공 학 석 사 학 위 논 문

개선된 AdaBoost를 활용한 상황정보 기반 보안 시스템 설계

2013 년 02 월

부 경 대 학 교 대 학 원

컴퓨터공학과

한 상 곤

공 학 석 사 학 위 논 문

개선된 AdaBoost를 활용한 상황정보 기반 보안 시스템 설계

지도교수 정 목 동

이 논문을 공학석사 학위논문으로 제출함

2013 년 2 월

부 경 대 학 교 대 학 원

컴퓨터공학과

한 상 곤

한상곤의 공학석사 학위논문을 인준함.

2013 년 2 월 22 일

주 심 이학박사 윤 성 대 (인)

위 원 이학박사 서 경 룡 (인)

위 원 이학박사 정 목 동 (인)

목차

그림 목차

표 목차

개선된 AdaBoost를 활용한 상황정보 기반 보안 시스템 설계

한상곤

부경대학교 대학원 컴퓨터공학과

요 약

기존의 컴퓨팅 환경과 비교해서 스마트 컴퓨팅 환경의 차이점은 사용자와 제품의 상호작용, 사용자 특성을 반영한 개인화된 서비스 제공, 자원의 제한적 활용으로 요약 할 수 있다. 그 중에서 사용자와 스마트 제품이 상호작용 하기 위해서는 사용자의 입력에 적절한 반응을 제공해야 한다. 스마트 제품이 사용자 입력에 적절한 반응을 제공하기 위해서는 사용자의 현재 상황에 대한 정보 수집과 분류가 선행되어야 한다.

본 논문에서는 사용자의 상황정보를 분류하기 위해서 AdaBoost 알고리즘을 개선한 AdaBoost.CA 알고리즘을 제안한다. 기존의 AdaBoost 알고리즘이 가진 이진 분류기의 장점 중 하나인 빠른 분류 속도를 살리면서, AdaBoost 알고리즘의 단점인 다중 분류가 불가능한 특성을 보완하여 다양한 범주의 상황정보를 분류할 수 있도록 개선하였다. 개선된 AdaBoost.CA 알고리즘은 상황정보를 빠르고 안정적으로 분류할 수 있다.

또한 AdaBoost.CA 알고리즘을 활용하여 상황정보 기반의 보안시스템을 설계 및 구현하였다. 상황정보 기반의 보안시스템 설계와 구현을 통해서 스마트 제품 및 기존의 컴퓨팅 환경에서

수집되는 다양한 상황정보를 분류하여 보안 시스템에 활용하는 방법을 제시하고, 보안 시스템의 성능을 비교 및 평가하였다. AdaBoost.CA 알고리즘을 활용한 상황정보 기반의 보안 시스템은 컴퓨터의 현재 상태를 분류하여 보다 안전한 보안 기법을 사용자에게 제공할 수 있고 사용자의 특성을 반영하여 보안 기법을 제공할 수 있으며 보안 시스템의 분류 정보를 통해서 시스템의 자원을 좀 더 효율적으로 사용 할 수 있다.

향후 연구를 통해서 사용자의 패턴을 분석하여 상황정보 보안 시스템을 활용하여 개별 사용자의 맞춤 보안을 제공할 수 있도록 사용자 패턴에 관한 연구와 상황정보 추론에 관한 연구가 병행되어야 한다.

Design of a Context-Aware Security System using Enhanced AdaBoost

Sangkon Han

Department of Computer Engineering, Graduate School,
Pukyong National University

Abstract

The differences between Smart computing and existing computing environment are as follows: the interaction of users and products, personalized service, the characteristics of the limited resources, and so on. To interact with Smart products and the user, Smart products should provide an appropriate response for the user's input. For the interaction between users and products, we should collect and classify the user's context-information.

We proposes an improved AdaBoost algorithm to classify the user's context-information. We proposes an improved AdaBoost algorithm to classify the user's context-information. AdaBoost.CA algorithm improves Multi-classification for the diverse context-information, while still preserving the property of the fast classification, one of the advantages of existing AdaBoost algorithm.

In this thesis, we also design and implement a security system based on context-information using AdaBoost.CA algorithm. Smart products and the existing computing environment, which is collected in a variety of classified context-information to evaluate and compare the performance of

the security system, and show how to take advantage of the security system. Security system based on context-information using AdaBoost.CA algorithm provides users with a more secure scheme to classify the current context of the computer. Security methods can provide security by applying the characteristics of the user. And through a system of information security classification system resources can be used more efficiently.

Future study is user patterns that are provided automatically by the security system using context-information. And Security system be able to inter from user context.

1. 서론

1.1 연구 배경

IT 기술의 급속한 발전과 함께 다양한 스마트 제품이 출시되고 있다. 스마트 제품은 기존 제품에 최신의 IT 기술이 통합되고 소형 운영체제와 다양한 소프트웨어인 앱(APP)을 통해서 사용자에게 부가적인 기능을 제공한다[1][2][3].

기존 PC 를 포함한 일반적인 컴퓨팅 환경과 스마트 제품을 포함한 스마트 컴퓨팅 환경의 차이점은 크게 세 가지로 나눌 수 있다. 첫째, 사용자와 스마트 제품의 상호작용이다. 기존의 컴퓨팅 환경은 외부 입력을 단순히 처리하는 환경이다. 반면 스마트 컴퓨팅 환경은 외부에서 주어지는 다양한 입력에 적절하게 반응한다. 그러므로 사용자가 스마트 제품에서 실행되는 지도를 터치했을 때 어떤 정보를 사용자에게 알려줘야 하는지 결정해야 하는 알고리즘이 CPU 의 속도보다 더 중요한 기술이 되었다[4]. 둘째, 개인화(personalization)이다. 기존의 컴퓨팅 환경은 사용자를 매우 포괄적이고 개념적으로 설정하였다. 그래서 일관된 디자인, 범용적인 기능을 중심으로 기술을 발전시켰다. 하지만 스마트 컴퓨팅 환경은 소형화 되고 휴대가 편리해지면서 사용자의 취향을 반영하는 쪽으로 기술이 발전하고

있다. 사용자에 대한 새로운 시각과 개인의 취향을 반영할 수 있는 기술은 맞춤형 서비스를 제공할 수 있는 환경을 제공한다[14]. 마지막으로, 자원의 제한적 활용이다. 스마트 제품은 다양한 기능을 제공하면서 크기는 작아지고 있다. 기능의 추가와 제품의 소형화는 이동성을 높였고, 배터리에 대한 의존도도 함께 높아졌다. 하지만 소형화된 크기에 비례해서 배터리의 용량도 작아졌다. 배터리 용량이 작아졌기 때문에 시스템의 가용 자원을 관리하고, 필요에 따라서 제한적으로 자원을 사용하기 위한 기술이 필수적이다[44].

기존의 PC 환경도 스마트 기기의 영향을 받아서 스마트 PC 환경으로 이행되고 있다. 사용자와 소통을 중시하고, 개인화를 서비스를 통해서 사용자 고유의 PC 환경을 제공한다. 이러한 컴퓨팅 환경 변화에 가장 주목해야 할 점은 컴퓨터와 사용자의 상호작용이 더욱더 중요해지고 있다는 점이다[5]. 따라서 기존의 PC 환경도 앞서 정의한 스마트 기기의 특성 중 제한된 자원을 제외한 두 가지 특징인 '상호작용'과 '개인화'가 중요한 기술로 논의되고 있다.

상호작용을 중시하는 컴퓨팅 환경에서 중요하게 고려되는 것은 '상황'과 '반응'이다[6]. 외부 반응에 즉각적으로 대응하기 위해서는 현재 상황을 정확하게 측정하여 사용자의 상황에 적절하게 반응해야 한다. 예를 들어서, 사용자가 지도를 표시하는 화면을 터치했을 때, 스마트 기기에 부착된 다양한 센서를 활용해서 기기가 작동되는 물리적 위치, 시간, 습도, 무선의 종류,

외부의 밝기, 주변 소음, 주변의 다양한 환경을 최대한 고려해서 사용자에게 가장 정확한 응답을 해야 한다. 그러므로 스마트 제품은 사용자의 입력에 반응하기 위해서 상황(context)을 정확하게 측정하고 평가 할 수 있어야 한다.

상황(context)에 관한 내용과 정의는 다양한다. 상황에 대한 용어를 처음으로 사용한 것으로 알려진 B. Shildt 와 M. Theimer[9]는 상황(context)의 정의를 장소, 주변의 사람, 객체의 집합 또는 시간에 따른 변화 등이라 정의하였다. Chen and Kotz[10]는 상황인식을 능동적(active) 상황인식과 수동적(passive) 상황인식으로 구분하여 정의하였다. 이러한 상황을 통합하여 현재 상황에 관한 정보의 집합을 '상황정보(context-information)이라 한다[11].

여러 가지 상황정보를 이용한 연구가 활발히 진행되고 있다. 외국의 대표적인 연구 중 하나는 Olivetti Research Ltd.사가 진행한 Call Forwarding 프로젝트이다. 이 프로젝트의 목표는 사용자에게 걸려온 전화를 현재 위치에서 가장 가까운 전화로 착신 시키는 서비스를 제공하는 것이다[12]. 그리고 스마트 제품을 위한 플랫폼 분야도 많은 연구가 진행되고 있다. 이 중에서 안정성, 견고성, 다양한 지원, 개방적 모델이라는 특징을 가진 구글의 개방형 운영체제인 안드로이드가 가장 활발히 연구되고 있다[13]. 스마트 제품의 플랫폼들은 뛰어나고 안정적인 성능으로 하드웨어와 응용 프로그램을 원활히 소통시켜주어 사용자에게 유용한 서비스를 제공할 수 있도록 한다[7].

그리고 상황정보 보안에 관련된 연구도 매우 중요하게 논의되고 있다. 스마트 제품에서 수집하는 상황정보는 사용자의 개인정보가 포함되어 있다. 현재 스마트 제품의 과도한 사용으로 개인정보 유출과 관련된 문제점이 제기되고 있다. 이와 관련되어 구글과 애플이 2011 년에 사용자 위치정보를 무단으로 수집하여 사회적으로 논란이 되었다[15]. 앞서 논의했듯이 상황정보는 개인화된 서비스를 제공하는데 매우 유용한 정보이다. 하지만 개인화된 서비스를 제공하는데 유용한 정보는 사용자 개개인의 맞춤 정보이기 때문에 유출되거나 수집되어서 곤란한 개인정보이다. 단순 상황정보는 의미를 가지지 못하지만 데이터가 모이고 분류되기 시작하면 개인의 생활 패턴을 유추할 수 있다. 상황정보를 활용한 연구와 함께 개인정보를 보호가 위한 보안 기술도 중요하게 논의되고 있다.

1.2 연구 범위 및 방법

다양한 상황정보를 이용하여 서비스를 제공하기 위해선 여러 가지 기술 요소들이 융합되어 사용된다. 주변 환경을 측정하기 위한 센싱 기술, 센싱을 통해 입력된 정보를 가공하고 분류 하는 기술, 전체 서비스를 위한 데이터 모델링 기술, 모델링 된 데이터를 교환하기 위한 교환 기술, 교환된 데이트를 서비스하기 위한 서비스 기술 등 굉장히 많은 기술들이 요구된다.

상황정보를 구성하는 다양한 기술 중에서 본 논문은 상황정보를 분류하는 것에 중점을 둔다. 즉, 가공되지 않는 정보(raw data)를 효율적이고 정확하게 분류하는 것을 목표로 한다. 상황정보의 효율적이고 정확한 분류를 위해서 AdaBoost 알고리즘을 개선하였다. AdaBoost 는 부스팅(Boosting)기법 중에서 단계를 반복하여 나온 여러 개의 약한 분류기와 가중치 값들의 조합으로 강한 분류기를 생성하여 두 개의 범주로 분류하는 혼합 분류기의 일종이다.

기존의 분류 알고리즘을 개선하여 상황정보를 분류하는데 적합한 알고리즘을 제안한다. 더 나아가서 분류된 상황정보를 활용하여 효율적인 보안방법을 제공하기 위해서 개선된 AdaBoos 알고리즘을 활용한 상황정보 기반의 보안 시스템을 설계하고 구현한다.

1.3 논문의 구성

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 통해서 상황과 상황인식, 혼합 분류기에 대해서 살펴본다. 3 장에서 분류기의 일종인 AdaBoost 를 상황정보 분류에 맞게 개선한 AdaBoost.CA 를 제안한다. 4 장은 AdaBoost 와 AdaBoost.CA 를 비교한다. 5 장에서는 AdaBoost.CA 를 활용한

상황인식 기반 보안 시스템 설계와 구현에 대해서 소개한다. 마지막 6 장에서 결론 및 향후 연구 방향을 논의한다.

2. 관련 연구

관련 연구에서는 상황(context), 상황정보(context-information), 상황인식(context-awareness)에 대해 살펴본다. 또한 대표적인 분류방법인 혼합 분류기법을 소개한다.

2.1 상황, 상황정보, 상황인식의 정의

2.1.1 상황(context)

상황에 대한 정의는 지금까지 다양한 개념으로 제시되었다. B. Shildt와 M. Theimer[9]는 상황의 정의를 "장소, 주변의 사람, 객체의 집합 또는 시간에 따른 변화 등"이라 정의하였다. 또 Pascoe[16]는 상황을 "특정한 관심이 가는 물리적이거나 개념적인 상태의 분류"로 정의하였다. Brwon[17]은 상황을 "사용자 환경을 구성하는 요소"로 정의하였으며, Franklin와 Flaschbart[18]는 "사용자가 놓여진 상황"으로 정의하였다. Dey et al.[19]는 상황을 "엔티티를 특징짓기 위하여 사용될 수 있는 모든 정보"라고 정의하였다. 엔티티란 사용자와 어플리케이션 사이에 상호작용과 관련된 인원, 장소, 객체, 사용자의 어플리케이션 자체도 포함된다. 이와 같이 상황에 대한 정의와 접근방법이 미세하고 상이하긴 하지만 본 논문에서는 상황을 다음과 같이 정의하여 사용한다.

"현재 상태를 개념적 혹은 물리적으로 특징화 시킬 수 있거나 인식 할 수 있는 개별적인 정보이다."

