2021년 2월

석사학위 논문

보안 공격 대응을 위한 추론 기반 접근제어 프레임워크

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 지 선

보안 공격 대응을 위한 추론 기반 접근제어 프레임워크

Inference-based Access Control Framework for Responding to Security Attacks

2021년 2월 25일

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 지 선

보안 공격 대응을 위한 추론 기반 접근제어 프레임워크

지도교수 최 준 호

이 논문을 공학석사학위신청 논문으로 제출함.

2020년 10월

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 지 선

김지선의 석사학위논문을 인준함

위원장 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 신 주 현 (인)

위 원 조선대학교 교수 최 준 호 (인)

2020년 11월

조선대학교 산업기술창업대학원

목 차

표 목 차

그 림 목 차

ABSTRACT

Inference-based Access Control Framework for Responding to Security Attacks

Jiseon Kim
Advisor : Prof. JunHo Choi, Ph.D
Industrial Technology and
Entrepreneurship Chosun University

Recently, security breaches have diversified types of attacks, including damage to companies, use and leakage of personal information, and the level of attacks has also been increasing. Since the number of attack case has been gradually increased, the needs of Access Control Technology and Security Analysis is becoming more and more important.

Therefore, in this paper, it will be suggested the way that applying the Intelligent Context Awareness Access Control Model to the System through these processes: 1) modelling the security context information ontology of system, 2) testing the Intelligent Context Awareness Access Control Model which uses the inference of security situation when the illegal approach is happened. In the experiment, it is used that the 'syslog' log file analysis and combining it into the Security Context Information to trace certain action which occurred in the system.

Since the Probability of Detection, according to the security context, was 70% and the accuracy of Response Rate on the attack situation was 72.8% in the experiment of this paper, it can be regarded as the detection and response was appropriate.

The proposed method in this paper is following. To respond on the attack which is comprehensive and intelligent security invasion accident. it is needed that designing and applying the Intelligent Context Awareness Access Control Model which can response simultaneously on the illegal approach of power system resource. This Model can be derived by Inferring Technology which uses ontology modelling based on the security context information of system.

Ⅰ. 서론

A. 연구 배경 및 목적

기술이 발달함으로써 개인의 삶의 질이 높아지고, 기업도 많은 이익을 창출하는 반면에 불법적으로 개인정보 유출, 기업의 중요정보를 탈취하거나 업무를 마비시키는 해킹사고 건수가 해마다 증가하고 있다. [표 1-1]은 한국인터넷진흥원이 발표한 해킹사고 건수 통계(연, 2010~2019)자료이다. 이 자료를 보면 10년 동안 발생한 총 해킹사고는 202,181건으로 조사되었고, 그중, 기관이나 기업 침해사고는 매년 발생 건수가 증가하는 추세이며 다른 해킹사고에 비해 증가율이 높음을 확인할 수 있다. 이런 증가율의 원인으로는 지능적이고 다양화 된 해킹 공격 기법의 출현을 들 수 있다[1].

[표 1-1] 연도별 해킹사고 건수

구분	홈페이지 변조	침해 신고접수	악성코드 은닉사이트 탐지	년도별 해킹사고 건수
2010년	3,043	53	-	3,096
2011년	1,854	63	11,805	13,722
2012년	3,157	91	13,018	16,266
2013년	1,700	82	17,750	19,532
2014년	1,115	175	47,703	48,993
2015년	615	225	46,850	47,690
2016년	1,056	247	11,044	12,347
2017년	1,724	287	13,347	15,358
2018년	567	500	14,754	15,821
2019년	639	418	8,299	9,356
총 계	15,470	2,141	184,570	202,181

최근 발생한 대표적인 개인정보 유출 보안피해 사례는 [표 1-2]와 같다[2].

[표 1-2] 대표적인 개인정보 유출 보안 사고 사례

시기	기관	피해규모	피해내용
2014년 3월	KT	1,200만 건	KT 홈페이지에서 '고객서비스 계약번호'가 변조 되었는지에 대한 여부를 확인하지 않는다는 취약점을 이용하여 1,200만명의 개인정보를 유출한 사건으로, 조사 결과 해커는 사이트에 약 1,266만번의 접속 기록이 확인되었으나, KT는 이상징후를 탐지하지 못했다[3].
2015년 9월	뽐뿌	196만 건	사용된 해킹기법은 SQL 인젝션이며, 홈페이지에서는 비정상적인 DB 질의에 대한 검증절차가 없었다. 또한, 개인정보 DB 서버 중 일부 서버에서만 로그를 저장하는 등의 기술적, 관리적 보호조치가 미흡했다[4].
2016년 7월	인터파크	1,030만 건	직원이 악성코드가 첨부된 e메일을 열람하여 악성코드가 감염되었다. 감염된 PC를 이용하여 파일공유서버에 접속 및 악성코드를 설치하여 악성코드를 유포 후 악성코드를 이용하여 정보를 수집했다. 파일을 공유하는 서버를 경유하여 개인정보를 취급하는 담당자 PC로 접속했으며, 기존에 연결되어 있던 DB 서버에 접근하여 개인정보를 탈취한 사건이다[5].
2017년 6월	빗썸	3만 6천 건	원격제어 형 악성코드가 포함된 이력서 파일을 첨부한 메일을 받고 첨부파일을 실행한 개인용 컴퓨터가 악성코드에 감염되었다. 해당 PC에는 다수의 개인정보가 포함되어 있는 파일이 저장되어 있어, 해당 개인정보 파일이 유출되었다[6].

위의 사례들을 보면 시스템의 취약점을 이용하거나 불법적인 권한 획득으로 개인정보를 탈취한 것을 확인할 수 있다. 이러한 사고를 탐지하고 대응하기 위해 기존에는 일반적으로 패턴이나 행위 기반 등의 통계적인 방법을 사용하였는데 이러한 방법은 지능적이고 다양해지는 보안 침해 공격을 대응하는 데 무리가 있다. 따라서 본 논문에서는 정보 유출 등 보안 문제에 대해서 기존 접근제어 모델들이 가지고 있는 단점을 보완하여 동적이고, 유용한 접근제어 모델인 온톨로지 추론 기반의 보안상황 인식을 위한 지능형 접근제어를 제안하고자 한다.

B. 연구 내용 및 구성

본 연구는 보안 공격 대응을 위한 추론 기반 접근제어 프레임워크를 설계를 주제로 한다. 이를 위해 보안 취약점을 분석 후 보안상황 정보를 수집하고 수집 된 시스템의 취약점을 가지고 추론 엔진에 적용하여 신속한 대응이 가능한 모델을 설계하고자 한다. 이를 위해 본 논문은 다음과 같은 구성으로 작성되었다.

