈도수를 특징값으로 행위 기반 탐지(behavior-based detection)를 할 때, 시스템 콜의 속성 개수보다 학습데이터 개수가 적더라도 효과적으로 악성 코드를 탐지하는 기법을 제안한다. 이 연구에서는, 프로그램 코드가 동작할 때, 발생시키는 윈도우 커널 데이터인 Native API를 수집하여 빈도수로 정규화한 것을 기본적인 속성 값으로 사용하였다. 또한 악성코드와 정상 코드를 효과적으로 분류할 수 있으면서, 악성코드를 분류하기 위한 기본적인 속성의 개수보다 학습데이터 개수가 적어도 적용 가능한 GLDA(Generalized Linear Discriminant Analysis)를 사용하여, 새로운 속성 값들로 전환하였다. 분류 기법으로는 베이지언 분류법의 일종인 kNN(k-Nearest Neighbor) 분류법을 이용하여 악성 코드를 탐지하였다. 제안된 탐지 기법의 성능을 검증하기 위하여 수집된 Native API 로 기존의 연구 방법과 비교 검증하였다. 본 논문에 제안된 기법이 탐지율(detection rate) 100%인 Threshold 값에서, 다른 탐지 기법보다 낮은 오탐율(false positive rate)을 나타내었다.

ABSTRACT

In this paper, we propose an effective Behavior-based detection technique using the frequency of system calls to detect malicious code, when the number of training data is fewer than the number of properties on system calls. In this study, we collect the Native APIs which are Windows kernel data generated by running program code. Then we adopt the normalized freqeuncy of Native APIs as the basic properties. In addition, the basic properties are transformed to new properties by GLDA(Generalized Linear Discriminant Analysis) that is an effective method to discriminate between malicious code and normal code, although the number of training data is fewer than the number of properties. To detect the malicious code, kNN(k-Nearest Neighbor) classification, one of the bayesian classification technique, was used in this paper. We compared the proposed detection method with the other methods on collected Native APIs to verify efficiency of proposed method. It is presented that proposed detection method has a lower false positive rate than other methods on the threshold value when detection rate is 100%.

Keywords: Malicious code, Intrusion detection system, GLDA

접수일(2011년 12월 12일), 수정일(2012년 2월 14일),
게재확정일(2012년 2월 14일)

† 주저자, baeseongjae@korea.ac.kr
‡ 교신저자, jsmoon@korea.ac.kr

I. 서 론

침입 탐지 시스템(intrusion detection system)은 호스트에서 동작하는 악성코드를 탐지하는 기술이다. 이러한 침입 탐지 시스템은 시그너처 기반 탐지(signature-based detection)[1]와 행위 기반 탐지(behavior-based detection)[2]로 분류 된다. 시그너처 기반의 탐지 기법은 악성코드의 특정 시그너처 또는 미리 정의된 설명들을 추출하여 탐지에 이용하는 방법이다. 이러한 기법은 알려져 있는 악성코드는 탐지할 수 있지만, 알려지지 않은 악성코드를 탐지할 수는 없다. 이에 반해, 행위 기반 탐지 기법은 정상 코드에 대한 모델을 생성한 후, 이 모델에서 벗어난 것을 악성코드로 판별하는 것이다. 이러한 방법을 이용하면 알려지지 않은 악성코드도 탐지가 가능하다. 하지만 행위 기반 탐지 기법은 탐지에 필요한 효과적인 모델을 생성하기 위하여 충분히 많은 샘플 데이터가 필요하다.

호스트에서 동작하는 악성코드를 탐지하기 위한 행위 기반 탐지에서 정상 코드를 모델링 하는 주요한 특징 값으로 시스템 콜을 이용할 수 있다[3]. 정상 코드와 악성 코드를 판별할 수 있는 효과적인 모델을 생성하기 위해서는, 시스템 콜 특징 개수보다 상당히 많은 샘플 데이터의 개수가 필요하다[4]. 샘플 데이터가 특징 개수보다 적으면 효과적인 모델을 생성할 수 없어서, 악성코드를 탐지하기 위한 행위 기반 탐지 기법을 적용할 수 없다.

하지만 본 논문에서는 충분한 샘플 데이터가 없어도 효과적으로 윈도우 상에서 동작하는 악성 코드를 탐지할 수 있는 행위 기반 탐지 방법을 제안한다.

본 논문에서는, 윈도우 환경에서 동작하는 악성 코드를 탐지하기 위하여, 윈도우에서 발생시키는 시스템 콜인 Native API[16]의 frequency property 값들을 정규화 한 후, 속성의 개수보다 학습데이터 개수가 적어도 적용 가능한 GLDA(Generalized Linear Discriminant Analysis)를 적용하여 새로운 특징값들로 변환한 후, 이 특징 값들을 kNN(k-Nearest Neighbor) 분류 알고리즘을 적용하여 악성 코드를 탐지하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 시스템 콜 데이터를 이용한 악성 코드 탐지에 관한 관련 연구에 대하여 기술하고, 3장에서는 악성 코드를 탐지하기 위한 방법을 제안하고, 4장에서는 제안된 기법에 대한 검증 결과를 보이고, 5장에서는 결론 및 향후 연구 방향에 대하여 기술한다.

II. 관련연구

2.1 시스템 콜을 사용한 악성 코드 탐지 기법

호스트 기반 탐지에 시스템 콜을 이용하는 연구는 Forrest[3]가 처음으로 제안하였다. 이후 시스템 콜을 이용한 호스트 기반 탐지 기법은 다양하게 연구되었다[6-7][9-15]. 유닉스 환경의 BSM(Basic Security Module)을 이용하여 수집할 수 있는 시스템 콜을 특징값으로 사용하거나, 윈도우 환경의 Native API 를 이용하여 생산한 특징 값들을 이용한, 호스트 기반 탐지 기법에는 시스템 콜의 발생 순서를 고려한 ordering property 기법과 시스템 콜의 발생한 횟수만을 고려한 frequency property 기법이 있다[5].

ordering property를 이용한 탐지 기법은 데이터가 발생한 순서적인 규칙성을 이용하여 악성 코드를 탐지하는 기법이다. 이에 반해 frequency property를 이용한 탐지 기법은 발생한 데이터의 빈도수의 규칙성을 이용한 악성 코드 탐지 기법이다.

ordering property를 이용한 탐지 기법은 정상인 프로세스들을 모델링할 때, 정상인 프로세스를 개별적으로 모델링하여 정상에 대한 여러 개의 모델이 생성되고, 각각의 임계값을 설정한다. 특정한 코드가 악성 코드인지 여부를 판단하기 위해서는 여러 개의 정상 모델의 임계값과 모두 비교하여 악성 코드를 판단한다. 이러한 기법은 시스템에 많은 부하를 발생시킨다. 반면에 frequency property를 이용한 탐지 기법은 정상인 프로세스들을 모델링할 때, 모든 프로세스를 한 번에 모델링하여 한 개의 정상 모델만을 생성한다. 따라서 이와 같은 탐지 기법은 시스템에 부하를 적게 한다.