2.1.2 상황정보(context-information)

상황은 언제 반응해야 하는지 결정하는데 아주 중요한 역할을 한다. 예를 들어 스마트폰의 위치가 회사내부이고 소프트웨어 실행 시간이 업무시간이면 사용자 인증 과정에 대한 로그를 기록해야 하지만, 업무시간이 끝나고 위치가 집이면 사용자 인증 과정에 대한 로그를 기록하지 않아야 한다. 이러한 상황인식 과정에서 활용되는 모든 정보를 상황정보라 할 수 있다. 본 논문에서는 상황정보를 다음과 같이 정의한다.

"상황정보는 현재 상황을 나타내는 정보의 집합이다."

상황정보에 대한 분류는 다양할 수 있으나, B. Schilit et al.[20]은 상황정보를 컴퓨팅 상황, 사용자 상황, 물리적 상황으로 구분지어 설명하였고, Chen 와 Kotz[10]는 세 가지에 덧붙여서 시간 역시 중요한 상황정보라 주장하며 네 번째 범주를 제안하였다. 표 1 은 대표적인 상황정보의 범주와 속성을 나타낸다.

표 1. 상황정보의 범주와 속성

상황정보 분류를 제안한 연구자	상황정보의 분류	상황 정보의 속성
B. Schilit et al.[20]	컴퓨팅 상황	네트워크 연결 상태, 통신 비용, 통신 대역폭, 화면, 기기 자원량 등
	사용자 상황	사용자 프로파일, 현재 위치, 거주지, 과거 선호 지역 등
	물리적 상황	빛, 소음 정도, 온도, 습도, 날씨 등
Chen and Kotz[10]	시간 상황	시간, 일, 주, 월 등

2.1.3 상황인식(context-awareness

앞서 정의한 바와 같이 상황은 현재 상태를 개념적 혹은 물리적으로 특징화 시킬 수 있거나 인식 할 수 있는 개별적인 정보라면 상황인식은 상황을 기술적으로 특징화 시키는 방법이라 할 수 있다. 상황인식은 B. Shilit 와 M. Theimer[9]에 의하여 처음으로 정의되었으며, 상황에 적응적인 어플리케이션에 한정되어 사용된다.

상황인식이란 용어는 "적응적", "반응", "상응", "환경과 직접적인 관련이 있는" 등과 같은 단어와 밀접하게 관련되어 있다. 용어의 사용에서 알 수 있듯이 상황인식의 목적은 상황을 사용하는 것에 초점을 맞춘다고 할 수 있다.

Hull[21]은 상황인식에 대한 정의를 "센서 등을 이용하여 상황을 인식하고, 해석하여, 사용자의 환경에 적절한 반응을 할 수 있는 능력"이라고 정의하였다. Dey[22]는 상황인식을 "사용자와 컴퓨터간의 인터페이스"에 한정하여 사용하였으며, 이후에 Dey et al.[19]은 사용자의 상황에 대한 지식에 기반을 둔 소프트웨어를 자동화 작업에 적용하기도 하였다. 본 논문에서는 상황인식을 다음과 같이 정의한다.

"상황인식이란 상황을 기술적으로 특징화 시키는 방법이다."

2.1.4 상황인식시스템

B. Schilit et al.[20]는 상황인식시스템을 "장소, 주변 사람, 객체의 모임에 따라 적응적이고, 시간에 따른 변화를 수용할 수 있는 소프트웨어"로 정의하였다. 이후 상황인식시스템의 정의를 여러 차례 시도하였으나 대부분의 경우, 지나치게 특색있게 정의되었다. 최근에 상황인식시스템의 정의는 "사용자의 현재 위치, 시간, 주변에 있는 다른 사람이나 정보가전 기기들, 사용자의 행동 및 작업 이력 등과 같은 사용자의 현재 상황정보를 파악하고

분석하여 사용자가 현 상황에서 필요로 하는 서비스를 검색하여 구동시켜 주는 기술[8]"이라 할 수 있다. 본 논문에서 상황인식시스템은 다음과 같이 정의한다.

"상황인식시스템은 사용자가 알고자 하는 의미를 상황에 맞게 제공해주는 시스템이다."

상황인식시스템을 구축하기 위해서 상황인식을 위한 인식모델이 필요하다. 상황인식 모델은 다양한 방법으로 구현 가능하지만, 그 중에서도 가장 대표적인 인식 모델인 패턴 인식 방법을 활용한 상황 인식에 초점을 맞춘다.

패턴인식기술은 인식문제를 상황 분류(Classification) 문제로 다룬다. 이러한 상황 분류 문제는 인접 이웃 알고리즘(Nearest Neighbor), SVM(Support Vector Machine), 신경망(Neural Network)등의 방법으로 해결할 수 있다. 이러한 방법들은 사용하기 간단하고, 직접적으로 문제를 해결하는 방법들이다.

인접 이웃 알고리즘은 각각의 확률을 구하지 않고 그대로 좌표에 표시하여 참조집합(reference set) 에서 가장 유사(similar) 하거나 거리 상으로 가까운(nearest) 곳에 속하는 것으로 분류하는 방법이다[24].

SVM 알고리즘은 분류하기 힘든 다차원의 공간을 분류할 수 있는 고차원의 공간으로 변경하는 커널 함수와 관련이 있다. 고차원의 공간상에서 서로 다른 클래스에 속한 데이터들 간의 거리를 최대화 시키는 초평면을 구한다[25].

신경망 알고리즘은 각 층의 노드와 노드사이의 연결 가중치를 조정하여 복잡한 결정 문제를 풀기 위하여 사용된다[26].

2.1.5 상황인식의 응용

B. Schilit et al.[20]은 상황인식의 응용에 관하여 응용 분야를 네 가지로 분류하였다. 사용자 근처의 객체를 쉽게 파악할 수 있도록 도와주는 인터페이스 관련 응용으로 Proximate selection, 상황에 따라 컴포넌트를 제거하거나 컴포넌트간의 연결을 도와주는 것과 관련된 응용으로 Automatic contextual reconfiguration, 결과를 산출하는 상황정보가 영향을 미치는 응용으로 Contextual information and commands, 상황인식 시스템이 어떻게 변경돼야 하는지 명시하는 데 사용되는 단순한 규칙인 Context-triggered actions 으로 분류하였다.

Pascoe[27]는 상황 인식의 기본적인 응용을 상황 수집, 상황 적용, 상황 자원 발견과 상황 확대로 나타내었다. 상황수집(Contextual Sensing)은 다양한 환경 상태를 인식하고 시스템에 반영한다. 상황적용(Contextual Adaptation)은

수집된 데이터를 기반으로 시스템을 현재 상황에 적용한다. 상황자원 발견(Contextual Resource Discovery)은 사용 가능한 자원을 발견하고 이것을 시스템에 연결하는 것이다. 상황증대(Contextual Augmentation)는 상황에 사용자의 디지털 데이터를 결합하는 것으로 정의 하였다.

Dey 와 Abowd[28]는 상황인식 응용을 세 개의 일반적인 범주로 묶었다. 정보 표현과 서비스(Presenting Information and Services)는 상황정보를 사용자에게 보여주거나 상황을 사용자에 대한 적절한 서비스를 제안하기 위해서 사용하고, 서비스 자동실행(Automatically Executing a Service)은 변화에 따라 사용자를 대신해서 명령을 내리거나 시스템을 재조정하는 작업으로, 상황정보 추가 (Attaching Context Information) 는 적절한 상황정보를 추가하여 차후 검색을 용이하게 하는 것(tagging)으로 정의하였다.

Chen 와 Kotz[10]는 그들이 내린 상황의 정의를 바탕으로 수동적인 응용과 능동적인 응용으로 구분하여 정의하였다. Active context awareness 는 새로운 상황이 발견되면 자동으로 동작을 변경하는 응용이고, Passive context awareness 는 새로운 상황에 관심을 갖는 사용자에게 제공하거나 차후의 서비스를 위해서 유지하는 응용으로 정의하였다.

상황인식의 응용에 관한 다양한 정의를 통해서 상황인식 서비스가 지원해야 하는 요구사항을 명확히 이해할 수 있으며, 서비스 제공자는

상황인식 서비스를 합리적으로 제공할 수 있으며, 개발자는 요구사항을 통해 응용 서비스를 쉽게 구현할 수 있게 된다. 표 2 는 상황인식의 응용을 위한 분류를 정리한 것이다.

표 2. 상황인식 응용에 관한 연구들

상황인식 응용을 제안한 연구자	상황인식의 응용 분야
B. Schilit et al.[20]	● Proximate selection : 인터페이스 관련 응용 ● Automatic contextual reconfiguration : 컴포넌트간의 연결과 관련된 응용 ● Contextual information and commands : 상황정보가 결과에 영향을 주는 응용 ● Context-triggered actions : 변경규칙을 명시하는 단순한 규칙들
Pascoe[27]	● 상황 수집(Contextual Sensing) : 상황를 인식 및 반영 ● 상황 적용(Contextual Adaptation) : 현재 상황에 적용 ● 상황 자원 발견(Contextual Resource Discovery) : 시스템에 자원을 연결 ● 상황 증대(Contextual Augmentation) : 추가 데이터 결합
Dey and Abowd[28]	● 정보 표현과 서비스(Presenting Information and Services) : 사용자에게 서비스를 보여주거나 제안하기 위해서 사용 ● 서비스 자동실행(Automatically Executing a Service) : 자동으로 명령을 내리거나 시스템을 재조정하는 작업 ● 상황정보 추가 (Attaching Context Information) : 검색을 용이하게 상황정보를 추가함(tagging)
Chen and Kotz[10]	● Active context awareness : 상황정보가 변경되면 자동으로 동작을 변경하는 응용 ● Passive context awareness: 상황정보가 변경되어도 서비스를 유지하고, 관심있는 사용자에게 알려줌

2.2 혼합 분류기

2.2.1 혼합 분류기의 기본적인 개념

혼합 분류기의 기본적인 개념은 여러 개의 가설에서 얻어지는 결과를 조합하여 보다 정확한 결과를 출력하는 것이다. 그림 1 은 가설들의 기본적인 혼합 방법을 나타낸 것이다. 각 가설들은 훈련데이터에 의하여 생성되며, 새로운 입력에 대하여 가설들은 각각의 결과를 생성한다. 이 결과들을 조합하여 더욱 정확한 결과를 생성하게 된다. 일정한 조건하에서 단일 가설들을 혼합하는 방법은 단일 가설을 혼합하지 않고 사용하는 경우보다 우수한 성능을 보인다.

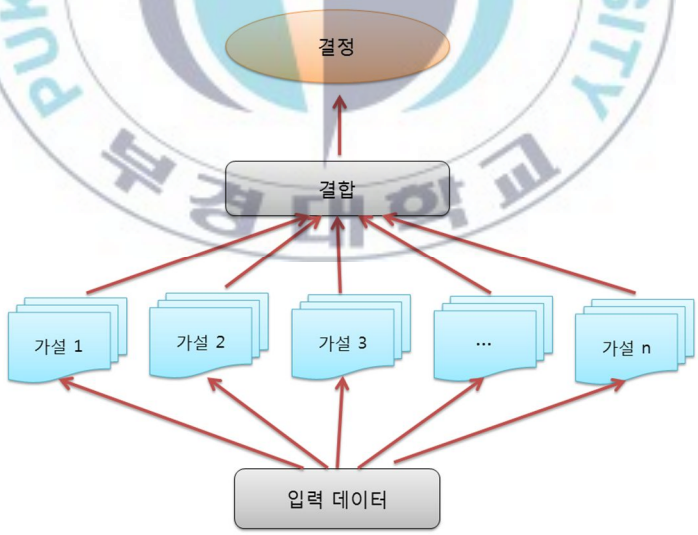

그림 1. 기본적인 혼합 분류기 모델

혼합 분류기 학습 방법은 위와 같은 개념을 도입한 것으로 하나의 분류기를 학습하는 방법이 아니라 여러 개의 분류기를 가진 하나의 분류기 집합을 학습하는 방법이다. 분류기 집합 내에 속한 각각의 단일 분류기는 적절한 조합을 통하여 최적화된 성능을 발휘하는 하나의 강인한 혼합 분류기를 생성하게 된다. 혼합 분류기는 단일 분류기에서 얻어지는 결과를 이용하여 최종 결과를 결정하는 방식이다. 일반적으로 혼합 분류기는 일정한 조건에서 단독 분류기 보다 우수한 성능을 보이고 있다.

2.2.2 대표적인 혼합 분류 방법 : 배깅(Bagging)과 부스팅(Boosting)

패턴인식 분야에서 가장 많이 사용되고 있는 알고리즘은 배깅과 부스팅 방법이다. 배깅과 부스팅 방법은 혼합 분류기 중 좋은 성능을 보여주는 대표적인 혼합 분류기 방법이다.

배깅(bagging) 알고리즘은 "bootstrap"과 "aggregating"의 조합어로서 단일 분류기를 훈련시키기 위하여 사용되는 훈련데이터집합을 동일하게 사용하지 않고, 원래 훈련 데이터에서 무작위로 추출하여 새로운 훈련데이터집합을 구성하여 사용한다. 각 분류기는 서로 다른 훈련데이터 집합을 사용하게 됨으로 인하여 서로의 다양성을 포함할 수 있다.

부스팅 알고리즘은 배깅 알고리즘과 달리 무작위로 데이터를 추출하여 훈련데이터를 생성하지 않고, 분류가 올바르게 되기 힘든 데이터에 가중치를 부여하여 분류기로 하여금 해당 데이터에 대하여 분류를 더욱 잘 할 수 있도록 유도한다.

부스팅의 기계학습(machine learning) 연구에서 이론적인 시작은 Valiant[29]의 PAC(probably approximately correct) 학습모델(learning model)이다. 이 학습모델은 학습기가 대충 맞는(approximately correct) 가설(hypothesis)을 확률적으로 찾아내면 학습(learning)이 성공했다고 본다. 그리고 Kearns and Valiant[30]는 약한 분류(weak classification) 알고리즘을 처음으로 제안하였다. Kearns and Valiant 의 알고리즘은 PAC 학습모델에 있어서 단순추측(random guessing)하는 것보다 약간 더 분류를 잘하는 약한 분류자(weak classifier)는 강 분류(strong classifier) 알고리즘으로 강화(boosted) 될 수 있음을 보여주었다.