서론에 이어 2장에서는 온톨로지 기반 추론에 관한 관련 연구와 본 논문의 이론적인 배경이 되는 상황인식에 대한 개념을 정리하고, 접근제어의 개념과 기존 접근제어 모델에 대해 간략히 설명한다. 3장은 본 논문에서 제안하고자 하는 보안 상황인식을 위한 지능형 접근제어 방법의 연구방법 및 범위, 보안상황 정보수집, 지능형 접근 제어설계, 온톨로지 기반 보안상황 인식서비스 설계 단계로 나누어 설명한다. 4장에서는 성능평가 항목 및 지표, 실험에 사용한 추론규칙 및 각종 데이터셋에 관해 설명하고, 실험을 통해 보안상황 접근제어 온톨로지 기반 접근제어 추론 정확률, 보안상황 인식서비스의 공격탐지 추론 정확률 마지막으로 지능형 통합 접근제어 시스템의 공격 대응률을 확인한다. 마지막으로 5장에서는 연구 결론과 향후 추가로 진행 할 연구에 관해 서술하며 마무리한다.

Ⅱ. 관련 연구

A. 온톨로지 기반 추론

온톨로지(Ontology)는 한국전파기술협회에서 발행한 '2018 신기술 용어'를 보면 "사물의 본질, 존재의 근본 원리를 사유나 직관으로 탐구하는 형이상학의 한 분야인 존재론을 기반으로 실제에 대한 정확한 이해를 추구하는 철학에서 유래한 존재하는 사물과 사물 간의 관계 및 여러 개념을 컴퓨터가 처리할 수 있는 형태로 표현한 것"이라고 정의하였고, 스탠포드 대학의 그루버 교수는 온톨로지를 "일반적으로 개념화된 것을 형식적으로 명백하게 기술하는 명세"라고 정의하였다[7]. 온톨로지는 특정 도메인에서 특정 지식과 관련된 용어 또는 용어 사이의 관계를 정의하는 일종의 사전이라고 할 수 있다. 온톨로지는 내용 중심, 추론 기능, 특화세분화, 정보통합공유의 특징이 있다.

온톨로지는 자연어의 기계 번역과 인공지능, 자연어 처리 등 여러 분야에서 활용되고 최근에는 특정 분야의 자원과의 관계를 기술하는 시맨틱웹과 여기서 파생된 시맨틱웹 서비스 등의 핵심 요소로 주목받고 있다.

온톨로지는 객체의 주체성 관련 정보를 명시하여 시맨틱웹의 자원을 식별하고 추론을 가능하게 하고, 시맨틱웹 모델 간의 관련성을 파악하고 이질성을 줄이는 데 도움을 주는 역할을 한다. 이런 온톨로지의 구성요소는 개념, 관계, 속성으로 구분할 수 있다. 개념은 특정 영역에 대한 Concept으로 클래스와 인스턴스로 나눌 수 있다. 클래스는 일반적으로 눈으로 볼 수 있는 사물이나, 보이지는 않지만 표현할 수 있는 개념 등에 붙이는 이름이라고 설명할 수 있다. 예를 들어 노트북, 휴대폰, 우정과 같은 것은 클래스라고 할 수 있다. 그리고 인스턴스는 앞에서 설명 한 클래스가 실질적인 형태로 나타난 것을 의미한다. LG Gram 노트북, 삼성 갤럭시 휴대폰은 일반적으로 인스턴스라 볼 수 있다. 클래스와 인스턴스를 구분하는 것은 응용 방법과 사용 목적에 따라 달라질 수 있다. 이는 같은 표현의 개체가 어떠한 경우에는 인스턴스가, 또 어떤 경우에는 클래스가 될 수도 있다는 것을 의미한다. 속성은

클래스나 인스턴스의 성질이나 성향 등을 나타내기 위해 클래스나 인스턴스를 특정 값과 연결한 것이다. 관계는 클래스, 인스턴스 사이에 존재하는 관계들을 말하며, 계층적으로 표현하는 의미 관계(Taxonomic Relation)와 그 이외의 개념 관계(Non-taxonomic Relation)로 구분할 수 있다. [표 2-1]는 온톨로지의 구성요소를 나타낸다.

[표 2-1] 온톨로지 구성요소

구성요소	상세설명
개념 (Concept)	·특정 영역에 대한 개념으로 클래스와 인스턴스로 나눌 수 있다. ·클래스 : 사물이나 개념 등에 붙이는 이름 ·인스턴스 : 사물이나 개념의 구체물 혹은 실질적인 형태로 나타낸 자체로 클래스를 구체화한 것이다.
속성 (Property)	·클래스나 인스턴스가 가지는 특성값을 의미한다.
관계 (Relation)	·개념 간의 관계를 의미한다. ·의미 관계(Taxonomic Relation)와 개념 관계(Non-taxonomic relation)로 구분된다.

온톨로지에서 주로 사용하는 언어에는 Resource Description Framework(RDF), Ontology Web Language(OWL), Semantic Web Rule Language(SWRL) 등이 있다. RDF는 XML에서 발전한 형태이며, 웹에 있는 자원에 대한 메타 정보를 표현하기 위한 언어이다. RDF는 주로 복잡한 제약조건이 필요 없는 응용을 산정할 경우에 사용한다. OWL은 관계들 간의 체계 또는 관계 인스턴스 내의 논리적 제약조건 등을 포함한 언어로써 논리적이고 정밀한 추론을 필요로 하는 경우에 사용한다. 마지막으로 SWRL은 온톨로지에서 정보들의 관계를 통해서 새로운 지식을 얻기 위한 규칙을 정의하는데 사용한다.

B. 상황인식

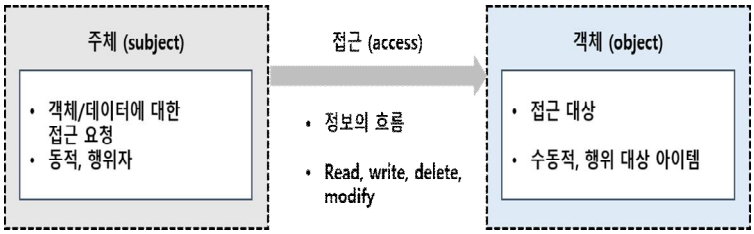
상황이란 "실제 존재하는 인간, 장소, 또는 인간과 서비스 간의 상호작용의 상태를 특징을 규정하는 정보" 라고 정의할 수 있다. 상황의 종류에는 네트워크 연결 상태, 대역폭과 같은 컴퓨팅 상황(Computing context), 사용자 프로파일과 같은 사용자 상황(User context), 조명, 소음, 온도 등의 물리적 상황(Physical context), 시간, 날짜 등의 시간적 상황(Time context)이 있다. 이런 상황들을 구체적으로 정형화하여 정의한 것을 상황 정보라 하며 이런 상황 정보들은 다양한 경로에서 파악되고 수집되어 다양한 응용서비스 제공에 이용되고, 다른 상황 정보와 연계하여 새로운 결론을 얻는 추론에도 사용된다. 마틴 파울러(Martin Fowler)는 "패턴이란 특정 상황(Context)에 대해 유용한 대처 방안이 다른 곳에서도 유용하게 적용 가능한 경우를 말한다."라고 하였다. 수집된 상황 정보에서 패턴을 발견해 미래에 발생할 사항에 대해 대비하는 것도 가능하다.