2.2 ordering property 를 이용한 탐지 기법

윈도우 환경에서의 탐지기법 중에서, ordering property 탐지 기법으로, Miao Wang[6]이 악성 코드와 정상 코드에서 발생하는 Native API 데이터가 나온 순서대로 6개만 사용하여, 특징값으로 생성하여, SVM(Support Vector Machine)을 사용하여 공격을 판별하는 기법을 제안하였다. 99%로 높은 탐지율을 보이나, Deborah Buckley[17]가 지적한 것

과 같이 6개씩 나눠진 데이터셋에 SVM 알고리즘을 이용하여 탐지하기 때문에, 시스템 부하를 많이 발생시켜 실시간 탐지에 이용하기 어려운 단점이 있다. 박남열[7]은 변형된 악성 코드를 탐지하기 위하여, 원형과 변형된 악성 코드의 API 데이터의 순서를 추출하고, 이에 대한 유사도를 측정하여 변형된 악성 코드를 탐지하는 기법을 제안하였다. 이 기법을 이용하면 실행 압축과 같은 알려진 변형 방법을 이용한 악성 코드를 효과적으로 탐지할 수 있지만, 알려지지 않은 변형 방법을 이용한 악성 코드를 분류하는데 적용할 수 없다는 한계점이 있다.

유닉스 환경에서의 탐지 기법 중, ordering property 탐지 기법으로, Forrest[3]는 짧은 구간으로 시스템 콜을 순서대로 나눠서 공격을 탐지하는 기법을 제안하였다. 이 기법은 정상 코드가 수행될 때에는 매우 일정한 시스템 콜의 순서를 가지는데, 이 성질을 이용하여, 설정된 임계값을 넘으면 공격으로 판단한다. 이러한 연구를 기반으로 Warrender et al.[8]은 시스템 콜의 순서에 STIDE(Sequence Time-Delay Embedding), T-STIDE(Threshold Sequence Time-Delay Embedding), RIPPER(Repeated Incremental Pruning to Produce Error Reduction) 그리고 HMM(Hidden Markov Model) 알고리즘 등, 4가지 방법을 적용하여, 각 방법들의 탐지 성능을 비교 분석하였다. 이 연구에서 HMM 알고리즘이 가장 좋은 성능을 보였다. 하지만 HMM 알고리즘은 모든 시스템 콜을 사용하여 정상을 모델링하고 적용하는데 계산량이 많이 소요되는 단점이 있다. 이러한 단점을 해결하기 위해서 조성배[11]는 HMM 알고리즘으로 정상을 모델링할 때 모든 시스템 콜을 사용하지 않고, 일반 사용자의 권한을 관리자의 권한으로 바꾸는 시스템 콜만으로 모델링을 진행하여 이러한 문제에 대한 해결 방안을 제안하였다. 조성배가 제안한 탐지 기법은 일반 사용자의 권한을 관리자의 권한으로 바꾸는 시스템 콜만을 사용하였기 때문에, 권한 상승과 관련된 공격의 탐지에는 효과적이지만 권한 상승을 하지 않는 악성 코드를 탐지하는 데는 한계가 있다.

윈도우 환경에서 악성 코드를 탐지하기 위한 기법을 사용자 호스트에 적용하기 위해서는 시스템의 부하가 적어야 한다. 하지만 대부분의 ordering property 탐지 기법은 정상을 모델링하기 위하여 짧은 시스템 콜의 순서 구간을 이용한다[3][6][8-11]. 이러한 기법은 각 프로세스마다 따로 정상에 대한 모델링을 진행해야 한다. 또한 공격에 대하여 탐지하기 위해서는 시스템 콜 순서 구간마다 검사를 진행해야 한다. 그래서 ordering property 기법은 시스템에 많은 부하를 발생시킨다. 윈도우 환경의 악성 코드를 탐지하기 위하여 이러한 기법을 적용하는데 한계점이 있다. 시스템 부하를 많이 주는 Orderging property 탐지 기법의 단점을 극복하기 위해서, 시스템 콜의 frequency property 탐지 기법들[12-15]이 제안 되었다.

2.3 frequency property 를 이용한 탐지 기법

Native API 데이터의 시스템 콜을 이용한 frequency property 탐지 기법으로 강태우[12]가 단계별 복합분석 기법을 제안하였다. API 데이터에 정적 분석과 동적 분석을 통하여 가중치를 부여하고 전처리한 후, 전 처리된 데이터에 분류 알고리즘인 SVM을 사용하여 악성 코드를 판별하였다. 96.67%로 높은 탐지율을 보이나 실행 압축하는 악성 코드에 최적화 되어 있는 기법이어서 실행 압축하지 않는 악성 코드를 탐지하는데 한계가 있다.

BSM 데이터의 시스템 콜의 frequency property 탐지 기법으로 Liao 와 Vemuri 가 제안한 기법[13]이 있는데, 이 기법은 시스템 콜의 빈도수를 텍스트마이닝 기법에서 사용되는 TF-IDF(Term Frequency-Inverse Document Frequency)로 가중치를 부여하는 전처리를 한 후, kNN 분류기를 이용하여 선택된 정상 데이터와 테스트 데이터의 유사도 값의 평균값을 구한 후, 특정 임계값보다 작으면 공격이라고 판단하는 이상행위 탐지 기법이다. 여기서 TF-IDF로 가중치를 부여하는 전처리 기법은 각 시스템 콜을 단어로 간주하고, 시스템 콜의 집합인 프로그램을 문서로 간주하여 가중치를 부여한다. 하지만 이 방법은 kNN 분류기의 k값이 5일때, 악성코드 탐지율(detection rate)이 100%인 Threshold 값 0.9934인 경우, 정상코드를 악성코드로 잘못 탐지하는 오탐률(false positive rate)은 22.84%로 높았다.