이후, Schapire[31]는 약한 분류자들을 부스팅을 통해 강 분류자로 변형할 수 있음을 수학적으로 보였다. 그 후에 Freund[32]는 더 효율적인 부스팅 알고리즘을 제안하였다. 부스팅 알고리즘을 처음으로 실제 데이터에 적용한 것은 Druker et al.[33] 에 의해 수행한 OCR(optical character recognition) 프로젝트이다.

2.2.3 AdaBoost(Adaptive Boosting) 알고리즘

Yoav Freund 와 Robert E. Schapire[34]에 의해 처음 소개된 AdaBoost(adaptive boosting)는 이전의 부스팅 알고리즘의 문제점과 어려움을 대부분 해결하였다. 이후 Schapire 와 Singer[35]에 의해 AdaBoost 는 일반화 되었다.

부스팅 알고리즘은 순차적으로 분류자를 생성하고, 분석용 데이터는 이전 분류자의 실행을 관찰하여 추출된 개별적인 관측값을 사용한다. 초기 분석용 데이터 관측값의 가중치는 모두 동일한 값으로 초기화한다. 분류자에 의해 오분류된 관측값은 높은 가중치로 설정하고, 제대로 분류된 관측값은 낮은 가중치를 부여한다. 이렇게 관측값들의 가중치 재조정을 통해서 새로운 분석용 데이터가 만들어진다. 분석용 데이터가 형성될 때 가중치가 높은 관측값이 많이 선택됨으로써 분류가 힘든 관측값을 잘 분류되도록 한다. Robert E. Schapire[36]에 의하면 분석 오차(training error)는 지수로 표현이 가능하기 때문에 분석 오차를 가장 빠르게 감소시킨다. 표 3 은 Yoav Freund 와 Robert E. Schapire[34]에 의해 소개된 AdaBoost 알고리즘이다.

표 3. AdaBoost 알고리즘

단계 1	입력 M 개의 데이터 : $(x_1, y_1), \ldots, (x_m, y_m),\ x_i \in X,\ y_i \in Y = \{-1,1\}$	
단계 2	초기 분포 D_1을 구한다. : $\mathrm{D}_1(\mathrm{i}) = 1/\mathrm{m}$	
단계 3	(3-1, 3-2, 3-3)의 과정을 T 번 반복한다. : for t=1,…,T	
	단계 3-1	ϵ_t가 가장 작은 $\mathrm{h_t}$를 구한다. : $\epsilon_\mathrm{t} = \mathrm{Pr_{i\sim D_t}}[\mathrm{h_t(x_i)} \neq \mathrm{y_i}]$
	단계 3-2	가중치를 계산한다. : $\alpha_\mathrm{t} = \frac{1}{2}\ln(\frac{1-\epsilon_\mathrm{t}}{\epsilon_\mathrm{t}})$
	단계 3-3	분포D_{i+1}를 갱신한다. : $D_{i+1} = \frac{\mathrm{D_t(i)}}{\mathrm{z_t}} \times \begin{cases} \mathrm{e}^{-\alpha_\mathrm{t}} \text{ if } \mathrm{h_t(x_i)} = \mathrm{y_i} \\ \mathrm{e}^{\alpha_\mathrm{t}} \text{ if } \mathrm{h_t(x_i)} \neq \mathrm{y_i} \end{cases}$
단계 4	(단계 3)에서 T 번 반복하여 최종 분류자 : $H(x) = \mathrm{sign}[\sum_{\mathrm{t}=1}^{\mathrm{T}} \alpha_\mathrm{t}\mathrm{h_t}(x)]$를 구성한다.	

표 3 은 AdaBoost 알고리즘이다.

단계 1 에서는 학습 데이터를 준비하는 과정이다. 학습 데이터는 총 m 개이며 x_i 는 입력 데이터, y_i 는 입력 x_i 의 소속 범주를 나타낸다. 소속 범주는 $Y = \{-1,1\}$로 두 개의 클래스로 구성된다

단계 2 에서는 초기 분포를 구한다. 초기 분포는 1 을 입력 데이터의 개수로 나눈 값($\frac{1}{m}$)이다. 분포 D의 값은 계속해서 수정된다.

단계 3 은 T 회 반복하면서 분포 D를 갱신한다.

단계 3-1 은 오분류율(ϵ_t)이 가장 낮은 약한 분류기 h_t를 설정한다. ϵ_t는 m 개의 입력 데이터를 계산하여 가장 에러가 작은 값이다.

단계 3-2 은 구해진 ϵ_t로 분포 D를 갱신하기 위하여 가중치를 계산한다. ϵ_t가 작을수록 그 값이 커져서 오분류율이 작은 분류기가 더 큰 중요도를 갖도록 정의되어, 이후 최종 판별함수 생성에서 분류기를 결합하는데 중요한 지표로 사용된다.

단계 3-3 에서는 분포 D의 값을 갱신한다. 잘 분류된 데이터는 가중치를 낮춰 적용하고 잘 분류되지 못한 데이터에 대해서는 가중치를 높여서 적용한다.

단계 4 에서 $H(x)$는 최종 분류기이다. $H(x)$는 T 반복을 하는 동안 구해진 가중치 α_t와 h_t의 선형적인 결합으로 구해진다.

3. AdaBoost 알고리즘의 개선

기존의 AdaBoost 알고리즘을 본 논문에서 제안하는 상황정보 기반의 보안시스템에 적용하기 위해서 AdaBoost 알고리즘을 개선 할 필요가 있다. 왜냐하면 기존의 AdaBoost 알고리즘은 다양한 범주로 분류할 수 없고, 다중범주를 지원하는 알고리즘은 상황정보 분류에 적합하지 않기 때문이다[37]. 예를 들어, 다중 범주 분류를 지원하는 AdaBoost 알고리즘인 AdaBoost.MH[37] 알고리즘은 분류되는 결과가 '-1'과 '1'사이의 실수 값으로 분류되기 때문에 입력되는 상황정보에 비해서 분류 결과가 너무 넓게 분포된다[38].

본 논문에서는 기존의 AdaBoost 알고리즘이 가지는 단점을 보완하고 상황정보 기반의 보안시스템에 적용할 수 있도록 개선된 AdaBoost 알고리즘을 AdaBoost.CA(AdaBoost for Context-Aware) 알고리즘을 제안한다.

3.1 AdaBoost 알고리즘의 단점

AdaBoost 알고리즘은 '약한 분류기' 혹은 '단순 분류기'를 조합하여 '강한 분류기'를 생성하는 Boosting 알고리즘의 일종이다. AdaBoost 는 Paul Viola 와 Michael Jones 의 논문을 통해 얼굴 검출에 사용되면서 빠른 얼굴 검출 시간과 정확도 높은 검출 능력으로 많은 주목을 받았다[39].

하지만 본 논문에서 제안하는 상황정보 기반의 보안시스템에 기존의 AdaBoost 알고리즘을 적용하게 되면 상황정보 분류에 몇 가지 단점이 생긴다. 첫째, AdaBoost 알고리즘은 이진 분류(two-class)를 위해서 고안된 알고리즘이기 때문에 상황정보를 다양하게 분류하지 못한다. 둘째, 약한 분류기를 강한 분류기로 만들기 위해서는 많은 데이터가 필요하고 데이터가 부정확하거나 잡음이 동반되면 분류기의 성능이 급격하게 떨어진다. 또한 분류기가 너무 약하거나 분류기가 너무 복잡할 경우에도 문제가 발생한다[40]. 이러한 단점을 보완하고 AdaBoost 알고리즘과 비슷한 성능을 낼 수 있는 알고리즘을 꾸준히 연구하고 있다[36][41][42].

그러므로 다양한 상황정보를 분류할 수 있어야 하고 분류기는 단순해야 한다. 따라서 본 논문은 기존의 AdaBoost 알고리즘을 상황정보 기반의 보안시스템에 적합하도록 알고리즘을 개선하였다.

AdaBoost 알고리즘의 개선사항은 다음과 같다.

첫째, 입력되는 상황정보의 종류가 다양하기 때문에 입력되는 데이터를 미리 범주화해서 입력하는 것이다. 입력 데이터를 미리 범주화해서 분류하는 장치를 중간 분류기라 한다. 다시 말해서, 중간 분류기는 입력 데이터의 종류에 따라 분류된 데이터의 집합이다. 각각의 중간 분류기는 입력되는 상황정보를 분류하여 결과 벡터(vector)로 출력한다.

예를 들어서, 시간 정보와 위치 정보는 강한 연계성을 가지고 있다. 반면, 네트워크 사용량과 위치 정보는 연계성이 상대적으로 낮다. 그런데 이런 정보를 여러 개의 약한 분류기를 통해서 강한 분류기로 분류하게 되면 입력되는 상황정보를 올바로 해석 할 수 없다. 또한 주기적인 정보와 비주기적인 정보를 계속해서 분류하게 되면 강한 분류기 생성을 위해서 소모적인 자원을 낭비하게 된다. 이러한 자원 낭비는 시스템 전반에 영향을 주게 된다. 중간 분류기를 통해서 시간 정보와 위치 정보는 같은 범주에서 분류하고, 네트워크 사용량은 따로 분류한다. 이 때, 중간 분류기는 두 개가 생성된다. 중간 분류기의 목적은 상황정보를 제대로 활용하기 위해서 동일 범주의 상황정보를 데이터가 입력될 때 미리 분류하는 것이다.

둘째, 입력되는 데이터의 변화가 없다면 중간 분류결과에 반영하지 않도록 한다. 상황정보는 시스템에서 주기적으로 발생하는 신호를 포함해서, 비주기적인 사용자 위치 정보, 네트워크 접속 정보 등이 포함되어 있다. 즉, 입력 데이터의 종류가 다양하고, 입력되는 시점이 불확실하다. 그리고 상황정보 중에서 거의 변화가 없는 상황정보의 경우에는 실시간으로 상황정보를 분류하지 않아도 된다. 따라서 상황정보를 적절히 처리하기 위하여 입력된 상황정보가 이전 상황정보와 동일할 경우 계산을 최소화 해야 한다.

3.2 AdaBoost.CA 알고리즘

표 4 는 상황정보 기반의 보안시스템에 적용할 수 있도록 AdaBoost 를 개선한 AdaBoost.CA 알고리즘이고 각 단계는 다음과 같다.

단계 1 에서는 기존의 AdaBoost 알고리즘과 동일하게 학습 데이터를 준비하는 과정이다. 학습 데이터는 총 m 개이며 x_i는 입력 데이터, y_i는 입력 x_i 의 소속 범주를 나타낸다. 소속 범주는 $Y = \{-1,1\}$ 로 두 개의 클래스로 구성된다. z_i는 입력 데이터가 소속될 중간 분류기이다.

단계 2 에서는 초기 분포를 구한다. 초기 분포는 1 을 입력 데이터의 개수로 나눈 값($\frac{1}{m}$)이다. 분포 D 의 값은 계속해서 수정된다. 그리고 중간 분류기에 따라서 분포값은 모두 다르다. 분포 D 의 값이 높을수록 분류에 부적절한 데이터이다.

단계 3 은 T 회 반복하면서 분포 D를 갱신한다.

단계 3-1 은 오분류율(ϵ_t)이 가장 낮은 약한 분류기 h_t를 구하는 과정이다. 약한 분류기는 식(1)과 같은 형태로 입력 x 에 대하여 분류 범주 $\{-1, 1\}$ 를 분류해 주는 간단한 형태의 분류기이다.

$$h_t \in \{-1, 0, +1\},\ h_t = z_i \tag{1}$$

ϵ_t는 약한 분류기에 대하여 m 개의 입력 데이터를 조사하여 가장 에러가 작은 ϵ_t를 구하는 과정이다. 초기 오분류율은 모든 데이터가 같은 가중치를 가지기 때문에 동일한 오분류율을 가지고 있다. ϵ_t는 0.5 보다 작아야 하며 이것은 부스팅을 위한 약한 분류기의 조건이다.

모든 약한 분류기는 자신이 속한 중간 분류기에 포함되어야 한다. 모든 약한 분류기가 중간 분류기에 포함되기 위한 조건은 R_k 가 R 에 관한 분할이어야 한다. 분할이란 공집합이 아닌 임의의 집합 R_k 는 서로소이고, 공집합이 아닌 R_k 의 부분집합으로 나눈 것을 R 에 관한 R_k 의 분할(partition)이라 한다. 즉, R_k는 분할 $\{R_1, R_2, R_3, \dots, R_k\}$은 다음과 같은 성질을 만족한다.

(1) $k = 1,2,\dots,n$에 대하여 R_k는 공집합이 아닌 집합 R의 부분집합이다.

(2) $R_k = R_1 \cup R_2 \cup R_3 \cup \dots \cup R_n$

(3) $i \neq j$이면 $R_{t,i} \cap R_{t,j} = \phi$

중간 분류기는 분할을 만족하고, 약한 분류기는 중간 분류기에 포함된다. 따라서 중간 분류기는 R_k에 유일하게 하나이며, 모든 중간 분류기는 개별적인 분류 속성을 지닌다.

단계 3-2 에서는 약한 분류기의 오분류률이 0 이면 반복과정을 종료한다. 오분류될 확률이 0 이라면 이전에 분류된 정보와 동일한 정보가 지속적으로 입력됨을 뜻하기 때문이다.

단계 3-3 은 구해진 ϵ_t로 분포 D를 갱신하기 위하여 가중치를 계산한다. 여기서 α_t는 약한 분류기 h_t의 가중치를 나타내는 값으로 만약 ϵ_t가 0.5 보다 작다면 α_t는 커지고, ϵ_t가 0.5 보다 클수록 α_t는 작아진다($\alpha_t \propto \frac{1}{\epsilon_t}$). 특히 ϵ_t가 0.5 보다 작다는 것은 난수(random) 분류기보다 분류 성능이 높을 때를 말하며, ϵ_t 이 작을수록 그 값이 커져서 오분류율이 작은 분류기가 더 큰 중요도를 갖도록 정의되어, 이후 최종 판별함수 생성에서 분류기를 결합하는데 중요한 지표로 사용된다.