상황인식은 상황 정보를 인지하고 상황 지식에 맵핑하는 것이라 할 수 있다. 이렇게 맵핑된 상황 지식을 이용하여 해석, 추론과 같은 처리 과정을 통해 사용자에게 각 상황에 적절한 서비스를 제공한다. 상황인식 서비스는 다양한 서비스 분야에 적용할 수 있다. 상황인식 서비스를 비즈니스에 적용한 예로는 아마존 도서 구매를 들 수 있는데, 아마존은 사용자들이 과거에 구매했던 이력을 저장하고 분석한 결과를 토대로 도서검색이나 구매 시에 사용자에게 추천도서를 보여주는 서비스를 제공하고 있다. 여기서 상황 정보는 사용자의 검색정보, 구매정보, 성별, 연령대 등이 될 수 있다. 주요 기업으로 아마존, 구글, 페이스북 등을 들 수 있다. 이러한 예로 봤을 때 앞으로 상황인식 서비스는 의료, 교육, 쇼핑 등 여러 사회 분야에서 큰 영향력을 가진 서비스가 될 것으로 예상된다.

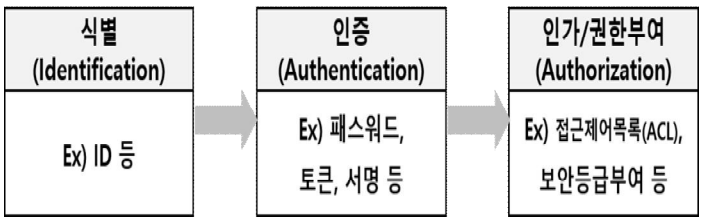
C. 접근제어

1. 접근제어의 개념

접근제어 기술이란 정상적으로 인증이 완료된 사용자가 허가받은 범위 내의 시스템 내부 정보에 접근하는 것을 허용하는 기술적인 방법으로 주체와 객체 사이의 정보 흐름을 통제하는 것을 의미한다. 주체는 사용자, 객체는 사용자가 접근하려고 하는 자원을 의미하고, 정보 흐름을 통제한다는 것은 사용자가 자원에 접근하려고 할 때 이 사용자가 접근하려 하는 자원 접근에 대한 합당한 권한이 있는지 검사하는 일련의 과정이다. 접근제어의 개념은 [그림 2-1], 절차는 [그림 2-2]로 정의할 수 있다.



[그림 2-1] 접근제어 개념



[그림 2-2] 접근제어 절차

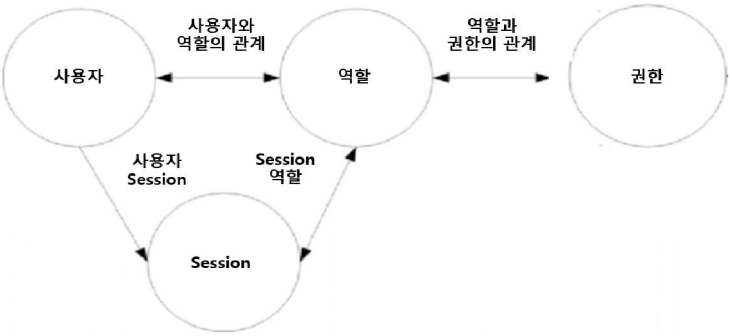
접근제어 절차는 세 단계로 나눌 수 있다. 사이트에 접속한다고 가정했을 때 로그인을 하려면 ID를 입력해야 하는데 이 과정을 식별이라고 한다. 이는 접근 주체가 누구인지 알리는 과정을 의미한다. 그리고 비밀번호를 입력해서 정확한 접근 주체인지에 대해 확인받아야 하는데 이 과정을 인증이라고 한다. 마지막으로 해당 사이트에서 사용자의 접근을 차단했는지를 확인하는 절차를 인가라고 한다.

2. 접근제어 모델

접근제어 모델의 종류는 크게 규칙기반 접근제어, 신분 기반 접근제어, 역할기반 접근제어로 나눌 수 있다. 규칙기반 접근제어 모델은 객체에 포함된 정보를 허용하는 등급과 접근하는 정보에 대해 주체가 갖는 접근 허가 권한을 근거로 하여 객체 접근에 제한을 두는 방법을 말하며 강제적 접근제어(Mandatory Access Control, MAC)와 동일한 개념으로 볼 수 있다. 이것은 정책을 중앙 집중적으로 관리하는 것을 의미하는데, 시스템에 의해 정책이 정해지면 모든 사용자는 정책을 따르면 된다. 관리하고자 하는 자원이 적을 경우 좋은 방법이지만 사용자 기능에 제한이 많고 관리적으로 어려움이 있다. 그리고 비용이 많이 발생되며 사용자에게 친화적이지 않아 상업적인 환경에는 적용이 힘들고 모든 접근에 대해서 lable을 정의하고 보안정책을 확인해야 하므로 성능저하의 우려가 있다는 단점이 있다.

두 번째로 신분 기반 접근제어 모델은 주체나 그룹의 신분을 근거로 객체 접근에 제한을 두는 방법을 의미하며 임의적 접근제어(Discretionary Access Control, DAC)와 동일한 개념으로 볼 수 있다. 이 모델은 모든 사용자가 정책을 관리할 수 있다. 타인의 자원에는 관여할 수 없지만, 자신이 관리하는 자원에 대해서는 정책을 설정할 수 있다. 구현이 대체로 쉽고 유연하게 권한 변경이 가능하다는 장점이 있지만 각각의 주체에 객체 접근 권한을 부여해야 하고, 타인의 신분으로 불법적인 접근이 이루어질 가능성이 있다는 단점이 있다. 규칙기반 접근제어 모델에 비해서 다소 유연한 정책으로 볼 수 있다.

세 번째로 역할기반 접근제어(Role-based Access Control, RBAC) 모델은 관리자가 주체와 객체의 상호관계를 통제하며 맡은 역할을 기초로 자원에 대한 접근 허용을 결정하는 것으로 사용자의 접근은 개인의 직무에 따라서 결정된다. 사용자 단위로 관리하는 것이 아니라 그룹으로 묶어서 그룹 기반으로 정책을 세팅한다. [그림 2-3]은 역할기반 접근제어 모델을 나타낸다.