이를 개선하기 위한 탐지 방법으로 Sanjay Rawat 이 유사도 측정 기법인 BWCM(Binary-Weighted Cosine Metric)을 제안하였다[14]. 이 방법에서는 k값이 5인 경우, 탐지율 100%인 Threshold 값 0.90099에서의 오탐률을 4.65%로 낮게 개선하였다.

Alock Sharma 는 높은 오탐률을 개선하고자 커널 유사도 측정의 텍스트 처리 기법[15]을 제안하였다. 이 논문에서는 k값이 5이고 Threshold 값이 0.99255인 경우, 탐지율은 100%, 오탐률은 0.38%로 개선했다.

TF-IDF 전처리를 한 후, 유사도 비교를 통하여 공격을 탐지한 연구[13-15]는 간단한 알고리즘을 이용하여 시스템 부하도 적고, 효과적인 탐지 기법이지만, 유닉스 환경의 1998년 MIT Lincoln Laboratory 데이터를 대상으로 한 탐지 기법이기 때문에, 현재의 윈도우에서 발생되는 악성 코드에 대한 탐지에 적용하기 어렵다.

이러한 단점을 해결하기 위하여, 본 논문에서는 새로운 전처리 기법을 적용한 시스템에 부하가 적은 frequency property 탐지 기법을 제안한다.

III. 제안하는 탐지 기법

3.1 탐지 모델 생성

악성 코드를 탐지하기 위한 탐지 모델 구축 시, 시스템의 부하를 적게 하기 위해서는 정상 코드와 악성 코드 데이터의 주된 속성 값을 추출하여 해당 속성 값만을 이용하면 모델 구축 시간 및 탐지 시간도 단축되는 장점이 있다. 자료의 주성분을 추출하는 방법에는 PCA(Principle Component Analysis)[18] 와 같은 방법도 있다. 하지만 이러한 방법은 해당 데이터를 대표하는 주성분을 추출하는 방법이지, 분류를 최대화하기 위한 방법이 아니다. 반면에 LDA(Linear Discriminant Analysis) 알고리즘은 정상 코드와 악성 코드를 최대한으로 분류하기 위한 최적의 속성을 생산하는 방법이다[19].

3.1.1 Linear Discriminant Analysis

주어진 행렬 $X \in R^{m \times n}$가 있을 때, m개의 차원을 가진 X의 각 열벡터 $x_i, 1 \le i \le n$를 l개의 차원을 가진 열벡터 $y_i, 1 \le i \le n$로 변환하는 선형 변환 행렬 $W^t \in R^{l \times n}$를 구해 보자.

즉, $W^t : x_i \in R^m \rightarrow y_i \in R^l$을 구한다. 이때, X가 이미 여러 개의 클래스로 나누어져 있다고 가정하면 W는 원래의 클래스를 그대로 유지하면서, 특징의 차원이 줄어든 새로운 데이터를 합성하는 행렬이다. 만일 클래스의 개수가 k 개일 경우, $X=[X_1, \cdots, X_k]$, $X_i \in R^{m \times n_i}$, $\sum_{i=1}^{k} n_i = n$로 표시할 수 있고, N_i는 i번째에 속하는 데이터의 인덱스 집합이다. 이때 클래스 내의 분산 행렬(within-scatter matrix)을 S_w라 하고, 클래스간의 분산(between-scatter matrix)를 S_b라 하고, 데이터 전체의 분산(total scatter)을 S_t라 하면, LDA의 목적은, 새로 변환 된 데이터에서 S_w는 최소화하고, S_b 또는 S_t는 최대화 하는 것이다. 즉, X로부터 변환된($W^t X$) Y로부터 이루어진 수식 (1)의 목적함수를 최대화 하는 W를 구하는 것이다. 우선 최적의 W를 구하기 위해서는 수식 (2), (3)과 같이 사영시키기 전의 데이터의 $S_w^{(x)}$와 $S_b^{(x)}$를 유도하여, 수식 (4)에 대입하면, 원하는 행렬 W를 구할 수 있다.

$$f(w) = trace\left\{\frac{S_b^{(y)}}{S_w^{(y)}}\right\} \tag{1}$$

여기서, $S_b^{(y)}, S_w^{(y)}$는 각각 Y 데이터의 S_b, S_w이다.

$$S_w^{(y)} = \sum^{i} S_{wi}^{(y)}, \quad S_b^{(y)} = \sum^{i} S_{bi}^{(y)},$$

여기서, $S_{wi}^{(y)}$와 $S_{bi}^{(y)}$는 각각 Y 데이터의 i번째 class의 within-scatter matrix와 between-scatter matrix이다.

이 matrix들은 다음 수식 (2)와 (3)으로 다시 표현된다.

$$\begin{aligned} s_{wi}^{(y)} &= \sum_{Y \in N_i} (Y - \mu_i^{(y)})(Y - \mu_i^{(y)})^T \\ &= \sum_{X \in N_i} (W^T X - W^T \mu_i^{(x)})(W^T X - W^T \mu_i^{(x)})^T \\ &= \sum_{X \in N_i} W^T (X - \mu_i^{(x)})(X - \mu_i^{(x)})^T W \\ &= W^T S_{wi}^{(x)} W \end{aligned} \tag{2}$$

$$\begin{aligned} s_b^{(y)} &= \sum_{Y \in N_i} (\mu_i^{(y)} - \mu^{(y)})(\mu_i^{(y)} - \mu^{(y)})^T \\ &= \sum_{X \in N_i} (W^T \mu_i^{(x)} - W^T \mu^{(x)})(W^T \mu_i^{(x)} - W^T \mu^{(x)})^T \\ &= \sum_{X \in N_i} W^T (\mu_i^{(x)} - \mu^{(x)})(\mu_i^{(x)} - \mu^{(x)})^T W \\ &= W^T S_b W \end{aligned} \tag{3}$$

단, $\mu_i^{(x)}, \mu_i^{(y)}$는 각각 i번째 그룹 X와 Y 데이터의 평균이고, μ는 각각 X와 Y 데이터의 전체 평균이다. 그래서 수식(1)은 다음과 같이 표현된다.