단계 3-4 에서는 분포 D의 값을 갱신한다. 잘 분류된 데이터는 가중치를 낮춰 적용하고 잘 분류되지 못한 데이터에 대해서는 가중치를 높여서 적용한다. 다음 반복에서의 약한 분류기 h_t를 선택함에 있어서 잘 분류되지 못한 데이터를 분류하여 높은 가중치 값을 피하고 제일 작은 ϵ_t 값을 가진 약한 분류기에게 우선순위를 주는 것이다. Z_t 는 분포 D 를 정규화 시키는 값으로 식(2)과 같다.

$$Z_t = \sum_{i=1}^{m} D_{t+1,i} \quad (2)$$

단계 4 에서 $M(x)$는 중 분류기이다. $M(x)$는 T 반복을 하는 동안 구해진 가중치 α_t 와 h_t 의 선형적인 결합으로 구해진다. 여기서 sign 은 signum function 으로 식(3)과 같다.

$$\mathrm{sign}(x) = \begin{cases} 1, \text{if } x > 0 \\ -1, \text{if } x < 0 \end{cases} \quad (3)$$

표 4. 개선된 AdaBoost 알고리즘

단계 1	입력 $M \subset R_k$개의 데이터 : $(x_1, y_1, s_1), \ldots, (x_m, y_m, s_m),\ x_i \in X,\ y_i \in Y = \{-1,1\}, s_i \in R_k = \{1,2,\ldots,k\}$	
단계 2	초기 분포 D_{1,s_m}을 구한다. : $D_{1,s_m}(i) = 1/m$	
단계 3	(3-1, 3-2, 3-3)의 과정을 T 번 반복한다. : for $t = 1, \ldots, T$	
	단계 3-1	ϵ_t 가 가장 작은 h_t 를 평면 S_m 에서 구한다. : $\epsilon_t = \Pr_{i \sim D_t, s_m}[h_{t,s_m}(x_i) \neq y_i]$
	단계 3-2	만약 h_t가 0 이면 반복과정을 종료한다.
	단계 3-3	가중치를 계산한다. : $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
	단계 3-4	분포$D_{i+1,k}$를 갱신한다. : $D_{i+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t(x_i) = y_i \\ e^{\alpha_t} \text{ if } h_t(x_i) \neq y_i \end{cases}$
단계 4	(단계 3)에서 T 번 반복하여 중간 분류자 $M(x) = \mathrm{sign}[\sum_{t=1}^{T} \alpha_t h_t(x)]$ 를 구성한다.	

3.3 AdaBoost.CA 분류 실험

AdaBoost.CA 분류 실험은 인텔 i5-3.1Ghz CPU 와 DDR3 16GB 를 장착한 데스크탑에서 진행하였다. S/W 는 MATLAB 2012a 버전을 사용하였고, 분포 및 분산을 계산하기 위해서 MATLAB 에서 지원하는 통계 패키지인 Statistics Toolbox v8.0 을 사용하였다

입력 데이터가 2 차원 좌표일 때 AdaBoost.CA 알고리즘을 이용하여 어떻게 입력 데이터를 부스팅하여 학습하게 되는지 실험해 본다. 실험의 편의성을 위해서 임의적으로 입력되는 상황정보의 분류를 4 가지로 고정한다. 따라서 2 차원 평면을 사분면으로 나누면 가장 적당하기 때문에 (0,0)을 기준으로 전체 평면을 직선으로 분할한다. 그림 2 는 2 차원 좌표에 25 개의 데이터를 무작위로 입력하였다. 실험을 위해서 1 사분면에 집중적으로 분포되도록 하였다. 평면 좌표에는 두 종류의 입력 데이터('+' 기호, '*' 기호)가 입력된다.

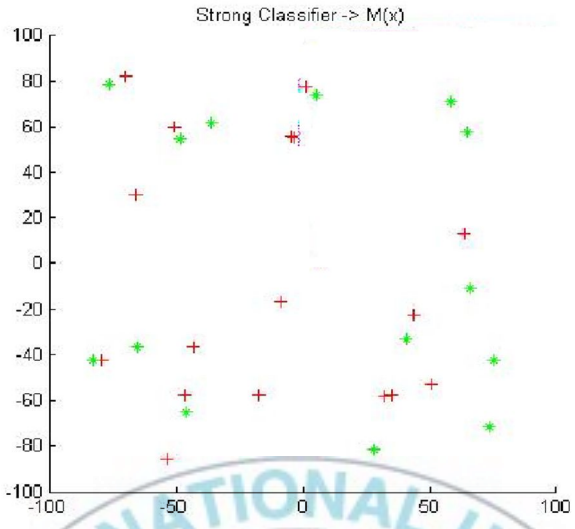

그림 2. 무작위로 입력된 25 개의 좌표

가장 먼저 입력 데이터를 분류하기 위해서는 약한 분류기를 생성하여야 한다. 표 5 는 1 사분면에 입력된 데이터를 나타낸다. 입력 데이터는 각각의 좌표와 소속 분류로 구성되어 있다. 입력된 데이터가 8 개 이므로 약한 분류기 생성을 위해서 약한 분류기는 가로, 세로로 나눌 수 있다. 가로, 세로로 나눌 수 있는 직선 7 씩 총 14 개의 직선이 사용되고, 클래스 분류를 위한 경우의 수를 고려해서 총 28 개의 약한 분류기를 생성한다.

표 5. AdaBoost.CA 분류 실험을 위한 입력 데이터

x 좌표	26.023	88.011	66.959	19.005	66.374	53.508	34.79	64.035
y 좌표	47.66	23.099	57.017	78.070	80.994	38.888	30.116	6.725
소속범주	-1	-1	-1	-1	-1	1	1	1

T 회 반복 후 중간 분류자를 만들기 위해서 초기 분포를 초기화 하고, 38 개의 약한 분류기중에서 에러 값이 가장 작은 분류기를 선택한다. 아래 표 6 은 첫 번째 반복에서 28 개의 약한 분류기의 에러 값을 나타낸 것이다. 첫 번째 반복에서 가장 좋은 분류기는 21 번째 분류기이다.

표 6. 첫 번째 반복(T=1) 후 약한 분류기 결정

1	0.500	8	0.250	15	0.2500	22	0.875
2	0.500	9	0.750	16	0.750	23	0.250
3	0.625	10	0.375	17	0.375	24	0.750
4	0.375	11	0.625	18	0.625	25	0.375
5	0.500	12	0.500	19	0.250	26	0.625
6	0.375	13	0.500	20	0.750	27	0.500
7	0.625	14	0.500	21	0.125	28	0.500

첫 번째 반복에서 선택된 약한 분류기는 21 번째 이며, $\epsilon_t = 0.125000$이며, $\alpha_t = 0.972955$이고, $Z_t = 0.661438$ 이다. 첫 번째 반복 후 분포의 값은 아래 표 7과 같다.

표 7. 첫 번째 반복(T=1) 후 분포 값

0.0714	0.5000	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714

이러한 반복과정을 거쳐서 ϵ_t와 h_t를 결정하여 중간 분류자 $M(x)$를 결정하면 그림 3 ,그림 4, 그림 5, 그림 6과 같은 형태로 분류된다.

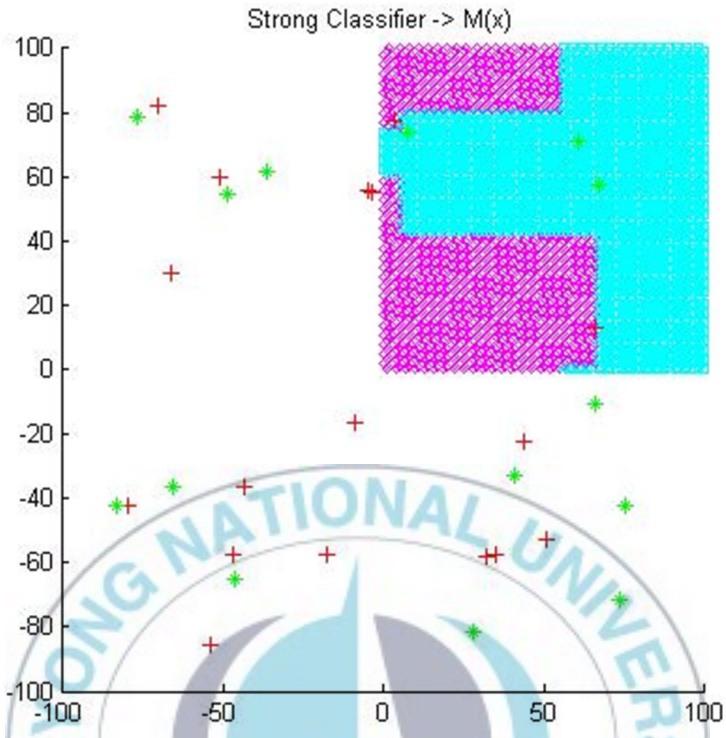

그림 3. AdaBoost.CA 를 활용한 1 사분면 분류

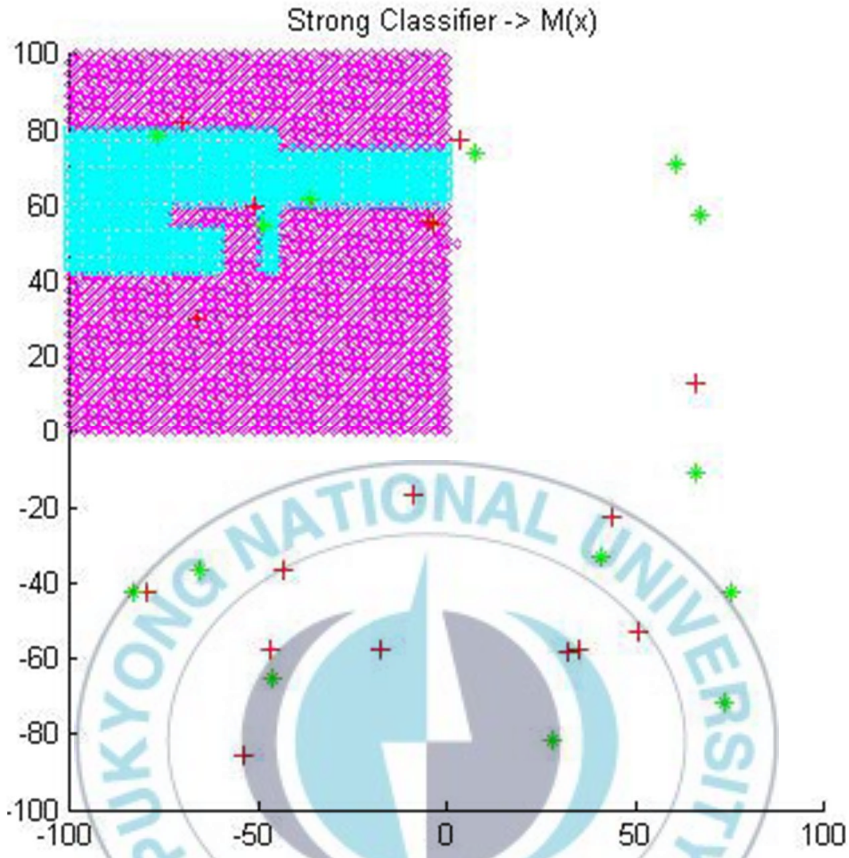

그림 4. AdaBoost.CA 를 활용한 2 사분면 분류

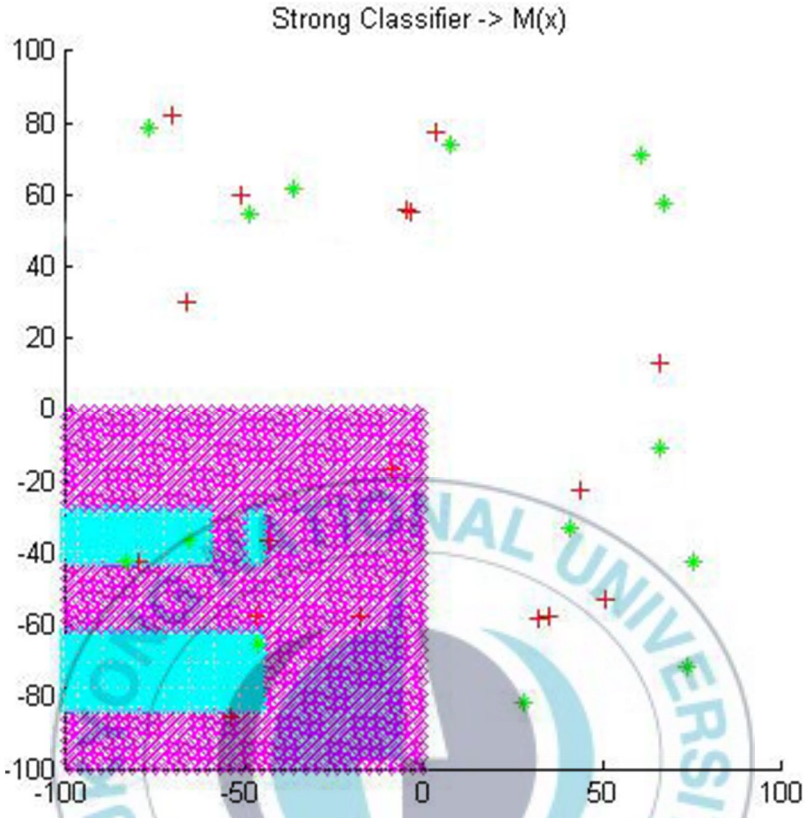

그림 5. AdaBoost.CA 를 활용한 3 사분면 분류

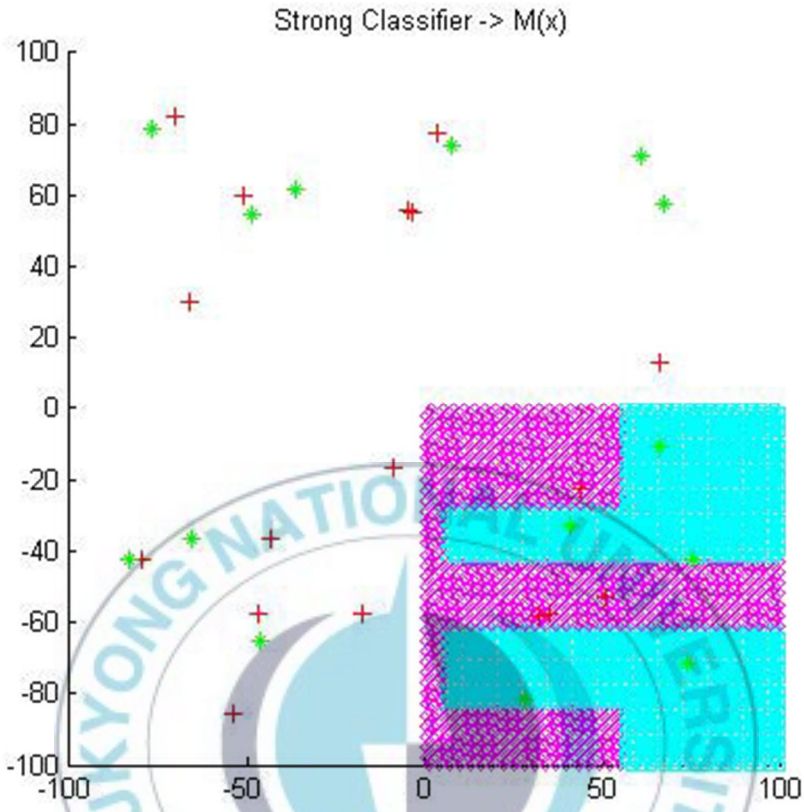

그림 6. AdaBoost.CA 를 활용한 4 사분면 분류

4. AdaBoost 와 AdaBoost.CA 비교

AdaBoost 와 AdaBoost.CA 를 비교하기 위한 시스템 사양은 CPU 는 인텔 i5-3.1Ghz, RAM 은 DDR3 16GB 를 장착한 PC 에서 수행되었다. SW 는 MATLAB 2012a 버전을 사용하였고, 분포 및 분산을 계산하기 위해서 MATLAB 에서 지원하는 통계 패키지인 Statistics Toolbox v8.0 을 사용하였다

AdaBoost 와 AdaBoost.CA 의 인식률을 비교하기 위해서 분류기를 10 개, 50 개, 100 개를 무작위로 생성하여 비교하였다. 각각 10 회 반복해서 나온 수치 중 가장 높은 수치와 가장 낮은 수치를 제외한 8 개의 수치를 평균해서 그래프로 나타냈다.