[그림 2-3] 역할기반 접근제어 모델

사용자와 역할의 관계는 사용자에게 어떠한 역할이 부여되었는지를 의미하고 역할과 권한의 관계는 각 역할이 어떠한 권한을 가지는지를 의미한다. 역할기반 접근제어는 유연하고, 효율적인 관리가 가능하다. [표 2-2]는 접근제어 모델들을 비교한 것이다.

[표 2-2] 접근제어 모델 비교

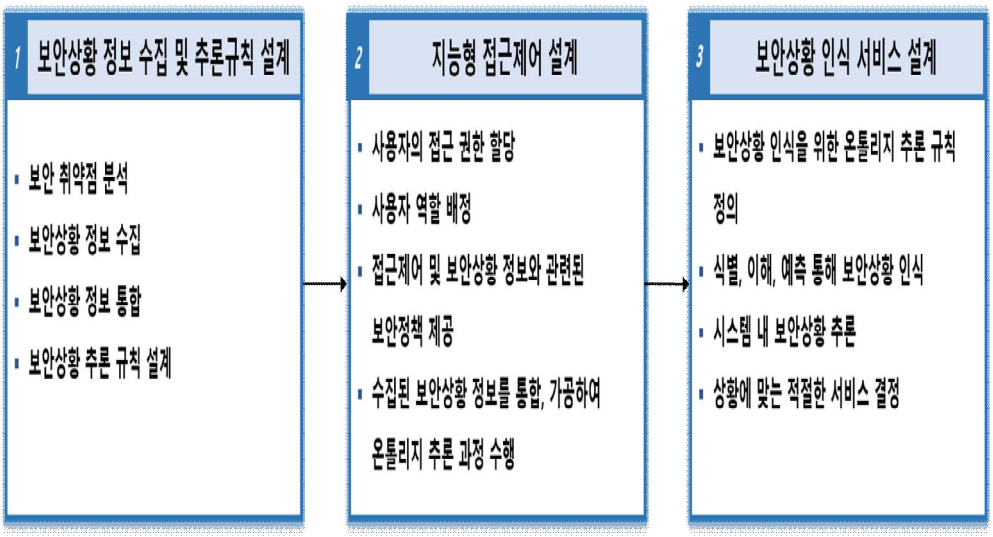
	강제적 접근제어	임의적 접근제어	역할기반 접근제어
정의	주체와 객체의 등급을 비교해 접근 권한을 부여	접근하고자 하는 주체나 그룹의 신분에 따라 접근 권한을 부여	주체과 객체 사이에 역할을 부여해 역할에 따라 접근 권한을 부여
특징	강제적 통제 정책 변경의 어려움 고정적	객체중심통제 정책변경 용이	그룹/Role 단위 통제 정책 변경 용이
제어주체 (권한부여)	System	데이터 소유자 (Data Owner)	중앙 관리자
접근결정	보안등급 (Lable)	신분 (Identity)	역할 (Role)
장점	중앙집중적, 안정적 보안성 매우 높음	구현이 용이하고 정책 변경에 유연	관리 및 구성변경 용이
단점	구현 및 운영의 어려움, 고비용	트로이목마(멀웨어), 아이디 도용 공격에 취약	동적인 접근제어 불가능
활용사례	방화벽, UNIX/Linux	Windows 등 OS	조직, 기업, ERP 등 상용 SW

역할기반 접근제어는 보안상황 요소를 고려하지 않았기 때문에 동적 접근제어가 불가능하다는 단점이 있는데 이것을 보완해서 나온 모델은 상황인식 역할기반 접근제어(Context-aware Role-based Access Control, C-RBAC) 모델이다. 이 모델은 기본적인 역할기반 접근제어 모델에 위치 정보에 따라 동적인 위치 정보에 따라 Role이 부여되고 권한이 현재 상황 정보에 맞는지 확인하여 권한이 부여된다. 하지만 2010년에 발표된 엄정호, 박선호, 정태명의 “내부자의 불법적 정보 유출 차단을 위한 접근통제 모델 설계”[8]에 따르면 C-RBAC 모델은 접근하려는 객체 사이의 보안등급을 고려하지 않아 정보의 기밀성과 무결성을 보장하지 못하고, 수행하고 있는 직무와 관련된 객체들에 합법적인 접근을 통해 정보를 유출하는 것을 막지 못한다는 연구결과가 나왔다. 따라서 본 논문에서는 역할기반 접근제어 모델을 참고하여 시스템에 적합한 내부 사용자의 상황 정보를 이용하여 자원에 접근하려는 사용자에게 접근 권한을 할당, 역할을 배정하고, 정당한 사용자가 유효한 접근 권한을 가지고 자원에 접근하여 서비스를 받을 수 있는 모델을 고안하였다.

Ⅲ. 보안 상황인식을 위한 지능형 접근제어

A. 연구방법 및 범위

본 논문에서는 보안 상황인식을 위한 지능형 접근제어를 위해 첫 번째로 보안상황 정보수집 및 보안상황 추론규칙 설계를 제안한다. 이는 특정 목적 수행을 위해 사용자를 대신하여 작업을 수행하는 자율적인 프로세스로서 본 연구의 핵심 모듈이다. 두 번째로는 보안정책 관리를 위한 지능형 접근제어 프레임워크 설계를 제안한다. 설계를 위해 상황인식 역할기반 접근제어(Context-RBAC) 모델을 이용하여 시스템에 적합한 내부 사용자의 상황 정보를 이용하여 정당한 사용자가 유효한 접근 권한으로 자원에 접근하여 서비스를 받을 수 있는 접근제어 모델을 설계한다. 세 번째로는 보안상황 온톨로지 기반 상황인식 서비스 프레임워크 설계를 제안한다. 보안 상황인식 서비스(Security Context-aware Service)는 보안상황 정보를 이해함으로써 보안상황에 맞게 적절히 반응하여 사용성과 효율성을 높일 수 있다. [그림 3-1]은 본 연구에서 다루는 단계별 내용을 나타낸다.



[그림 3-1] 연구 단계별 내용

B. 보안상황 정보수집 및 분석

1. 보안취약점 분석

서버시스템에서 발생할 수 있는 보안위협 요소는 통신 포트 접근을 통한 공격, 펌웨어 취약점을 이용한 공격, 원격 펌웨어 업데이트 기능의 취약점을 이용한 공격, 위장 및 서비스 거부 공격, 데이터베이스 공격, Zero-day 취약점을 이용한 공격 등이 있다. 네트워크에서 발생할 수 있는 보안위협 요소는 스니핑 공격, ARP 스푸핑 공격, 도청, 메시지 재전송 공격, 패킷 전송 방해 공격 등이 있다. [표 3-1]은 서버시스템 및 네트워크에서의 구분별 보안위협 요소를 나타낸다.