$$f(w) = trace\left\{\frac{S_b^{(y)}}{S_w^{(y)}}\right\} = trace\left\{(W^t S_w^{(x)} W)^{-1}(W^t S_b^{(x)} W)\right\} \quad (4)$$

수식 (4)를 최대화 시키는 W는 $S_b^{(x)}W = \lambda S_w^{(x)}W$를 만족시키는 eigenvector를 열벡터로 하는 행렬이 되고, 이때, 최대의 l은 $k-1$이 된다. 따라서, 클래스가 2개인 경우는 l은 1 이 되어서, 변환된 y_i는 1차원 데이터로 변한다. 즉, 이 경우

$$W = (S_w^{(x)})^{-1} \bullet (u_1^{(x)} - u_2^{(x)}) \quad (5)$$

가 된다. 수식 (5)에서 보면 $S_w^{(x)}$ 의 역행렬을 구해야 하는데,. 데이터 개수가 속성 개수(m) 보다 적으면, $S_w^{(x)}$가 singular 가 되어서 역행렬을 구할 수가 없다. $S_w^{(x)}$역 행렬이 존재 하지 않으면, LDA를 직접 사용할 수 없다. 따라서, $S_w^{(x)}$가 singular가 되는 것을 피하기 위해서, $S_w^{(x)}$대신에 $S_w^{(x)}+\alpha I$ (I 는 Identity 행렬)를 사용하는 regularized LDA[20]을 사용하거나, Pseudo-inverse based LDA[21]를 사용하기도 하나, $S_b^{(x)}$ 마저 singular이면, 이 마저도 사용하기가 힘들다.

따라서, 본 논문에서는 GSVD(Generalized Singular Value Decomposition)[22]을 사용하며, 이 수식은 아래에 나와 있다.

3.1.2 Generalized Linear Discriminant Analysis

데이터 X에 대하여, $u_i = \frac{1}{n_i} X_i e^{(i)}$,

단 $e^{(i)} = (1,1,...,1)^t \in R^{n_i}$이고, 전체 평균 u는 $u = \frac{1}{n} Xe$, 단 $e = (1,1,...,1)^t \in R^n$ 하면,

$$H_w = [X_1 - u_1(e^{(1)})^t, ..., X_k - u_k(e^{(k)})^t]$$
$$H_b = [\sqrt{n_1}(u_1 - u), ..., \sqrt{n_k}(u_k - u)] \quad (6)$$

$$S_B = H_b H_b^T,\ S_W = H_w H_w^T \quad (7)$$

$$Z = \begin{pmatrix} H_b^T \\ H_w^T \end{pmatrix} \in R^{(r+n)\times m} \quad (8)$$

가 된다, 이때, Z는,

$$Z = P\begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} U^T \quad (9)$$

가 된다.

여기서 $P \in R^{(r+n)\times(r+n)}$, $U \in R^{m\times m}$,

$\Lambda \in R^{s\times s}$, $s = rank(Z)$이다. 수식 (9)에서 P, U는 orthogonal matrix이고, Λ는 diagonal matrix 이다. 다시 P를 SVD(Singular Value Decomposition)한다. 즉,

$$P(1:r, 1:s) = WTV^T \quad (10)$$

수식 (9)의 U, Λ와 수식 (10)의 V를 사용하여,

$$A = U\begin{pmatrix} \Lambda^{-1}V & 0 \\ 0 & I \end{pmatrix} \quad (11)$$

A를 산출하고, 다음 수식 (12)에서, 변환 행렬 W를 구한다. 즉,

$$W^T = A_{\delta^*}^t,\ \delta^* = rank(H_b) \quad (12)$$

A^t 에서 δ^* 열까지의 vector 구성된 matrix 가 사영축(W)이 된다. 따라서,

$$Y = W^T \bullet X \in R^{l\times n} \quad (13)$$

수식 (12) 구한 사영축을 이용하여, 수식 (13)과 같이 각 코드의 Native API를 사영축에 사영시켜 1 차원 데이터로 사영시킨다.

3.1.3 k-Nearest Neighbor method

베이지안 분류기는 오 분류될 확률을 최소화 하는 분류 방법이다. 분류해야 할 그룹 $w_i, i=1,...,m$가 있고, 데이터 x가 발생 했을 때, x가 그룹 w_i 에 속할 확률은

$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)},\quad P(x) = \sum_{i=1}^{m} P(x|w_i)P(w_i) \quad (14)$$

이때

$$P(w_i|x) > P(w_j|x),\ 1 \le j \le m,\ j \ne i \quad (15)$$

이면 x는 그룹 w_i 속한다. 수식 (14)에서 분모는 모든 그룹에서 동일하므로, (15)를 (14)에 대입하면,

$$P(x|w_i)P(w_i) > P(x|w_j)P(w_j),\ \forall j, j \ne i \quad (16)$$

이 된다. 즉 x를 (16)을 만족하는 그룹 w_i 에 귀속시킨다.

위의 베이지안 분류기를 이용하기 위해서는 $P(x|w_i)$

를 추정해야 한다. 하지만 데이터 X에 대한 분포함수를를 알고 있을 경우, 이 분포함수를 추정하는 모수를 최대우도비 방법(maximum likelihood method[19])을 사용하여 추정할 수 있다. 그러나, 분포함수 자체를 모를 경우는, 모수 추정없이 바로 함수를 추정해야 하는데, 비모수 밀도 추정 방법을 이용해야 한다. 비모수 밀도 추정 방법에는 KDE(Kernel Density Estimation) 접근법인 parzen window 방법 [23] 과 kNN [24] 방법이 있다. parzen window 방법은 데이터가 많은 경우에는 계산량이 너무 많아지고, 일반화의 성능이 많이 떨어진다. 이 논문에서는 계산량이 간단한 kNN 방법을 사용한다. kNN의 이론적 근거는 아래와 같다.