아래 그림 7, 그림 8, 그림 9 는 부스팅 횟수에 따른 오인식률(error rating)을 나타낸다. 가로축은 약한 분류기의 갯수를 의미하고, 세로축은 오인식률이다. 그래프에서 '+'는 기존의 AdaBoost 이고, '□'는 본 논문에서 제안한 AdaBoost.CA 이다.

그림 7 은 약한 분류기가 10 개 일 때 오인식률이다. AdaBoost.CA 는 기존의 AdaBoost 에 비해서 초기 오인식률이 높다. 그리고 AdaBoost 가 약한 분류기가 5 개 정도만 되어도 오인식률이 낮아지는데 반해서, AdaBoost.CA 는 약한 분류기가 10 개 정도가 되어야 AdaBoost 와 비슷한 수치를 보여주고 있다.

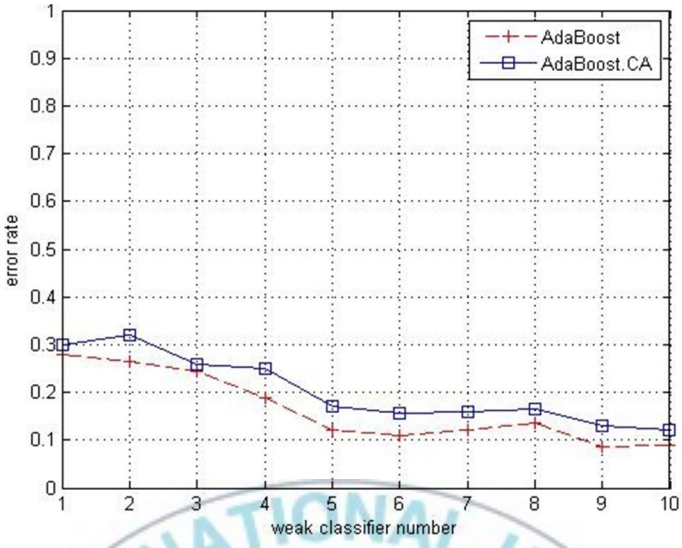

그림 7. 분류기 10 개의 오인식률.

그림 8 은 분류기가 50 개 정도로 늘어나게 되면 고른 오인식률을 보여주고 있다. 기존의 AdaBoost 에 비해서 매우 안정적인 분류가 가능함을 알 수 있다.

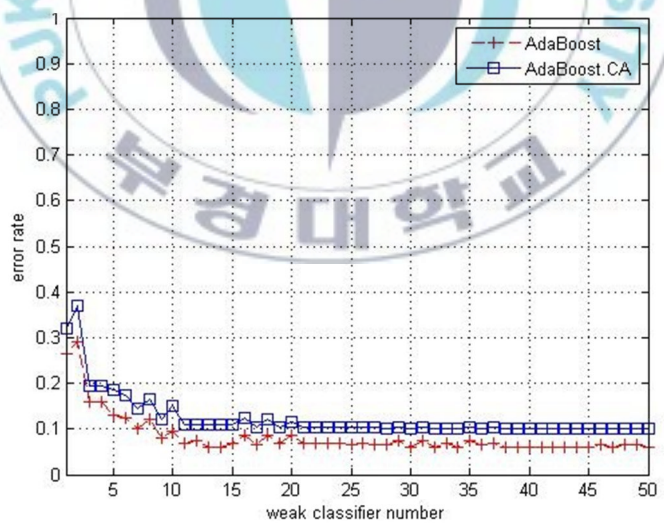

그림 8. 약한 분류기 50 개 일 때 오인식률

마지막으로 그림 9 는 분류기가 100 개 정도로 많아지면 오인식률이 하나의 값으로 수렴하는 것을 볼 수 있다. AdaBoost 의 경우 중간 과정에서 오인식률이 주기적으로 높아지는 경향성을 보이고 있다. 반면에 AdaBoost.CA 의 경우 매우 균등한 분포를 보이고 있다.

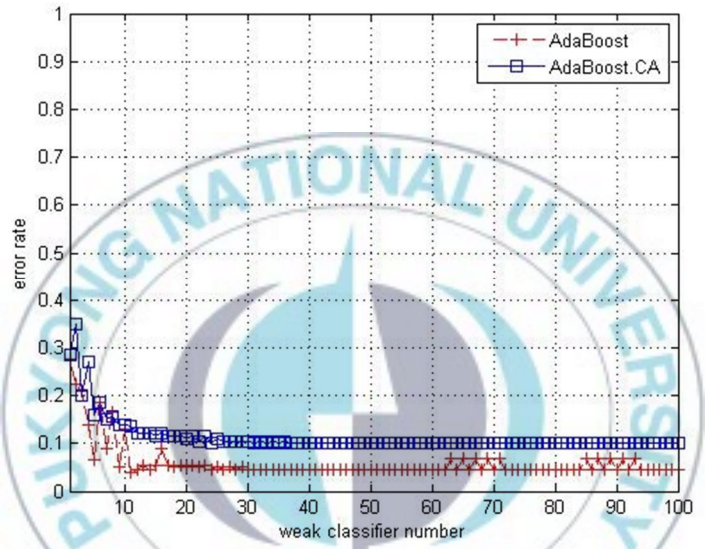

그림 9. 약한 분류기 100 개 일 때 오인식률

5. AdaBoost.CA 를 활용한 상황인식 기반 보안 시스템 설계

다양한 보안 기법을 사용하기 위해서는 시스템에서 기본적으로 지원해야 되는 것들이 있다. 예를 들어, 얼굴인식을 제공하기 위해서는 전면 카메라나 후면 카메라가 부착되어 있어야 한고, 음성인식을 위해서는 마이크가 장착되어 있어야 한다. 다양한 보안 기법을 활용하기 위해서는 컴퓨팅 상황, 사용자 상황, 물리적 상황 등을 종합적으로 고려해서 현재 상황에 가장 적합한 보안 기법을 선택해야 한다. 현재 상황을 고려해서 보안 기법을 제공하게 되면 사용자 편의성을 높일 수 있으며, 보안성도 증가시킬 수 있다. 그리고 시스템 상태를 반영할 수 있기 때문에 시스템 자원을 좀 더 효율적으로 사용할 수 있다.

5.1 개선된 AdaBoost 를 활용한 상황정보 기반 보안시스템의 개요

기기에 부착된 센서, 하드웨어, 소프트웨어는 다양한 상황정보를 생성하고 수집한다. 수집된 상황정보는 분류기에 의해서 분류된다. 분류된 상황정보를 분석하여 보안등급을 결정한다. 결정된 보안등급은 관리자가 설정한 보안정책과 연동하여 사용자에게 현재 상황에 가장 적합한 보안기법을 제공한다. 그림 10 은 개선된 AdaBoost 를 활용한 상황정보 기반의 보안 시스템의 개요를 나타내고 있다.

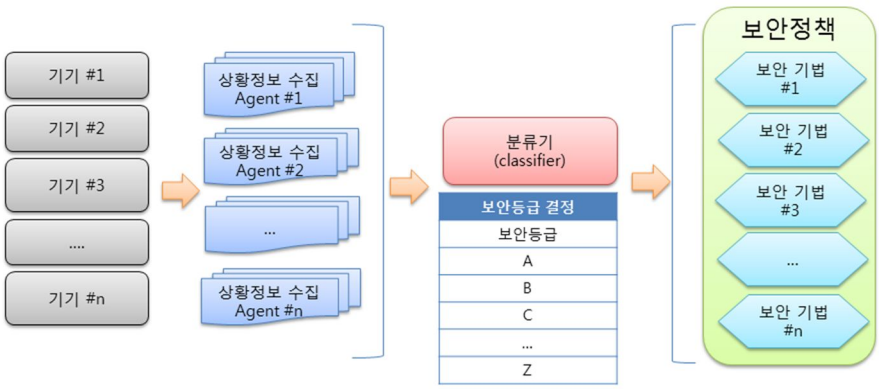

그림 10. AdaBoost.CA 를 활용한 상황인식 기반 보안 시스템의 개요

5.2 개선된 AdaBoost 를 활용한 상황정보 기반 보안시스템의 구성

개선된 AdaBoost 를 활용한 상황정보 기반의 보안시스템의 구성은 Client 와 Server 로 나뉜다. Client 는 사용자 기기에 설치되며, Server 는 Client 의 상황정보를 저장하고 분석하는 것이다. 그림 11 은 Client 와 Server 의 세부적인 구성을 나타낸다.

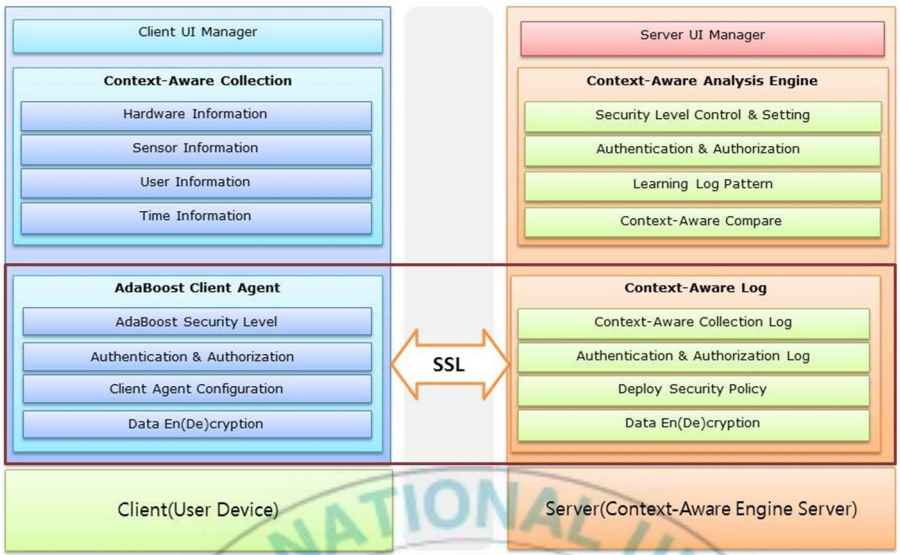

그림 11. 상황정보 기반 보안 시스템의 Client 와 Server 구성도

5.2.1 Client 와 Server 의 통신

Client 와 Server 는 SSL 을 통해서 연결된다. SSL 통신이란 'Secure Socket Layer'의 줄임말로 넷스케이프사에서 전자상거래등과 같은 인터넷 통신 보안을 목적으로 개발되었다. 이후 SSL 이 많이 사용되자 IETF(Internet Engineering Task Force)에서 TLS(Transport Layer Security)라는 이름으로 표준화되었다. SSL 은 특히 전송계층(Transport Layer)의 암호화 방식이기 때문에 HTTP 뿐만 아니라 NNTP, FTP, XMPP 등 응용계층(Application Layer) 프로토콜의 종류에 상관없이 사용할 수 있는 장점이 있다. 기본적으로

Authentication, Encryption, Integrity 를 보장한다[43]. 그림 12 는 SSL Record Protocol 의 구조이다.

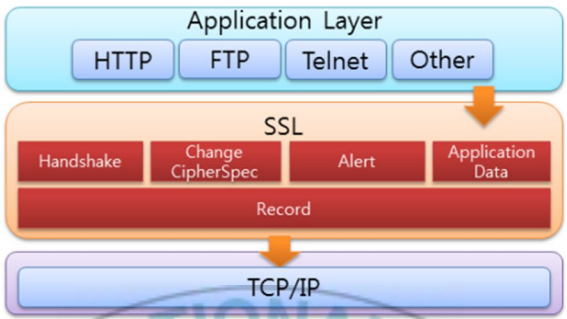

그림 12. SSL Record Protocol

5.2.2 Client 의 세부 구성

Client 는 Client UI Manager, Context-Information Collection, AdaBoost.CA Client Agent 로 구성된다..

'Client UI Manager'는 현재 수집되고 있는 상황정보와 서버와의 연결 상태 및 보안 설정을 사용자에게 알려준다. 사용자는 UI 를 통해서 상황정보를 제공받을 수 있다.