[표 3-1] 서버시스템 및 네트워크 보안위협 요소

구분	위협 요인	보안위협 요소	설명
서버 시스템	물리적 접근 용이성	통신 포트 접근을 통한 공격	서버/시스템의 광역적 배치와 물리적 접근 용이성으로 인한 보안위협 발생
		비인가 접근	
		위장	
	비인가 보조 기억매체 사용	악성코드 삽입 공격	서버/시스템 내 비인가 보조 기억매체 사용으로 인한 보안위협 발생
	부적절한 펌웨어 업그레이드	원격 펌웨어 업데이트 기능 취약점 악용 공격	서버/시스템의 펌웨어 업그레이드 시 펌웨어 코드의 무결성 검증 미흡 등으로 인한 보안위협 발생
	부적절한 보안 설정 및 관리	서비스 거부 공격	서버/시스템의 부적절한 보안 설정 취약으로 인한 보안위협 발생
		데이터베이스 공격	

	부적절한 계정관리	부적절한 계정관리를 이용한 공격	서버/시스템의 취약한 계정 정보, 계정 관리 미흡 등으로 인한 보안위협 발생
		무차별 대입 공격을 통한 관리자 권한획득 공격	
	운영체제 및 펌웨어 취약점	펌웨어 취약점 공격	비인가 된 백신 프로그램 사용으로 인한 인증되지 않은 F/W 및 소프트웨어 또는 운영체제와 펌웨어의 설계 및 구현까지 소요되는 시간에 발생하는 취약점 등으로 인한 보안위협 발생
	비인가 백신 프로그램 사용		
	응용프로그램 취약점	Zero-day 취약점을 이용한 공격	서버/시스템 운영 및 서비스 관련 응용 프로그램의 설계 및 구현까지 소요되는 시간에 발생하는 취약점 등으로 인한 보안위협 발생
네트워크	부적절한 네트워크 접근제어	스니핑 공격	비인가 된 트래픽의 네트워크 유입 등으로 인한 보안위협 발생
	인증 프로토콜 취약점	ARP 스푸핑 공격	상호인증 및 키 교환 및 관리 등의 취약점 등으로 인한 보안위협 발생
	안전하지 않은 네트워크 구성		
	통신 프로토콜 취약점	도청	Modbus, DNP3, TCP/IP 프로토콜 설계 및 구현 취약점 등으로 인한 보안위협 발생
		메시지 재전송 공격	
		트래픽 분석	
		패킷 전송 방해 공격	
		부적절한 키 생성 공격	

2. 보안상황 정보수집 범위

보안상황 정보수집 방법은 SIEM(Security Information & Event Management) 플랫폼을 참조하여 설계하였다. 2005년 가트너에 의해 만들어진 통합로그 분석 솔루션인 SIEM 플랫폼의 핵심기능은 네트워크 및 보안 장비로부터 정보를 수집, 분석, 제시하는 것이다. SIEM 플랫폼의 일반적인 기능은 다음 [표3-2]과 같다.

[표 3-2] SIEM 플랫폼의 기능

기능	세부기능
통합 보안 로그 수집 및 보관	- 보안시스템 및 운영체계 등 대용량 로그 통합관리 - 로그 실시간 수집 - 위/변조 방지 스토리지 내의 원본 로그 보관 - 기밀성과 무결성을 지원
탐지 및 분석을 통한 대응체계 구축	- 로그의 주요 이벤트 연관 분석 - 실시간 보안사건 탐지 및 경보 - 위험도 분류 등 새로운 위협 요인 관리
실시간 모니터링 및 관리	- 전용 콘솔을 이용한 실시간 모니터링 - 로그 현황 모니터링 및 조건 검색(수집/보관/탐지) - 분석을 통해 발생된 특정조건에 대한 실시간 알람

시스템 기반 보안상황 정보는 리눅스, 유닉스, Windows 등의 운영체제 시스템에서 생성되는 정보다. 이를 통해 관리자가 시스템 또는 응용프로그램에서 발생하는 각종 메시지의 확인이 가능하다. 그 중, ‘syslog’는 운영체제와 관계없이 사용할 수 있다는 장점이 있고, 프로세스의 정상적인 운영 메시지, 오류 메시지 등 다양한 이벤트를 포함하여 기록하고 있다.

본 연구에서는 시스템에서 발생한 특정 행위와 이에 대한 추적을 위해 ‘syslog’ 로그 파일의 구조를 분석하여 보안상황 정보로 통합한다. SYSLOG 데이터 내용은 [표 3-3]과 같다.

[표 3-3] SYSLOG 로그 데이터의 내용

로그종류	설명
syslog	UNIX 시스템에서 로그 메시지 처리를 위한 표준화된 인터페이스
messages	시스템 부팅에서부터 부팅 이후의 실행된 데몬의 메시지에 대한 로그
lastlog	사용자의 마지막 로그인 정보를 저장
xferlog	ftp 접근에 대한 사용 로그
sercure	시스템 접속 보안 인증(ssh) 로그 파일

침입탐지시스템은 탐지 위치에 사용 목적 및 탐지 위치에 따라서 다양한 시스템 사용 행위를 감시 및 탐지하도록 설계되었고, 침입탐지시스템의 특징은 데이터 위치, 침임 탐지 방법, 침입 대응 방법에 따라 다음 [표 3-4]와 같이 분류된다.

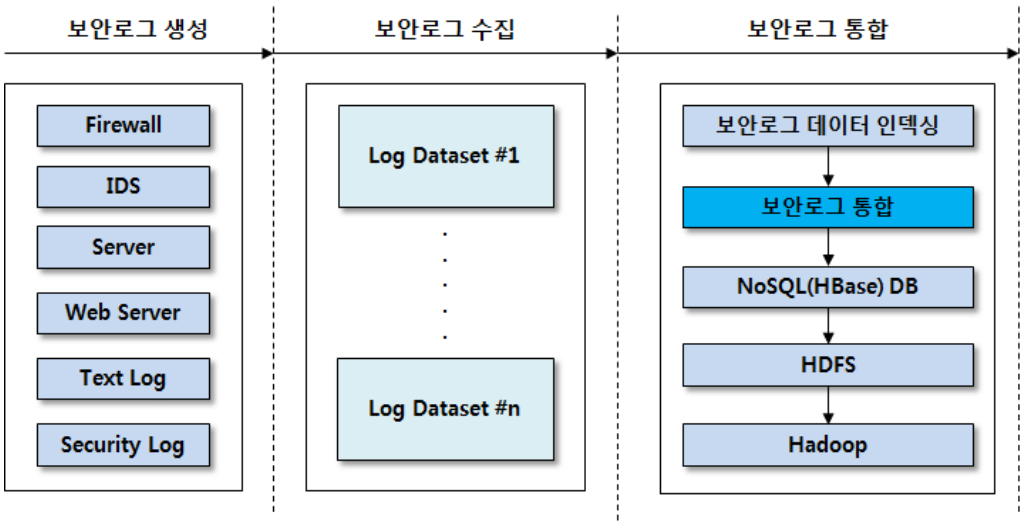
[표 3-4] 침입탐지시스템의 위치에 따른 분류

종류	내용
네트워크 기반 침입탐지시스템	네트워크 패킷을 수집하여 내/외부 탐지
단일 호스트 침입탐지시스템	Host의 로그 파일로부터 정보를 수집하여 탐지
다중 호스트 침입탐지시스템	다중의 Host로부터 정보를 수집하여 탐지
혼합형 침입탐지시스템	Host와 Network 조합하여 탐지