샘플의 빈도를 사용하여, 확률을 계산할 때, 일정한 구간 h 내에 k 개의 샘플이 존재하면, 확률은

$$P \cong \frac{k}{N} \tag{17}$$

로 추정한다. 따라서, 구간의 중간 위치를 $\hat{x}$할 때, 구간 내의 임의 x에서 개수가 k_N 일 때, 확률 추정 함수는

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{h}\frac{k_N}{N}, |x-\hat{x}| \le \frac{h}{2} \tag{18}$$

로 표현된다. 이때, $N \to \infty$이고 다음 조건 (19)이 만족되면, (18)은 진짜 확률 p함수로 수렴한다.

$$h \to 0, k_N \to \infty, \frac{k}{n} \to 0 \tag{19}$$

식 (18)에서, 각 x가 l 차원이면, 구간 h는 체적 h^l로 표기된다. 즉, (18)은

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{h^l}\frac{k_N}{N}, |x-\hat{x}| \le \frac{h}{2} \tag{20}$$

과 같이 표현된다.

만일, 체적을 일정하게 하지 않고, 일정한 데이터의 개수(k)를 가진 체적을 결정하게 한다면, 수식 (20)은

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{V(x)}\frac{k}{N} \tag{21}$$

로 변환되고, 식 (21)을 베이지안 분류기 식 (16)에 대입하면,

$$\begin{aligned} &\frac{k}{V_i N_i} P(w_i) > \frac{k}{V_j N_j} P(w_j), \forall j, j \neq i \\ &=> \frac{V_j}{V_i} > \frac{N_i}{N_j}\frac{P(w_j)}{P(w_i)} \end{aligned} \tag{22}$$

된다. 여기서 V_i는, 그룹 w_i에서 x에서 가장 가까운 k개의 샘플을 포함하는 체적이고, N_i는 그룹 w_i의 갯수이다.

3.2 탐지 모델 검증

검증 단계에서는 시험 데이터를 수식 (12)의 사영축에 사영시켜 1차원 데이터를 구한 후, 1차원 데이터를 kNN 기법에 적용하여 탐지 결과를 검증한다.

우선 수식 (28)의 데이터를 훈련 데이터와 시험 데이터로 수식 (23), (24)와 같이 각각 구분한다.

$$\begin{aligned} X_{tr} &= [X_{tr_1}, X_{tr_2}] \subset [X_1, X_2], \\ X_{tr} &\in I^{m \times (n_k 1 + n_k 2)} \end{aligned} \tag{23}$$

$$\begin{aligned} X_{te} &= [X_{te_1}, X_{te_2}] \subset [X_1, X_2], \\ X_{te} &\in I^{m \times (n_k 1 + n_k 2)} \end{aligned} \tag{24}$$

여기서 m은 속성의 개수이고, $n_{tr}1$, $n_{tr}2$는 학습 데이터의 악성코드(X_{tr_1})와 정상 코드(X_{tr_2})에 대한 데이터 개수이고, $n_{te}1$, $n_{te}2$는 시험 데이터의 악성코드(X_{te_1})와 정상 코드(X_{te_2})에 대한 데이터 개수이다.

데이터를 악성 코드 판별에 최적화된 1차원 데이터로 생성하기 위하여, GLDA 기법을 이용하여 구한 수식 (12)의 사영축에 X_{tr}, X_{te}을 수식 (25), (26)과 같이 사영시킨다.

$$\begin{aligned} Y_{tr} &= \{y_{tr_1}, y_{tr_2}, \ldots, y_{tr_n}\} = [Y_{tr_1}, Y_{tr_2}] \\ &= W^T \bullet [X_{tr_1}, X_{tr_2}] \in R^{1 \times n_k}, \\ &W^T \in R^{1 \times m}, n_{tr} = n_{tr}1 + n_{tr}2 \end{aligned} \tag{25}$$

$$\begin{aligned} Y_{te} &= \{y_{te_1}, y_{te_2}, \ldots, y_{te_n}\} = [Y_{te_1}, Y_{te_2}] \\ &= W^T \bullet [X_{te_1}, X_{te_2}] \in R^{1 \times n_k}, \\ &W^T \in R^{1 \times m}, n_{te} = n_{te}1 + n_{te}2 \end{aligned} \tag{26}$$

1차원의 시험 데이터(Y_{te})를 kNN 기법을 이용하여 판별하기 위해서 수식 (22)에 대입하여 설명하면 다음과 같다. 여기서 kNN 기법의 k가 3이고 거리의 척도로 euclidean distance를 사용한다. 특정 y_{te_i}를 정상 코드 또는 악성 코드로 분류하기 위하여, y_{te_i}와 모든 y_{tr_j} 사이의 euclidean distance 값을 수식

(27)과 같이 구한다.

$$d_{ij} = \sqrt{(y_{te_i} - y_{tr_j})^2}, \quad 1 \le i \le n_{te}, 1 \le j \le n_{tr} \tag{27}$$

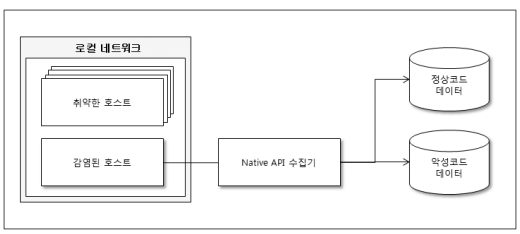
이 중 d_{ij}값이 가장 작은 3개의 y_{tr_j}를 선택한다. 선택된 y_{tr_j} 데이터가 가장 많이 속하는 Y_{tr_i}를 y_{te_i}의 클래스로 분류한다. 이후에 분류된 클래스가 원래의 Y_{te_i}에 속하는 클래스로 분류되었는지 검증한다.

IV. 실험 및 평가

제안된 탐지 기법의 전처리 과정이 효과적임을 입증하기 위하여, 윈도우 환경에서 악성 코드와 정상 코드가 발생시키는 Native API 데이터를 수집하고, 수집된 데이터만 사용한 탐지모델과 제안된 전처리 방법을 적용한 탐지모델을 비교 분석하였다. 또한 유닉스 환경의 frequency property 기법 중에 가장 높은 탐지율과 낮은 오탐율을 보인 Alock Sharma 가 제안한 탐지 기법인 SRBF(Smooth Radial Basis Function)〔15〕와 비교 분석하여, 본 논문에서 제안한 악성 코드 탐지 기법의 우수성을 검증하였다.