'Context-Information Collection'는 상황정보를 수집하고 저장한다. 'Hardware Information'은 CPU, RAM, 네트워크 등과 같이 하드웨어에 관한 정보를 저장한다. 'Sensor Information'은 GPS, 온도 센서, 습도 센서, 방향

센서 등과 같이 기기의 부착된 외부 센서정보를 통해서 입력되는 정보를 저장한다. 'User Information'은 사용자 ID, 허가된 보안등급, 접근권한 등과 같이 사용자 정보를 저장한다. 'Time Information'은 기기 사용 시간, 로그인/로그아웃 시간 등과 같이 시간관 관련된 정보를 수집한다. 표 8 은 Context-Information Collection 의 세부 구성을 표로 정리한 것이다.

표 8. Context-Information Collection 의 세부 구성

Hardware Information	CPU, RAM, Network 등과 같은 하드웨어 정보 수집.
Sensor Information	GPS, 온도, 습도, 방향 등과 같은 외부 센서 정보 수집.
User Information	ID, 사용자 보안등급, 접근권한 등과 같은 사용자 정보 수집.
Time Information	현재시간, 기기 사용 시간, 로그인/로그아웃 기록등과 같은 시간 관련 정보 수집.

'AdaBoost.CA Client Agent'는 수집된 상황정보를 분석하여 현재 상황에 적절한 보안기법을 제공하기 위한 상황정보 분류 및 처리 모듈이다. 'Security Level'은 AdaBoost.CA 를 이용해서 현재 상황을 분류한다. 현재 상황을 분류하는 기준은 서버에서 결정한 보안정책을 참고해서 분류한다. 'Server

Connection'는 Server 와 연결하고 연결상태를 관리한다. 'Client Agent Configuration'는 서버로부터 전송된 보안정책을 시스템에 적용한다. 'Data En(De)cryption'은 파일의 암/복호화를 지원한다. 보안 정책과 관련된 설정 파일은 매우 중요하기 때문에 암호화되어 전송된다. 또한 사용자 기기에 저장된 파일들을 보호하기 위해서 암/복호화 기능이 필요하다. 표 9 는 AdaBoost.CA Client Agent 의 세부 구성을 표로 정리한 것이다.

표 9. AdaBoost.CA Client Agent 의 세부 구성

Security Level	AdaBoost.CA 를 이용해서 현재 상황에 적합한 보안 등급을 결정한다.
Server Connection	Server 와 연결하고 연결 상태를 관리한다.
Client Agent Configuration	보안 정책에 관련된 설정 파일을 시스템에 적용한다.
Data En(De)cryption	파일의 암/복호화를 지원한다.

5.2.3 Server 의 세부 구성

Server 의 구성은 Server UI Manager, Context-Information Server Agent', Client, Context-Information Analysis Engine 으로 구성된다.

'Server UI Manager'는 관리자가 손쉽게 관리할 수 있도록 사용자 관리에 필요한 정보를 확인하고 설정 할 수 있다.

'Context-Information Server Agent'은 사용자의 상황정보를 보관하고 분석한다. 수집된 상황정보는 사용자 행동 분석 및 사용자 업무 분석 등에 사용한다. 'Context-Information Log'는 상황정보를 보관하고, 'Authentication & Authorization Log'는 사용자의 인증 정보 및 보안 정책에 관련 정보를 수집한다. 'Deploy Security Policy'는 보안 정책을 배포한다. 'Data En(De)cryption'는 보안 정책을 배포할 때 필요한 암/복호화를 제공한다. 표 10 은 Context-Information Server Agent 의 세부 구성을 표로 정리한 것이다.

표 10. Context-Information Server Agent 의 세부 구성

Context-Information Collection Log	사용자 상황정보를 보관한다.
Authentication & Authorization Log	사용자 인증 관련 정보를 보관한다.
Deploy Security Policy	관리자가 설정한 보안정책을 배포한다.
Data En(De)cryption	기본적인 보안기법을 제공한다.

'Context-Information Analysis Engine'은 관리자가 보안 정책을 설정하고, 수집된 상황정보를 분석한다. 분석된 상황정보는 관리자가 보안 정책을 수립하거나 보안 상태를 점검하기 위한 자료로 관리자에게 제공한다. 'Security

Policy Setting'은 관리자가 보안 정책을 결정 할 수 있다. 'Learning Context-Information Pattern'은 개별 사용자의 상황정보를 학습한다. 'Analysis Context-Information'은 사용자의 상황정보를 학습한 것을 활용하여 사용자의 현재 상태를 분석하여 관리자에게 보고한다. 표 11 은 Context-Information Analysis Engine 의 세부 구성을 정리한 것이다.

표 11. Context-Information Analysis Engine 의 세부 구성

Security Policy Setting	관리자가 보안 정책을 결정하고 설정한다.
Learning Context-Information Pattern	로그 정보를 분석한다.
Analysis Context-Information	사용자의 상황정보를 분석하여 관리자에게 보고한다.

5.3 개선된 AdaBoost 를 활용한 상황정보 기반의 보안시스템의 구현

5.3.1 구현 환경

개선된 AdaBoost 를 활용한 상황정보 기반의 보안시스템은 PC, Server, 스마트폰에서 실행 할 수 있도록 구현하였다. Client 프로그램은 PC 에서 구현하였다. PC 사양은 CPU 가 인텔 i5-3.1Ghz, RAM 은 DDR3 4GB 가

장착되어있고, 외부센서는 장착되어 있지 않다. Server 에서는 Server 프로그램은 Server 에서 구현하였다. Server 사양은 CPU 가 인텔 i7-3.9Ghz, RAM 은 DDR3 64GB 가 장착되었고, SSD 가 Raid-0 으로 동작한다. PC 와 마찬가지로 외부센서는 장착되어 있지 않다. 스마트폰은 노키아에서 출시한 Lumia 710 에서 구현되었다. Lumia 710 은 ARM Single Core 기반의 1.4Ghz CPU 가 사용되었고, RAM 은 DDR2 512M 를 사용할 수 있다. GPS 와 Front-Camera 가 장착되어있고 3G(WCDMA) 통신이 가능하다. 표 12 는 구현에 사용된 시스템 사양을 정리한 것이다.

구현에 사용된 프로그램은 마이크로소프트사에서 출시한 VisualStudio 2010 을 사용해서 구현하였고, 구현 언어는 C#으로 선택하였다. 스마트폰은 WindowsPhone SDK v7.1.1 을 추가로 설치하였다. Client 와 Server 통신에 사용되는 SSL 프로토콜은 마이크로소프트에서 제공하는 것을 사용하였다.

표 12. 구현 시스템 사양

시스템 종류 / 시스템 항목	PC	Server	스마트폰(Lumia 710)
CPU	Intel i5-2400 3.1Ghz	Intel i7-3770 3.9Ghz	ARM Core 1.4Ghz
RAM	DDR3 4GB	DDR3 64GB	DDR2 512MB
Sensor	장착되어 있지 않음	장착되어 있지 않음	GPS, Front-Camera,
특이사항	없음	SSD (Raid-0)	WCDMA 지원

5.3.2 Client 프로그램의 구현

Client 프로그램은 'Context-Information Collection' 모듈과 'AdaBoost.CA Client Agent' 모듈로 구성된다. 그림 13 은 PC 에서 Client 프로그램을 실행한 화면이다.

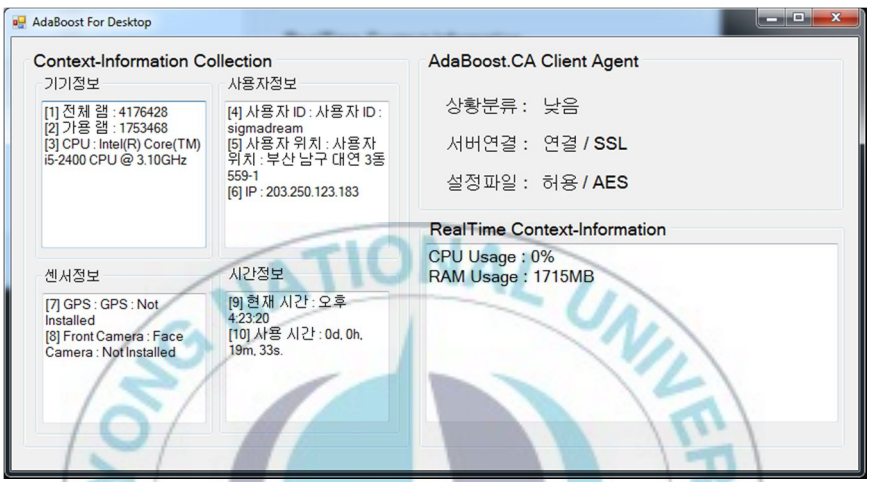

그림 13. PC 용 Client 프로그램

'Context-Information Collection'은 총 10 개의 상황정보를 수집하도록 구현하였다. '기기정보'는 장착된 램의 전체 용량, 현재 사용 가능한 램의 용량, CPU 성능을 수집한다. '센서정보'는 GPS 를 사용한 위치 정보와 전면 카메라 장착 여부를 수집한다. '사용자 정보'는 사용자 ID, 사용자 위치, , 네트워크 IP 주소를 수집한다. '시간정보'는 현재 시간과 Client 프로그램의 사용시간을 수집한다. 'CPU Usage'와 'RAM Usage'는 실시간으로 CPU 와 RAM 사용량을 실시간으로 파악한다. 표 13 은 'Context-Information Collection'이 수집하는 상황정보 목록이다.

표 13. 본 논문에서 구현한 Client 프로그램에서 수집하는 상황정보 목록

[1] 전체 램	시스템에 장착된 물리적인 램의 크기
[2] 가용 램	현재 운영체제에서 사용할 수 있는 램의 크기
[3] CPU	CPU 제조사와 속도
[4] 사용자 ID	현재 사용자의 접속 정보
[5] 사용자 위치	현재 사용자의 접속 위치
[6] IP	기기에 할당된 IP 주소
[7] GPS	GPS 센서를 통해서 입력된 좌표의 위치
[8] Front Camera	얼굴 인식 및 동작인식을 위한 전면 카메라
[9] 현재 시간	현재 시스템의 시간
[10] 사용시간	사용자가 현재 시스템을 사용한 시간
CPU Usage	CPU 의 실시간 사용률
RAM Usage	RAM 의 실시간 사용률

'AdaBoost.CA Client Agent'는 세 가지 주요한 기능을 담당한다. '상황분류'는 백그라운드에서 서비스로 동작하고 있는 AdaBoost.CA 알고리즘을 사용해서 현재 상황를 분류한다. 보안 정책을 참고해서 보안 등급을 결정한다. '서버연결'은 Server 와 연결 상태를 나타낸다. '설정파일'은 관리자가 배포한 보안정책 파일의 적용 여부와 설정 파일을 보호하기 위해서 사용된 암호화 알고리즘 종류를 화면에 출력한다. 설정파일은 XML 로 구성되어 있다. 보안 정책에 사용되는 XML 파일에 대한 자세한 사항은 5.3.3 Server 프로그램의 구현의 '보안 정책 설정'에 자세히 설명하였다.

5.3.3 Server 프로그램의 구현

Server 'Context-Information Server Agent' 모듈과 'Context-Information Analysis Engine' 모듈로 구성된다. 그림 14 는 Server 프로그램을 실행한 화면이다.

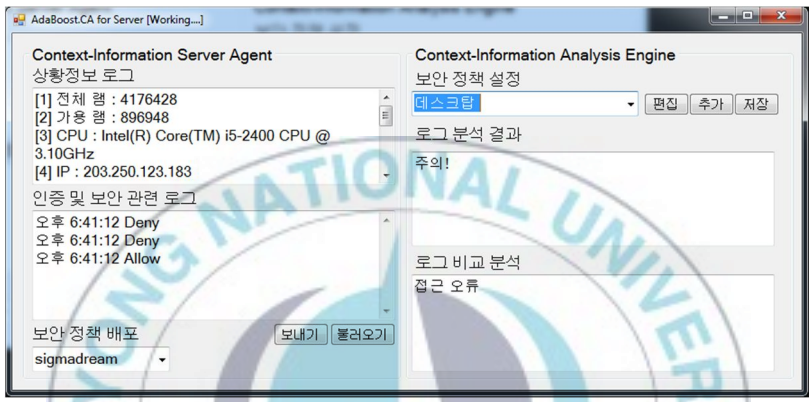

그림 14. Server 프로그램 구현

'Context-Information Server Agent'는 총 두 개의 세부 기능을 수행한다. '상황정보 로그'는 접속한 사용자의 상황정보를 수집하여 저장한다. '인증 및 보안 관련 로그'는 사용자 인증 결과와 보안 관련 정보를 수집하여 저장한다.

'Context-Aware Analysis Engine'은 세 가지의 세부 기능을 수행한다. '보안정책 설정'은 시스템의 보안 정책을 설정한다. 예를 들어서, 데스크탑, 모바일 등과 같이 보안 정책을 설정 할 수 있다.

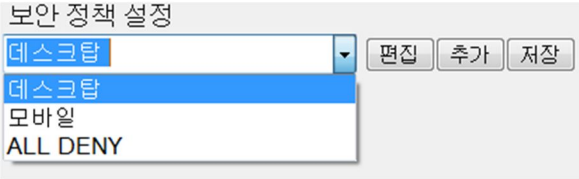

그림 15. 보안정책 설정의 예

보안정책은 XML 파일을 통해서 배포된다. XML 을 구성하는 태그를 이용해서 정책을 설정한다. <SecurityTarget> 태그는 시스템의 보안 정책을 설정한다. <platform> 태그를 사용해서 보안 정책을 적용할 대상을 결정하고, <SecurityLevel>은 보안 등급을 설정한다. 보안 등급을 설정하는 세부적인 내용은 <ContextInformation> 태그를 이용해서 시스템의 상황정보와 연계해서 사용한다. <SecurityMethod>는 보안 시스템에서 사용할 보안 기법을 설정한다. <ifSeverNotConnection>태그는 서버와 연결이 끊어지면 보안 정책을 갱신할 수 없기 때문에 연결이 끊어지게 될 경우 실행하는 보안 기법을 표기한다. 그림 15 는 보안정책 설정을 위한 관리자 화면이다.

예를 들어, 표 14 에 설정된 보안 정책 중 '데스크탑'은 CPU 와 RAM 의 사용률이 30% 이하이면 분류 등급을 '낮음'으로 분류한다. 사용률이 낮기 때문에 얼굴인식을 사용 할 수 있다. 만약 만약 분류 등급이 '높음'으로 변경되면 시스템 자원을 고려하여 ID/PW 인증을 시도한다. 서버와 연결이 끊어지면 '프로세스 제어(ProcessContol)'와 함께 내부 파일을 '암호화(FileCrypto)' 한다.