침입탐지시스템은 탐지 위치에 따라 다양한 로그가 발생하지만, 실제로는 같은 내용을 기록하고 있다. 본 논문에서는 단일 호스트 기반 침입탐지시스템 로그 파일을 분석하여 보안상황 정보 구조를 설계하였다.

3. 보안상황 정보수집 프레임워크

보안상황 정보는 시스템 내 각종 보안시스템의 로그 구조 분석을 통하여 보안상황 정보 패턴을 찾고 데이터를 원하는 형태로 가공 후 저장해야 한다. [그림3-2]은 시스템 내 보안 로그 데이터를 수집하는 과정을 도식화한 것이다.



[그림 3-2] 보안상황 정보수집 프레임워크

선별된 보안 로그는 분석 가능한 형태로 변환한다. 보안상황 정보수집에서 중요한 사항은 방대한 보안 로그 데이터에서 불필요한 데이터를 삭제하고, 유용한 데이터를 추출하기 위한 과정이다. 전처리 과정에서는 불필요한 문자열을 제거하고, 데이터 공통 영역을 찾아야 한다. 또한, 보안 로그 데이터 필드 구조 및 필드 내용을 검증하여 저장 전에 생성된 보안 로그의 정상 유무를 판단한다. 보안 로그 필드 구조는 [표 3-5]와 같다.

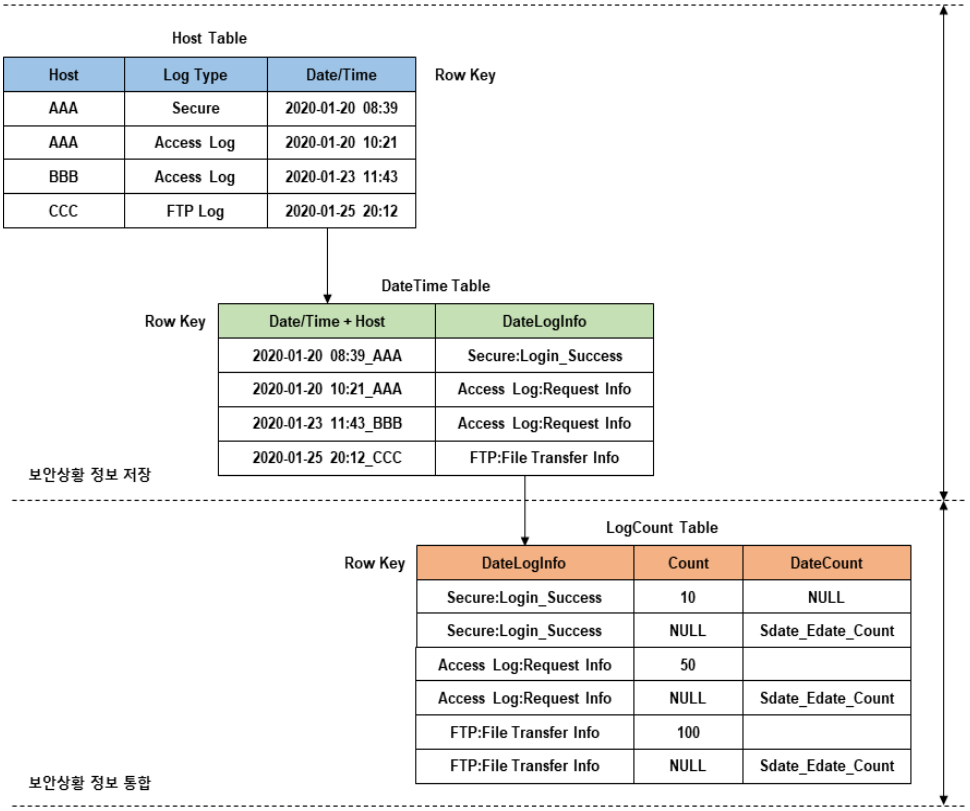
[표 3-5] 보안 로그 필드 구조

- SysLog :
  DateTime : Hostname : Daemon Name[Daemon Process ID]: Log Info
- WebLog :
  ClientIP : DateTime : Request Info : Return Code : Size
- IDSLog :
  DateTime : Hostname : SIP : DIP : Log Info

보안상황 정보수집 시 보안 로그 데이터 통합 과정은 다음 [표 3-6]과 같다.

[표 3-6] 보안 로그 데이터 통합 과정

1. LogData Set에 있는 공통된 필드 구분 확인. 2. Row 키는 공통된 필드 두 가지를 키로 조인하여 선정. 3. 키 이외의 정보를 입력. 4. 통합 작업을 입력된 보안 로그 정보를 기준으로 진행. 5. 통합 작업 시 지정된 시간(시작 데이터 시간과 끝 데이터 시간)의 기준값을 함께 저장함. 6. 1 ~ 5번의 과정을 반복함.