4.1 실험방법

본 논문에서 수집하는 데이터는 네트워크 기반 데이터가 아닌, 감염된 호스트에서 악성코드가 동작할 때 발생시키는 데이터만을 수집하기 때문에 대규모 네트워크를 구성하지 않고, 소규모 네트워크로 구성하여 실험을 진행하였다. 탐지 모델의 성능을 비교 분석하기 위하여 〔그림 1〕과 같은 환경에서 실험을 진행하였다. 악성 코드와 정상 코드의 Native API 데이터를 수집하기 위하여 Windows XP SP2가 설치된 호스트 5대로 구성하였다. 호스트 4대는 감염이 전파될 수 있는 취약한 호스트이고 호스트 1대는 악성 코드가 감염되어 동작하는 호스트이다.



〔그림 1〕 실험 환경

실험에서 사용할 악성 코드는 Offensive Computing〔25〕에서 제공하는 악성 코드로 실험하였다. classic virus, network worm, trojan 같은 다양한 악성 코드를 탐지 기법에 적용하기 위해서 Kaspersky lab〔26〕에서 제공하는 분류 기준에 근거하여 243개의 악성 코드로 실험하였고, 정상 코드는 Internet Explorer, Acrobat, MS WORD, HWP 등 84개로 실험하였다. 윈도우 환경에서 동작하는 악성 코드와 정상 코드가 발생시키는 Native API 데이터를 수집하였다. 그리고 Native API 속성의 개수는 총 468개이다.

4.2 데이터 수집 방법

악성 코드와 정상 코드에서 발생시키는 Native API 데이터를 수집하기 위해서 IDT(Interrupt Descriptor Table) 커널 데이터 가로채기 기법〔27〕을 이용한다. IDT는 인터럽트를 처리하는데 사용되는 테이블이다. 이 인터럽트 테이블에는 프로세스가 시스템 콜을 발생시킬 때 처리하는 루틴의 주소가 저장되어 있다. 이 주소를 변경하면 이에 대한 정보를 중간에 가로챌 수 있다. IDT 커널 데이터 후킹 기법을 사용하면 Native API 데이터를 수집하게 되면 시스템에서 동작하고 있는 모든 프로세스에 대한 정보가 기록된다. 해당 코드의 Native API 데이터를 추출하기 위하여 해당 코드의 프로세스 ID로 분류하였다. 프로세스 ID로 분류된 Native API 데이터를 각 Native API 속성 별 빈도수로 정리한다. 정리된 빈도수는 다시 각 프로세스별로 정규화한다. 즉, 아래의 다음의 과정을 거친다.

각각의 process i 가 발생시키는 속성 데이터를 x_i라 하고, 전체 API를 m개라 하면, j 번째 Native API 의 빈도수를 $x_i^t = (a_{i1}, ... a_{ij}, ... a_{im})$라 하면, 수집된 전체 데이터는

$$X = [x_1, \dots, x_n] = [X_1, X_2] \in I^{m \times (n1+n2)}, \quad 0 \le a_{ij} \le 1 \tag{28}$$

가 된다. 이때, 악성 코드(X_1)와 정상 코드(X_2)에 대한 데이터 개수가 각각 $n1$, $n2$ 이다. 수식 (28)에서 사용된 a_{ij}는 이미 정규화 된 회수이다. 즉, 각 pro-

cess i에서 발생한 회수가 b_{ij} 일때, $a_{ij} = \frac{b_{ij}}{\sum_{j=1}^{n} b_{ij}}$이다.

4.3 실험 결과

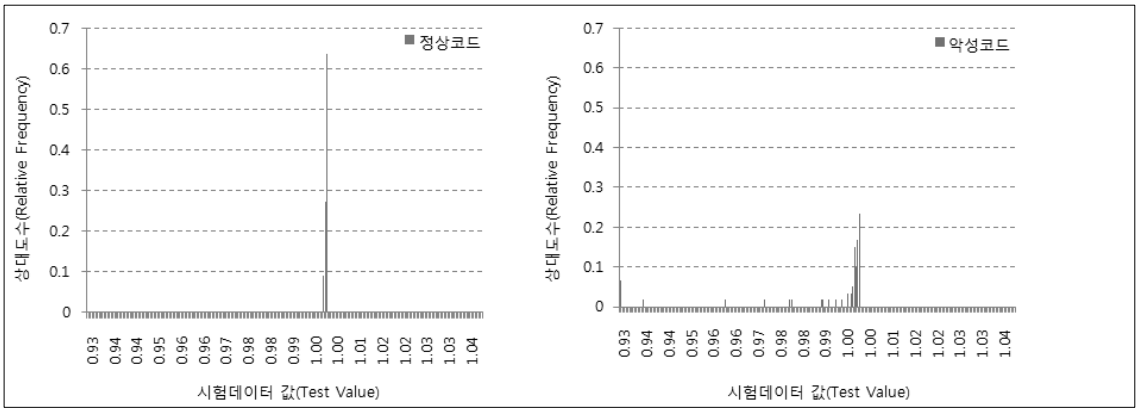
본 논문에서 제안한 전처리 과정을 검증하기 위하여 Native API 빈도수만을 이용하여 kNN 분류기로 분류한 것과 Native API 빈도수를 정규화한 후, GLDA 전처리 기법 적용한 데이터를 kNN 분류기로 분류한 것에 대하여 각각 탐지 결과를 구해서 비교하였다. kNN 분류기로는 Weka[28]에서 제공하는 분류기를 이용하였다. k 값은 3이고, 거리를 구하는 척도로는 Euclidean distance 알고리즘을 사용하였다. 제안된 전처리 과정을 검증하기 위하여 전체 수집된 데이터의 80%정도를 무작위 추출하여 학습데이터로 구성하고, 나머지 20%정도를 시험데이터로 구성하였다. 악성코드를 분류하기 위하여 악성코드 중 183개, 정상코드 중 73개로 학습데이터를 만들어 파라메터를 생성하고, 나머지코드를 시험데이터로 만들어 실험하였다. 시험데이터를 사용한 실험한 결과는 [표 1]과 같다.