표 14. 본 논문에서 사용한 시스템 보안정책을 위한 XML 파일

```
<?xml version="1.0"?>
<SecurityPolicy>
    <platform="데스크탑">
        <ContextLevel>낮음
            <ContextInformation1=CPU>30%</ContextInformation1>
            <ContextInformation2=RAM>30%</ContextInformation2>
            <SecurityMethod>FaceRecognition</SecurityMethod>
            <SeverNotConnection>ProcessControl,FileCrypto</SeverNotConnection>
        </ContextLevel >
    </platform>

    <platform="모바일">
        <SecurityLevel>일반</SecurityLevel>
            <ContextInformation1=CPU>50%</ContextInformation1>
            <ContextInformation2=GPS,IP>used</ContextInformation2>
              <SecurityMethod>PW,OTP,PIN</SecurityMethod>
              <SeverNotConnection>ProcessControl</SeverNotConnection>
    </platform>

    <platform="ALL DENY">
        <SecurityLevel>DENY</SecurityLevel>
    </platform>
</SecurityPolicy>
```

보안 정책은 '보안 정책 배포'를 통해서 시스템에 배포된다. '로그 분석 결과'는 사용자 로그를 분석하여 결과를 관리자에게 알려주고, '로그 비교 분석'은 분석된 로그를 취합해서 사용자의 인증 및 보안 특이사항 등을 관리자에게 보고한다. 분석과 비교 작업을 통해서 관리자는 사용자에게 적합한 맞춤형 보안을 제공할 수 있다.

5.4 개선된 AdaBoost 를 활용한 상황정보 기반의 보안시스템의 성능

5.4.1 성능 테스트를 위한 실험 환경

개선된 AdaBoost 를 활용한 상황정보 기반의 보안 시스템 성능을 테스트하기 위한 설정은 표 15 와 같다.

표 15. 성능 테스트를 위한 시스템 사양

플랫폼 / 항목	PC	스마트폰(Lumia 710)	Server
CPU	Intel i5-2400 3.1Ghz	ARM Core 1.4Ghz	Intel i7-3770x2
RAM	DDR3 16GB	DDR2 512	DDR3 64GB
HDD	SSD 128GB	Flash 8GB	SSD RAID-0

보안 시스템의 성능 평가를 수행하기 위해서 자동화 스크립트를 만들어서 임의적으로 시스템에 부하를 주도록 설정하였다. 자동화 스크립트는 python 으로 작성하였다.

성능 테스트는 구현방법은 아래와 같다.

1) 보안 시스템이 적용된 시스템은 서버와 SSL 연결을 수행한다. 보안 시스템이 미적용된 시스템은 TCP/IP 를 사용해서 서버와 연결한다.
2) 시스템에 과부하를 임의로 발생시킨다.
3) 보안 시스템이 적용된 시스템은 ID 인증과 OTP 인증을 실해하도록 설정한다. 보안 시스템이 미적용된 시스템은 ID 인증만 수행한다. ID 인증은 임의의 5 개 ID 를 설정해서 자동으로 인증

하도록 설정하였고, OTP 인증은 자동화 테스트를 수행하기 위해서 200 자리 난수열을 임의로 생성하고, 화면에 출력한 뒤 자동적으로 난수열이 자동으로 입력되도록 설계되었다.

4) 인증과정이 종료하면 현재의 상황정보(CPU 사용률, RAM 사용률)를 RSA 알고리즘을 이용해서 암호화하여 서버에 전송한다.
5) 전송이 종료되면 1)~4)를 반복한다.

성능 평가는 PC 의 경우 6 시간, 스마트폰의 경우 4 시간동안 진행된다. 운영체제는 PC 의 경우 Microsoft 사의 Windows7 에서 진행하고, 스마트폰은 Windows Phone 7.5 에서 실행한다. 자동화 스크립트는 Microsoft 에서 제공하는 프로세스 제어 API 를 사용해서 약 30% ~ 100% 사이의 부하를 발생시킨다. 테스트에 사용된 모든 보안 기법은 공정한 평가를 위해서 운영체제에서 기본적으로 지원하는 기능을 사용하였다. 보안기법은의 경우 보안 시스템이 적용된 시스템은 ID 인증과 OTP 인증을 AdaBoost.CA 의 분류에 따라서 차등으로 적용하게 되고, 보안 시스템이 미적용된 시스템은 ID 인증만 수행하도록 설정하였다. 모든 평가는 동일한 조건에서 수행하였다. 아래 표 16 은 성능 테스트를 위해서 보안 기법 설정과 시스템 환경 설정을 정리한 것이다.

표 16. 보안 시스템 성능 테스트를 위한 보안 기법 설정 및 시스템 환경 설정

보안 적용 여부 / 환경 설정	보안 시스템 적용 PC		보안 시스템 미적용 PC		보안 시스템 적용 스마트폰		보안 시스템 미적용 스마트폰	
테스트 시간	6 시간				4 시간			
SW 환경	Windows7				Windows Mobile 7.5			
자원 소모	30~100% 임의부하 실행							
네트워크	SSL		TCP/IP		SSL		TCP/IP	
보안기법	인증기법	ID, OTP	인증기법	ID	인증기법	ID, OTP	인증기법	ID
	암호화	RSA	암호화	없음	암호화	RSA	암호화	없음

성능 평가의 결과는 보안 시스템의 프로세스가 실행되고 종료되는 동안 사용되는 CPU 와 RAM 점유율을 서버에서 수집하여 실시간으로 평균을 구한다. 스마트폰의 경우 CPU, RAM 과 함께 배터리 사용시간을 추가해서 평가한다. 배터리 사용 시간은 스마트폰이 방전될 때 발생하는 이벤트 시간을 기준으로 측정한다.

5.4.2 PC의 보안 시스템 성능 테스트 결과 및 분석

보안 시스템이 적용된 경우와 미적용된 경우 PC 에서 사용되는 프로세스에 관한 자세한 정보는 표 17 과 같다. 보안 시스템이 적용된 경우 백그라운드에서 실행되는 서비스인 AdaBoost.CA 모듈이 CPU 와 RAM 을 가장 많이 사용한다. 보안 시스템이 적용되지 않은 경우 Client 실행 프로그램이 CPU 를 가장 많이 사용하고 ID 인증 모듈이 RAM 을 가장 많이 사용한다.

표 17. PC 에 사용된 보안 시스템의 모듈별 CPU 점유율 및 RAM 사용률

적용 시스템 / 적용 모듈	모듈별 CPU 점유율		모듈별 RAM 사용률	
	보안 시스템 적용	보안 시스템 미적용	보안 시스템 적용	보안 시스템 미적용
AdaBoost.CA.dll	65.00%	0%	75.00%	0%
Client.exe	6.25%	78%	3.25%	15%
AdaBoost.CA.SSL.dll	10.50%	0%	10.50%	0%
MS.RSA.dll	8.25%	0%	9.11%	0%
MS.Auth.dll	5.00%	22%	1.24%	85%
MS.OTP.dll	5.00%	0%	1.00%	0%

표 18 은 PC 에서 수행한 성능 테스트 결과이다. 보안 시스템이 적용된 PC 가 보안 시스템이 미적용 PC 에 비해서 CPU 사용률이 약 5% 줄었고, RAM 사용량은 2% 증가하였다.

표 18. PC 에 적용한 보안 시스템 성능 테스트

테스트 항목 / 테스트 시스템	평균 CPU 사용률	평균 RAM 사용률
보안 시스템 적용 PC	약 26.4%	약 17.3%
보안 시스템 미적용 PC	약 31.7%	약 15.2%
차이	약 -5.3%	약 +2.1%

보안 시스템이 적용된 PC 의 CPU 사용률이 줄어든 이유는 시스템 사용률이 높아지면 AdaBoost.CA 에 의해서 ID 인증 관련 모듈을 미리 불러오기 때문이다. 그러므로 CPU 사용률이 약 100%에 도달했을 때, OS 가 ID 인증을 수행하기 위해서 실행되는 문맥교환(context switch) 과정이 간소화

되기 때문이다. 반면 보안 시스템이 적용되지 않은 PC 의 경우 인증에 필요한 모듈을 호출하기 때문에 오버헤드가 증가한다.

반면에 램 사용량은 보안 시스템이 적용된 PC 의 경우 증가하게 되는데, AdaBoost.CA 알고리즘에 의해서 보안등급을 실시간으로 분류하고, 분류결과를 적용하기 위해서 백그라운드에서 실시간으로 모듈을 호출하기 때문이다.

그림 16 과 그림 17 은 ID 인증 실행 시 보안 시스템이 적용 되었을 때와 미적용 되었을 때 CPU 사용률을 나타내는 그림이다. 그림 17 는 그림 16 에 비해서 ID 인증 모듈을 호출하기 위한 오버헤드 때문에 CPU 사용률이 가파르게 상승하는 것을 볼 수 있다.

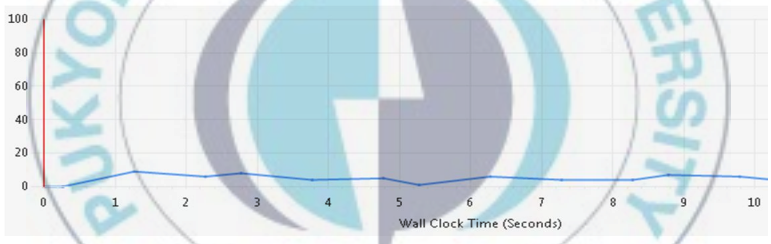

그림 16. PC 에서 보안 시스템이 적용시 사용되는 CPU 사용률

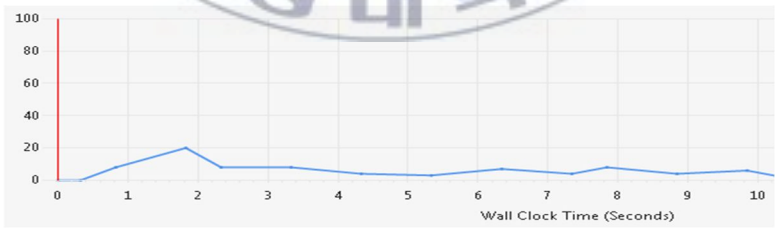

그림 17. PC 에서 보안 시스템이 미적용시 사용되는 CPU 사용률

5.4.3 스마트폰의 보안 시스템 성능 테스트 결과 및 분석

보안 시스템이 적용된 경우와 미적용된 경우 스마트폰에서 사용되는 모듈에 대한 정보는 표 19 와 같다. 보안 시스템이 적용된 경우 백그라운드에서 실행되는 서비스인 AdaBoost.CA 모듈과 wpClient 앱이 CPU 와 RAM 을 가장 많이 사용한다. 보안 시스템이 적용되지 않은 시스템의 경우 wpClient 앱이 CPU 를 가장 많이 점유하고 RAM 을 가장 많이 사용한다. PC 에 비해서 스마트폰의 경우 wpClient 앱이 자원을 많이 점유하고 있다. 이것은 스마트폰의 특성상 최상단에 위치한 앱에게 가장 많은 자원을 할당하기 때문이다.

표 19. 스마트폰에 사용된 보안 시스템의 모듈별 CPU 점유율 및 RAM 사용률

적용 시스템 / 적용 모듈	모듈별 CPU 점유율		모듈별 RAM 사용률	
	보안 시스템 적용	보안 시스템 미적용	보안 시스템 적용	보안 시스템 미적용
AdaBoost.CA.dll	35.12%	0%	60.00%	0%
wpClient.xap	30.50%	95%	20.25%	85%
AdaBoost.CA.SSL.dll	8.27%	0%	5.00%	0%
MS.RSA.dll	8.25%	0%	10.64%	0%
MS.Auth.dll	10.86%	5%	2.11%	15%
MS.OTP.dll	7.00%	0%	2.00%	0%

표 20 은 스마트폰에서 수행한 성능 테스트 결과이다. 스마트폰의 경우 CPU 와 RAM 의 사용량은 감소되었고, 배터리 사용 시간도 증가하였다. CPU 사용량이 감소한 이유는 문맥 교환에 관련된 오버헤드가 감소했기 때문이다. 그리고 RAM 의 경우 PC 에 비해서 사용량이 감소한 이유는 백그라운드 서비스로 작동하는 모듈의 경우 RAM 을 회수하지 않고

고정적으로 할당했기 때문이다. 예를 들어서, 보안 시스템이 미적용된 스마트폰의 경우 wpClient 에 너무 많은 자원을 할당해서 문맥 교환에 따른 CPU 오버헤드와 wpClinet 에 집중된 자원을 재분배하는 과정에서 동적 할당에 따른 RAM 의 오버헤드가 동시에 발생한다. CPU 와 RAM 의 사용량 감소로 스마트폰의 배터리 사용시간이 증가하였다.

표 20. 스마트폰에 적용한 보안 시스템 성능 테스트 결과

테스트 항목 / 테스트 시스템	평균 CPU 사용률	평균 RAM 사용률	평균 전원 유지시간
보안 시스템 적용 스마트폰	24.3%	15.6%	3 시간 35 분
보안 시스템 미적용 스마트폰	32.6%	20.9%	3 시간 22 분
차이	-8.3%	-5.3%	+13 분(약 6.5%)

그림 18 와 그림 19 는 ID 인증이 실행될 때 보안 시스템 적용 유무에 따른 CPU 와 RAM 사용률을 보여준다. 그림 18 에서 볼 수 있듯이 스마트폰의 자원관리 능력이 엄격하기 때문에 백그라운드에서 수행 중인 모듈에 대한 자원 소비가 일정하게 나타난다. 그림 19 는 wpClient 에서 ID 인증 모듈을 호출 할 때 발생하는 오버헤드 때문에 CPU 사용량이 급격하게 증가하는 것을 확인 할 수 있다.