[그림 3-3] 대용량 보안상황 정보통합

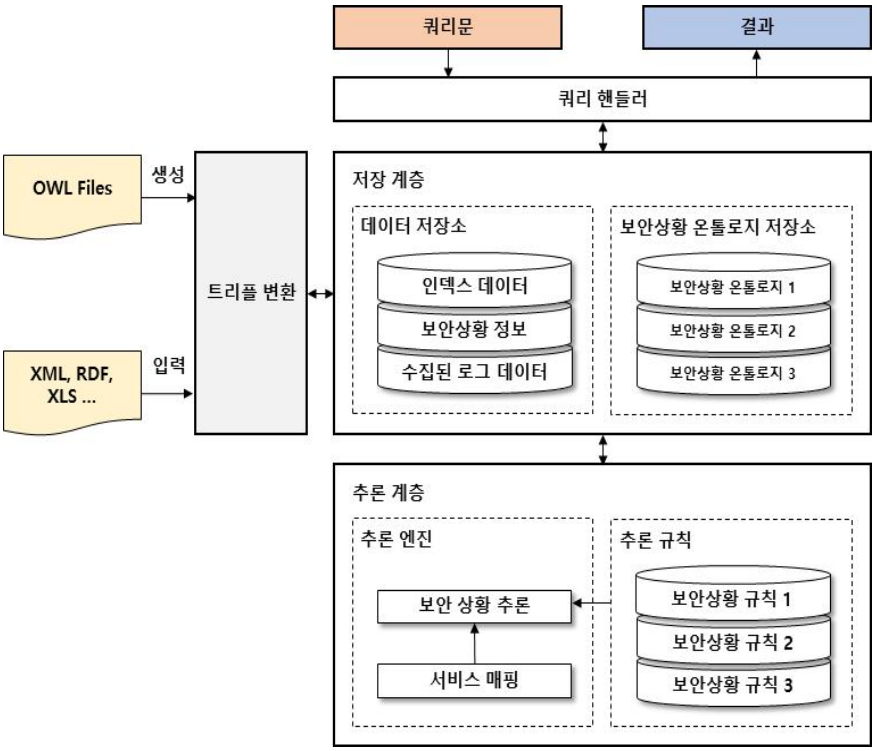
보안상황 정보통합 구조를 살펴보면([그림 3-3]), Host 테이블은 LogData 테이블의 데이터를 식별하기 위한 Row Key로 'Host' 정보가 입력되고, 컬럼 데이터로는 발생한 'Log의 종류', '로그 발생 시간', '횟수'가 입력된다. 발생 시간은 서비스에서 발생하는 로그의 종류를 구분한다. 또한, 로그는 시간의 흐름에 따라서 차례로 데이터가 저장되기 때문에 시간 정보는 침해사고 및 특정일의 운영 상황 분석 시 가장 기본이 되고 중요한 정보이다.

Log의 실제 내용이 저장되는 LogCount 테이블은 'DateTime' + 'Host' + 'Count' 정보를 Row 키로 선정하였다. 이는 데이터 검색 시 '누가', '언제', '발생 횟수' 에 해당하며 호스트와 시간을 구분한다. LogCount 테이블 Column을 살펴보면, 발생하는 로그의 특징에 따라서 Accesslog, Errorlog, Agentlog로 구분하여 데이터를 입력하였고, 로그의 특징에 따라 발생하는 데이터의 고유성을 보존하였다.

4. 보안상황 분석 엔진 설계

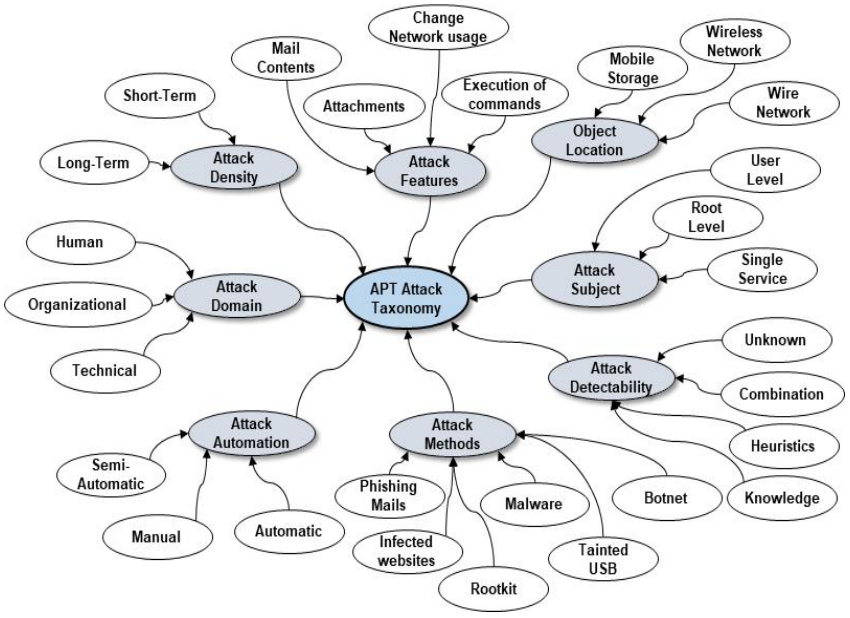
보안상황 분석 엔진은 온톨로지 추론기술을 적용하여 시스템의 보안상황 정보를 수집하고, 분석하는 작업을 수행하게 된다. 이를 위해 보안상황 엔진 설계는 다양한 시스템 환경에서 보안취약점 분석이 가능하고, 변화된 시스템 환경에 적응할 수 있어야 한다. 보안상황 추론 모듈은 추론 엔진(Inference Engine)과 온톨로지 저장소(Ontology Repository)로 구성된다. 먼저 추론 엔진의 구성으로는 트리플 번역기(Triple Translator)와 쿼리 핸들러(Query Handler), 그리고 리즈너(Reasoner)가 속해 있고 온톨로지 저장소와 연동되어 실행된다.

[그림 3-4]에서 보여주는 보안상황 분석 엔진에서는 자바 기반의 룰 엔진인 Jena를 이용하여 온톨로지 저장소를 초기화하거나 쿼리 작업을 수행하였다. 트리플 번역기는 온톨로지 저장소에 OWL 파일을 빌드하고, 입력받은 XML, RDF, XLS 등과 같은 형식의 파일을 트리플 구조로 변경하여 온톨로지 저장소에 저장한다. Reasoner는 사용자가 정한 추론 규칙에 따라 추론을 수행하는 모듈이다. Query Handler는 요청에 따라 SWRL 형식으로 작성되고, select 구문을 이용하여 시스템 환경에서 보안상황을 추론하여 결과를 내보낸다.



[그림 3-4] 보안상황 분석 엔진 구조

보안상황을 인지하고 분석을 위하여 보안상황 추론규칙을 설계하는 것은 매우 중요한 작업이다. 시스템에 가해지는 공격은 다양한 취약점을 이용하여 공격을 수행한다. [그림 3-5]은 시스템 공격 중 대표적인 APT 공격을 모델링한 온톨로지 개념 정의이다.



[그림 3-5] APT 공격 온톨로지 모델링

설계한 보안상황 분석 엔진을 이용하여 APT 공격 상태를 감지하기 위하여 APT 공격 전후의 보안상황 정보를 활용하여 각각의 단계(공격단계, 침투단계, 수집단계)에 대한 추론을 위한 SWRL 언어를 이용하여 규칙을 설계하였다.

APT 공격단계는 공격자가 바이러스를 침투시키는 단계로 공격목표를 지정하고, 목표에 대한 공격 행위를 선택하여, 추론규칙을 통해 공격단계인지 확인할 수 있다. [표 3-7]는 공격단계 탐지를 위한 추론규칙이다.