[표 1] 빈도수 및 GLDA 탐지 결과의 Confusion Matrix

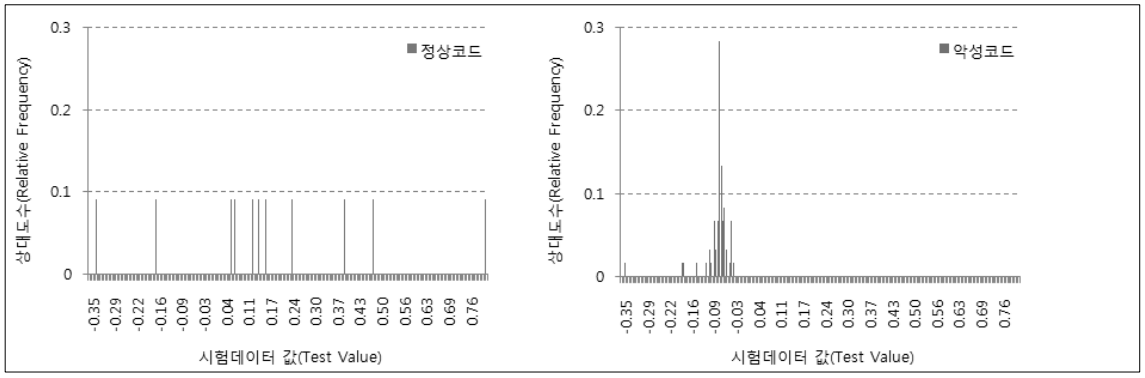
빈도수			GLDA		
실제 \ 예측	악성 코드	정상 코드	실제 \ 예측	악성 코드	정상 코드
악성코드	60	0	악성코드	60	0
정상코드	6	5	정상코드	2	9

Native API 데이터의 빈도수로 탐지 모델을 구축한 경우와 Native API 데이터의 빈도수를 정규화한 후, GLDA 전처리 기법을 적용하여 탐지 모델을 구축한 경우 모두 60개의 악성코드를 정확하게 탐지를 하였다. 하지만 빈도수만 가지고 탐지 모델을 구축한 경우에는 정상코드 6개를 악성코드로 분류한 반면, 빈도수를 정규화한 후, GLDA 전처리 기법을 적용한 탐지 모델은 정상코드 2개만을 악성코드로 분류하였다. [표 1]의 실험 결과로 Native API 데이터의 빈도수로 탐지 모델을 구축하는 것보다 Native API 데이터의 빈도수를 정규화한 후, GLDA 전처리 기법을 적용하여 탐지모델을 구축한 것이 오탐율을 줄이는데 효과적이라는 것을 실험으로 입증하였다.

제안된 탐지 기법을 검증하기 위한 마지막 단계로, frequency property 기법으로 가장 우수한 성능을 보인 Alock Sharma 의 SRBF기법과 본 논문에서 제안된 기법인, 데이터의 빈도수를 정규화한 후, GLDA를 사용한 탐지 모델을 비교 분석하였다. 탐지 기법을 비교 분석하기 위하여, 윈도우에서 수집한 동일한 Native API 데이터를 SRBF 탐지 기법과 본 논문에서 제안한 탐지 기법에 적용하였다. 제안된 탐지 기법은 학습데이터의 빈도수를 정규화 한 후, GLDA를 적용하여 악성코드와 정상코드를 구분하는 최적의 파라메터를 추출하여, 시험데이터에 추출된 속성을 사용하여, 변환한 1차원 데이터를 생성하였다. 또한 Alock Sharma 가 제안한 탐지 모델은 정상코드의 학습데이터와 시험데이터 간의 SRBF 값을 kNN 알고리즘을 적용하여 1차원 데이터를 생성하였다. 두 개의 탐지 기법을 비교하기 위하여, 1차원 데이터의 분포를 [그림 2], [그림 3]과 같이 나타내었다.



[그림 2] SRBF 데이터 분포



〔그림 3〕 GLDA 데이터 분포

〔그림 2〕는 정상 코드와 악성 코드를 SRBF 탐지 기법으로 만든 1차원 데이터의 분포를 나타낸다. 정상 코드와 악성 코드의 분포가 같은 값의 범위에 집중되어 있음을 확인할 수 있다. 정상 코드와 악성 코드의 분포가 대부분 겹치기 때문에, 이러한 탐지 기법을 적용하면 탐지율이 100%일 경우, 오탐율이 높을 수밖에 없다. 실제로 탐지율 100%인 Threshold 값이 0.9999 일 경우, 오탐율이 100%였다.

〔그림 3〕은 정상 코드와 악성 코드를 본 논문에서 제안한 탐지 기법으로 만든 1차원 데이터의 분포를 나타낸다. 정상 코드의 일부 데이터를 제외하고 정상 코드의 분포와 악성 코드의 분포가 다른 범위에 집중되어 있는 것을 확인할 수 있다. 실제로 탐지율 100%인 Threshold 값이 0.0644 일 경우, 오탐율이 18.2%였다.

위와 같은 결과로 본 논문에서 제안한 GLDA 기법이 SRBF 기법보다 최고의 탐지율에 도달했을 때, 낮은 오탐율을 보임을 확인하였다. 이러한 실험 결과를 통하여 제안된 탐지 기법이 윈도우 환경의 악성 코드를 탐지하는데 효과적임을 입증하였다.

V. 결 론

본 논문에서는 윈도우 환경에서 발생하는 악성 코드를 탐지하기 위한 높은 탐지율과 낮은 오탐율을 가진 탐지 모델을 제안하였다. 이 탐지 모델을 구축하기 위하여 악성 코드 또는 정상 코드에서 발생시키는 Native API 데이터의 빈도수를 구하였다. 그리고 본 논문에서 제안한 GLDA 전처리 방법을 적용하여, Native API 속성을 추출한 데이터로 탐지 모델을 구축하였다. 탐지 모델의 효과성을 비교하기 위하여 기존에 연구되었던 탐지 기법과 비교 검증하여 본 논문에서 제안한 탐지 기법의 효과성을 입증하였다. 제안된 탐지 기법에서는 Native API 속성을 전부 사용하지 않고 탐지에 최적화된 주요한 속성만 추출하여 사용함으로써 탐지 구조를 간소화 하였으며, 비교적 속도가 빠르고 대규모의 데이터를 처리할 수 있는 분류 알고리즘을 사용하여, 실제 탐지 시스템으로 구축하였을 때 발생할 수 있는 시스템 부하를 줄였다.

향후 연구로는 악성 코드를 실시간으로 탐지할 수 있게 하기 위한 성능적인 측면을 고려한 탐지 모델을 구축하고 또한 악성 코드와 정상 코드만을 탐지하는 것이 아니라 악성 코드로 탐지 되었으면 classic virus, network worm, trojan 과 같은 코드로 분류할 수 있는 탐지 모델을 연구하는 것이 있겠다.