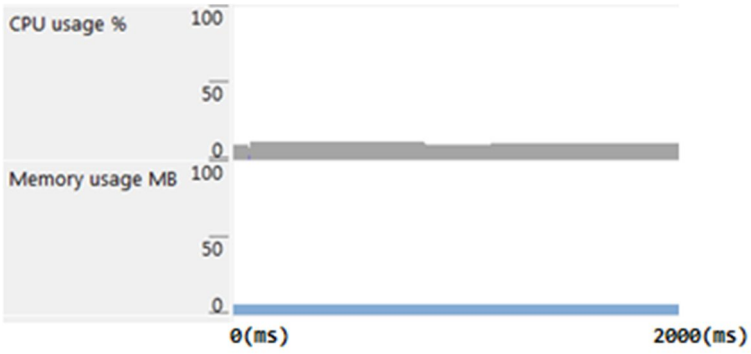

그림 18. 스마트폰 환경에서 보안 시스템 적용시 사용되는 CPU 와 RAM 사용률

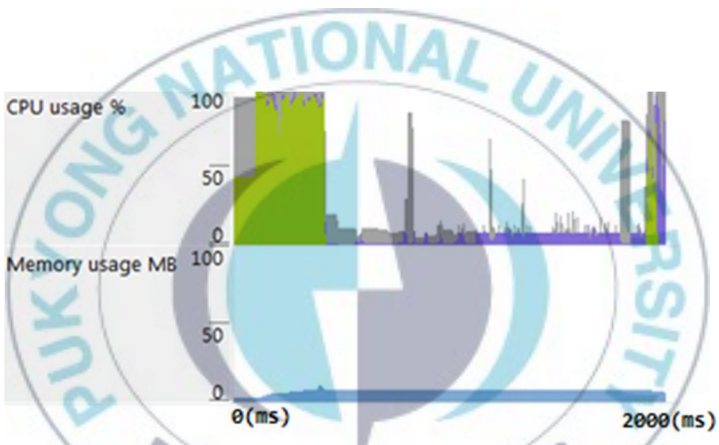

그림 19. 스마트폰 환경에서 보안 시스템 미적용시 CPU 와 RAM 사용률

5.5 개선된 AdaBoost 를 활용한 상황정보 기반의 보안시스템의 특징

개선된 AdaBoost 알고리즘을 활용한 상황정보 기반의 보안시스템의 특징은 다음과 같다. 첫째, AdaBoost.CA 를 사용하여 현재 시스템 상태에 가장 적절한 보안 기법을 실행한다. 그러므로 제한된 자원을 제공하는

시스템에서 자원을 효율적으로 사용할 수 있다. 앞선 성능 평가 결과에서 보듯이 CPU 와 RAM 의 사용률을 일정하게 유지할 수 있으며 과도한 시스템 부하 상황에서 보안 기법을 적용 할 때 생기는 오버헤드를 줄일 수 있기 때문에 스마트폰의 경우 배터리 사용시간도 증가한다. 둘째, AdaBoost.CA 를 사용해서 현재 상황을 반영하여 보안정책을 실행한다. 보안정책은 서버와 통신이 끊어져도 Client 프로그램에서 계속 유지할 수 있다. 기존의 보안 시스템은 연결이 끊어지면 매우 단순한 형태의 보안 기법을 제공하지만, AdaBoost.CA 를 활용한 보안 시스템은 Server 와 연결이 끊어져도 Client Agent 에 의해서 보안정책을 계속적으로 유지할 수 있다. 셋째, 관리자는 보안 정책 설정과 관리 기능을 통해서 다양한 보안 정책을 배포 할 수 있다. 또한 개별 사용자의 보안 정책을 설정하면 사용자 맞춤 보안을 제공 할 수 있다.

6. 결론

본 논문은 상황정보를 분류하기 위해서 기존의 AdaBoost 알고리즘을 개선한 AdaBoost.CA 알고리즘을 제안하였고 AdaBoost.CA 를 활용해서 다양하게 수집되는 상황정보를 분류하여 현재 상황에 적절한 보안기법을 제공할 수 있는 보안 시스템을 설계하였다. 보안 시스템의 설계를 바탕으로 시스템을 구현하여 AdaBoost.CA 의 활용성과 보안 시스템의 성능 및 평가를 진행하였다.

본 논문은 다음과 같은 의의를 지닌다. 첫째, 상황정보 분석을 위해서 사용된 AdaBoost.CA 알고리즘은 다양한 상황정보를 분류 할 수 있다. AdaBoost.CA 알고리즘은 기존 AdaBoost 알고리즘이 가진 장점인 분류 속도와 안정성을 기반으로 다양한 범주에 적용할 수 있다. 둘째, 상황정보 기반의 보안 시스템 모델을 제시하였다. 기존의 보안 시스템이 강력한 보안 기법을 중심으로 단일한 방법을 제공하거나, 두 개 정도의 보안 기법을 적절히 혼용하여 보안 성능을 향상시켰다. 하지만 본 논문에서 제안한 보안 시스템은 사용자의 현재 상태에 가장 적합한 보안 기법을 다양하게 제공할 수 있도록 설계하였다. 제안한 보안 시스템을 통해서 상황정보를 활용한 유연하고 강력한 보안 시스템의 모델을 제시하였다. 마지막으로 본 논문에서 제안한 보안 시스템은 사용자의 특성을 반영하여 보안 기법을 제공할 수

있으며, 제한적인 자원을 효율적으로 사용 할 수 있다. 시스템이 과부하 상태일 때 AdaBoost.CA 분류기가 자동으로 보안등급을 결정하여 현재 시스템 상태에 가장 적합한 보안 기법을 제공한다. 보안 기법 제공시 운영체제에서 발생하는 오버헤드를 줄일 수 있다. 이러한 오버헤드의 감소는 배터리 사용시간을 증가시킨다.

향후 연구를 통해서 관리자가 보안정책을 결정할 때 사용자의 패턴을 분석하여 맞춤 보안을 제공할 수 있도록 사용자 패턴에 관한 연구가 필요하다. 더 나아가서 보안정책을 관리자의 설정 없이 자동적으로 조절 할 수 있도록 상황정보 추론에 관한 연구를 진행할 예정이다.

참고문헌

[1] 공영일, "스마트폰의 함의와 시사점," 방송통신정책, 제 22 권, 제 4 호, pp. 3, 2010.

[2] 정두남, 최성진, "스마트 TV 의 기술과 방송정책, 방송통신연구," 방송통신연구 2011 년 겨울호, pp. 81, 2012.

[3] 심미선, 김은미, 오하영, 김반야, "스마트함이란 무엇인가," 한국방송학보, 제 26-3 호, pp. 260, 2012.

[4] 이해룡, 박준석, 이전우, "감성 UX 기술동향," 전자통신동향분석, 제 26 권, 제 5 호, pp. 81, 2011.

[5] 문성철, 윤해진, "태블릿 PC 채택의도에 미치는 영향요인 연구," 한국언론학회, 제 56 권, 제 3 호, pp 322-323, 2012.

[6] 배선욱, 김정한, 엄영익, "모바일 환경에서 사용자 반응성 향상을 위한 OS 지원 기법," 정보과학회논문지, 제 18 권, 제 7 호, pp. 533-534, 2012.

[7] 강준명, 고탁균, 서신석, 성백제, John Strassner, 김종, 박찬익, 홍원기, "상황인식 서비스를 위한 스마트 모바일 플랫폼," 정보과학회지, 제 28 권, 제 5 호, pp. 57, 2010.

[8] 류승완, 장효선, 신동천, 박세권, "상황인식(Context Awareness) 컴퓨팅 기술 동향," 정보통신산업진흥원, 주간기술동향 1435 호, pp. 5, 2010.

[9] Bill Schildt and Marvin Theimer, "Diseminating Active Map Information to Mobile Hosts," IEEE Networks, Vol. 8, No. 5, pp. 22-32, 1994.

[10] Guanling Chen and David Kotz, "A survey of context-aware mobile computing research," Dept. of Computer Science, Dartmouth College, Technical Report TR2000-381, pp. 1, 2000.

[11] Senaka Buthpitiya et al., "Hermes - a Context-Aware Application Development Framework and Toolkit for the Mobile Environment," 26th International Conference on Advanced Information Networking and Applications Workshops, pp. 664, 2012.

[12] Roy Want et al., "The Active Badge location system," ACM Transactions on Information Systems, pp 91-102, 1992.

[13] Amit Kumar Saha, "A Developer's First Look At Android," LINUX FOR YOU Magazine, Vol. 1, pp. 48, 2008.

[14] Antonio Miele et al., "A methodology for preference-based personalization of contextual data," Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 288, 2011.

[15] John D. Sutter, "Report: iPhones secretly track their users' locations," CNN, 2011.; http://edition.cnn.com/2011/TECH/mobile/04/20/iphone.tracking

[16] Jason Pascoe, "Context-aware software," Doctoral Dissertation, University of Kent at Canterbury, Dept. Computer Science, pp. 74, 2000.

[17] P.J. Brown, "The Stick-e Document : a Framework for Creating Context-Aware Applications," Electronic Publishing, Vol. 9, No. 1, pp. 2, 1995.

[18] David Franklin and Joshua Flaschbart, "All Gadget and No Representation Makes Jack a Dull Environment," AAAI Technical Report SS-98-02, pp. 155, 1998.

[19] Anind K. Dey et al., "CyberDesk : A Framework for Providing Self-Integrating Context-Aware Services," Knowledge-Based Systems, Vol. 11, No. 1, pp. 3, 1998.

[20] Bill Schilit et al., "Context-Aware Computing Applications," IEEE, pp. 85, 1994.

[21] Richard Hull et al., "Towards situated computing," Proceedings of the 1st IEEE International Symposium on Wearable Computers, pp. 146, 1997.

[22] Anind K. Dey, "Context-aware computing The CyberDesk project," AAAI Technical Report SS-98-02, pp. 52, 1998.

[24] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information, Vol. 13, No. 1, pp. 22-23, 1967.

[25] Christopher J.C. Burges, "A tutorial on support vector machines for pattern recognition," Data mining and knowledge discovery, Vol. 2, No. 2, pp. 121-167, 1998.

[26] Anil K. Jain et al., "Artificial neural networks: A tutorial," IEEE Computer, Vol. 29, No. 3, pp. 31-44, 1996.

[27] Pascoe Jason, "Adding generic contextual capabilities to wearable computers," Proceedings of 2nd International Symposium on Wearable Computers, pp. 95, 1998.

[28] Anind K. Dey and Gregory D. Abowd, "Towards a Better Understanding of context and context-awareness," Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, pp. 3, 1999.

[29] Valiant. L. G., "A theory of the learnable," Communication of the ACM, Vol. 27, No. 11, pp. 1135, 1984.

[30] Kearns M. and Valiant. L. G., "Cryptographic limitations on learning Boolean formulae and finite automata," Journal of the Association for Computing Machinery, Vol. 41, No. 1, pp. 67-95, 1994.

[31] Robert E. Schapire, "The strength of weak learnability," Machine Learning, Vol. 5, No. 2, pp. 197-227, 1990.

[32] Yoav Freund, "Boosting a weak learning algorithm by majority," Information and Computation , Vol. 121, No. 2, pp. 256-285, 1995.

[33] Harris Drucker et al., "Boosting performance in neural networks," International journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4 , pp. 705-719, 1993.

[34] Yoav Freund and Robert E. Schapire, "Experiments with a new Boosting algorithm," In Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148-156, 1996.

[35] Robert E. Schapire and Yoram Singer, "Improved Boosting algorithms using confidence-rated predictions," Machine Learning, Vol. 37, No. 3, pp. 297-336, 1999.

[36] Robert E. Schapire, "The Boosting Approach to Machine Learning An Overview," MSRI Workshop on Nonlinear Estimation and Classification, pp. 3, 2002.

[37] B Wu et al., "Fast rotation invariant multi-view face detection based on real adaboost," Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'04), pp. 79-80, 2004.

[38] Grigorios Tsoumakas and Ioannis Katakis, "Multi-Label Classification: An Overview," International Journal of Data Warehousing and Mining (IJDWM), Vol. 3, No. 3, pp.1-13, 2007.

[39] Paul Viola and Michael Jones, "Robust Real-time Object Detection," Second International Workshop on Statistical and Computational Theories of Vision, pp. 137-154, 2001.

[40] Alexander Vezhnevets and Vladimir Vezhnevets, "Modest AdaBoost - Teaching AdaBoost to Generalize Better," Graphicon-2005, 2005.

[41] Jerome Friedman et al., "Additive Logistic Regression: a Statistical View of Boosting," The Annals of Statistics, Vol. 28, No. 2, pp. 337–407, 2000.

[42] Rainer Lienhart, Alexander Kuranov and Vadim Pisarevsky," Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection," MRL Technical Report, pp. 297, 2002.

[43] William Stallings, Cryptography and Network Security, Prentice Hall, 2011.

[44] Aaron Carroll and Gernot Heiser, "An analysis of power consumption in a smartphone," Proceedings of the 2010 USENIX Annual Technical Conference, pp.1 2010.

감사의 글

2 년전 원서를 제출하고, 면접을 보던 기억이 엊그제 같은데 졸업을 눈앞에 두게 되었습니다.

그 동안 많은 가르침과 도움을 주신 많은 분들께 감사드립니다. 특히 바쁘신 중에도 아낌없는 보살핌과 가르침으로 지도해주신 정목동 교수님께 진심으로 감사 드립니다. 교수님의 따듯하고 열정적인 지도와 교훈으로 학문의 첫걸음을 무사히 시작할 수 있었습니다. 좋은 스승을 만나는 것은 어렵지 않은 일이지만, 존경할 수 있는 스승을 만나게 된 것은 제 인생의 다시 없을 행운이라 생각합니다. 부족한 저의 논문을 심사 및 조언해 주신 윤성대 교수님과 서경룡 교수님에게도 깊은 감사의 마음을 전합니다. 또한 많은 가르침과 격려을 아낌없이 주신 학과의 교수님들께 감사를 드립니다.

연구실 생활을 함께하면서 많은 도움을 받았던 재천 선배, 석환 선배, 현동 선배와 연구실 실원들에게 감사드립니다.

끝으로 세상에서 가장 따듯한 사랑과 과분한 정성을 주시는 할머니에게 깊은 감사의 마음과 사랑을 전하고 싶습니다.

후회와 아쉬움은 뒤로하고, 새로운 시작의 앞에서 한걸음 더 나아가려 합니다. 힘들 때 뒤를 돌아볼 수 있는 좋은 추억과 따뜻한 마음을 마음 속에 간직하며 이만 줄입니다.

2012.12.17, 한상곤 올림