참고문헌

〔1〕 C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," In Proceedings of the 6th International Workshop on the Recent Advances in Intrusion Detection, LNCS vol. 2820, pp. 173 - 191, Sep. 2003.

〔2〕 A.K. Ghosh, A. Schwatzbard, and M. Shatz, "Learning program behavior profiles for intrusion detection," Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring, vol. 1, Apr. 1999.

〔3〕 S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff, "A sense of self for

unix processes," Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 120-128, May 1996.
[4] S.J. Raudys and A.K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 3, pp.252-264, Mar. 1991.
[5] N. Ye, X. Li, Q. Chen, S. Emran, and M. Xu, "Probabilistic techniques for intrusion detection based on computer audit data," IEEE Transactions on System, vol. 32, no. 4, pp.266-274, Jul. 2001.
[6] M. Wang, C. Zhang, and J. Yu, "Native api based windows anomaly intrusion detection method using svm," Proceedings of the IEEE International Conference on sensor Networks, Ubiquitous and Trustworthy Computing, vol 1, Jun. 2006.
[7] N. Park, Y. Kim, and B. Noh, "A behavior based detection for malicious code using obfuscation technique," Journal of the Korea Institute of Information Security and Cryptology, vol. 16, no. 3, Jun. 2006.
[8] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," 1999 IEEE Symposium on Security and Privacy, pp. 133-145, May 1999.
[9] Q. Qian and M. Xin, "Research on hidden markov model for system call anomaly detection," PAISI 2007, LNCS vol. 4430, pp. 152-159, Apr. 2007.
[10] S. Radosavac and J.S. Baras, "Detection and classification of network intrusions using hidden markov models," 2003 Conference on Information Sciences and System, Mar. 2003.
[11] S. Cho and H. Park, "Efficient anomaly detection by modeling privilege flows using hidden Markov model," Elsevier Computers and security, vol. 22, no. 1, pp. 45-55, Jan. 2003.
[12] T. Kang, J. Cho, M. Chung and J. Moon, "Malware detection via hybrid analysis for api calls," Journal of the Korea Institute of Information Security and Cryptology, vol. 17, no. 6, pp. 89-98, Dec. 2007.
[13] Y. Liao and V. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Elsevier Computers and Seucurity, vol. 21, no. 5, pp. 439-448, Oct. 2002.
[14] S. Rawat, V.P. Gulati, A.K. Pujari, and V. Vemuri, "Intrusion detection using text processing techniques with a binary-weighted cosine metric," Journal of Information Assurance and Security, pp. 43-50, 2006.
[15] A. Sharma, A. Pujari, and K. Paliwal, "Intrusion detection using text processing techniques with a kernel based similarity measure," Elsevier Computers and Security, vol.26, no.7-8, pp. 488-495, Dec. 2007.
[16] G. Nebbett, "Windows nt/2000 native api reference," Macmillan Technical Publishing, 2000.
[17] D. Buckely, I. Altas, and J. Howarth, "A real time intrusion detection system for the windows environment," IADIS, 2007.
[18] I. Jolliffe, "Principal component analysis," Encyclopedia of Statistics in Behavioral Science, 2002.
[19] G.J. McLachlan, "Discriminant analysis and statistical pattern recognition," John Wiley & Sons, Inc., 2005.
[20] D.Q. Dai and P.C Yuen, "Regularized discriminant analysis and its application to face recognition," Pattern Recognition, vol. 36, pp. 845-847, 2003.
[21] R.P.W. Duin, "Small sample size generalization," Proc. Ninth Scandinavian conf. Image Analysis, vol. 2, pp. 957-964, Jun. 1995.
[22] J. Ye, R. Janardan, C. Park, and H. Park,

"An optimization criterion for generalized discriminant analysis on undersampled problems," IEEE Transactions on Pattern Recognition Analysis and Machine Intelligence, vol. 26, no. 8, Aug. 2004.
〔23〕 A. Papoulis, "Probability random variables and stochastic processes," 3rd Ed., MC-graw-HILL, 1991.
〔24〕 E. Parzen, "On the estimation of a probability density function and mode," Ann.Math, Stat. vol. 33, no. 3, pp. 1065-1076, Sep. 1962.
〔25〕 Offensive Computing, http://www.offensivecomputing.net
〔26〕 Kaspersky lab, http://www.viruslist.com
〔27〕 G. Hoglund and J. Butler, "Rootkits: subverting the windows kernel," Pearson Education Inc., 2006.
〔28〕 Machine Learning Project at the University of Waikato in New Zealand, http://www.cs.waikato.ac.nz/ml/

〈著者紹介〉

배 성 재 (Seon-jae Bae) 정회원
2005년 8월: 서울시립대학교 컴퓨터통계학과 학사 졸업
2012년 8월: 고려대학교 정보경영공학전문대학원 공학석사
〈관심분야〉 시스템 보안, 네트워크 보안, 침입탐지

조 재 익 (Jae-ik Cho) 학생회원
2005년 2월: 동국대학교 컴퓨터학과 학사 졸업
2008년 2월: 고려대학교 정보보호대학원 공학석사
2012년 8월: 고려대학교 정보보호대학원 공학박사
2012년 5월~현재: Samsung Electronics S-LSI 선임연구원
〈관심분야〉 무선/모바일 네트워크 보안, 무선 센서 네트워크, 이상탐지

손 태 식 (Tae-shik Shon) 정회원
2000년 2월: 아주대학교 정보컴퓨터공학부 공학사
2002년 2월: 아주대학교 정보통신전문대학원 공학석사
2005년 8월: 고려대학교 정보보호대학원 공학박사
2004년~2005년: Research Scholar, Univ. of Minnesota
2005년~2011년: Samsung Electronics DMC 연구소 책임연구원
2011년~현재: 아주대학교 정보컴퓨터공학부 조교수
〈관심분야〉 스마트그리드 보안, 디지털 포렌식, 비정상행위 탐지, 기계학습

문 종 섭 (Jong-sub Moon) 종신회원
1981년 2월~1985년: 금성 통신 연구소 연구원
1991년: Illinois Institute of technology 졸업(전산학 박사)
1993년~현재: 고려대학교 전자 및 정보공학부 교수
〈관심분야〉 생체인식, 침입탐지, 운영체제