[표 3-7] 보안상황 정보 온톨로지를 이용한 APT 공격단계 추론 규칙 예

공격 추론 대상	추론 규칙
이메일 공격	(?T behaviour Spearphising_mail)∧(Spearphising resultIn RemoteAccess)∧ (RemoteAccess resultIn System_Control) ⇒ (?T step VariousAttack)
웹 사이트 공격	(?T behaviour InfetedWebsite)∧(InfetedWebsite resultIn RemoteAccess)∧ (RemoteAccess resultIn System_Control) ⇒ (?T step VariousAttack)

APT 침투단계에서는 Backdoor 단계와 Movement 단계로 나누어진다. Backdoor 단계에서는 공격자는 개인의 PC를 장악하고, 통로를 만들어 해당 PC를 제어한다. Movement 단계에서는 공격자가 개인의 PC를 이용하여 상위 목표인 기업 서버로 침투하기 위해 계속적으로 공격을 시도한다. [표 3-8]은 침투단계 탐지를 위한 추론규칙이다.

[표 3-8] 보안상황 정보 온톨로지를 이용한 APT 침투단계 추론규칙 예

공격 추론 대상	추론 규칙
네트워크 트래픽 증가	(?T behaviour RemoteAccess)∧(RemoteAccess resultIn System_Control)∧(System_Control resultIn Networktraffic_increase) ⇒ (?T step Backdoor)
서버 공격	(?T behaviour DenialOfService)∧(DenialOfService resultIn LossOfIntegrity)∧(LossOfIntegrity resultIn MaliciousCodeExecution) ⇒ (?T step Movement)

APT 수집단계의 Data Gathering 단계는 공격자가 서버에 접근하여 공격을 실행하는 단계로, 이 단계에서는 공격자가 기업의 시스템을 파괴하거나 기밀정보를 유출하는 단계이다. [표 3-9]는 수집단계 탐지를 위한 추론규칙이다.

[표 3-9] 보안상황 정보 온톨로지를 이용한 APT 수집단계 추론규칙 예

공격 추론 대상	추론 규칙
서버 접속 후 정보 유출	(?T behaviour Server_Connect)∧(Server_Connect resultIn Information_Leakage) ⇒ (?T step Data Gathering)
서버 접속 후 시스템 파괴	(?T behaviour Server_Connect)∧(Server_Connect resultIn System_Destruction) ⇒ (?T step Data Gathering)

C. 지능형 접근제어 설계

최근 가상화 기술 및 애플리케이션 보안, 대용량 분산 처리 기술 및 트래픽 핸들링, 접근제어, 인증과 같은 다양한 보안 이슈가 발생하고 있다. 특히, 시스템의 다양한 자원에 접속할 경우 통합적인 관리와 제어를 할 수 있는 지능형 접근 제어모델이 필요하다.

접근제어 모델은 역할기반 접근제어(Role-based Access Control, RBAC)나 상황인식 역할기반 접근제어(Context-aware Role-based Access Control, C-RBAC) 모델 등을 이용한 내부자 접근 차단이 대표적이다.

기존 접근제어 모델은 다단계 위임 시 발생 가능한 최소 권한 원칙 및 의무 분리 정책의 위배, 정보 유출 등 다양한 보안 문제에 대해서 효율적인 해결책을 제시하지 못하고 있다. 따라서 이러한 단점을 보완하여 동적이고, 유용한 접근제어 모델이 필요하다.

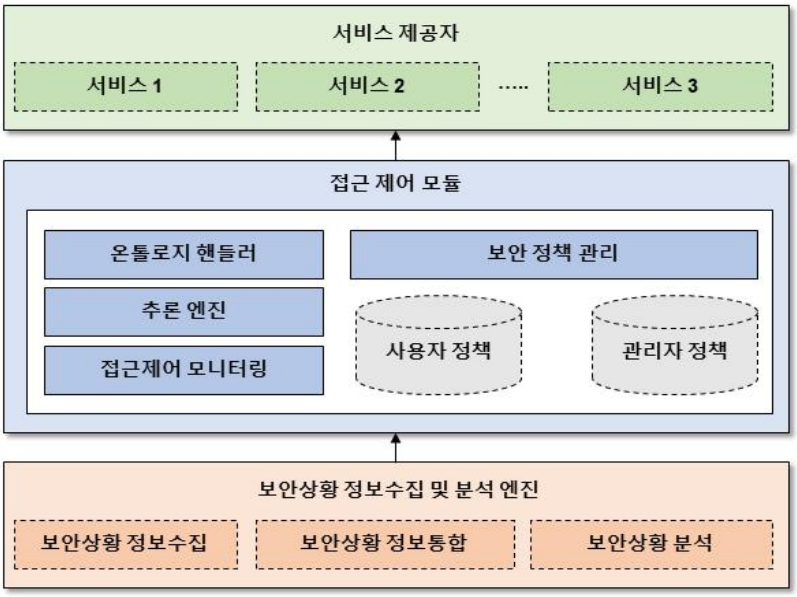
본 논문에서는 역할기반 접근제어 모델을 참고하여 시스템에 적합한 내부 사용자의 상황 정보를 기반으로 자원에 접근하려는 사용자에게 접근 권한을 부여하고, 역할에 배정하며, 사용자가 부여받은 접근 권한으로 서비스를 받을 수 있는 모델을 설계하였다. 이를 위해, 자원에 접근하려는 사용자의 상황 정보와 기존 사용자의 정보를 이용하여 보안상황 정보 온톨로지를 모델링하고, 추론 엔진을 이용하여 자원 접근에 대한 능동적이고 지능화된 상황인지 접근 제어모델을 설계하였다.

1. 지능형 접근제어 프레임워크

a. 전체 프레임워크

시스템에서 동적인 접근제어를 위해서 만족되어야 하는 조건들은 아래와 같다. 리소스 관리자 권한 변경으로 인한 역할 위임 시 부분적으로 위임이 가능해야 하고, 허가 역할 제약과 데이터 접근에 대한 목적이나 조건 등이 고려되어야 한다. 그리고 필요에 따라서는 접근이 거부될 수 있어야 한다. 또한 시간과 장소, 접근 리소스의 특정한 상태에 대한 접근제어가 가능하여야 하고, 시스템 내에서 해킹피해로 인해 악용되거나 권한이 오용되는 것에 대하여 관리자 권한으로 방지할 수 있어야 한다.

지능형 접근제어 프레임워크는 보안상황 인식 기술기반의 애플리케이션이나 시스템에 적합한 보안 구조로써 구성은 [그림 3-6]과 같다. 지능형 접근제어 프레임워크의 구성은 보안상황 정보수집, 분석, 통합기능을 제공하는 보안상황 정보수집 및 분석 엔진과 보안상황 인식 기반으로 관리 가능한 기능을 제공하는 접근제어 모듈 등으로 구성된다.



[그림 3-6] 지능형 접근제어 프레임워크

지능형 접근제어는 내부 사용자가 접근을 요청하면 접근제어 모듈을 통해 사용자 인증 및 접근을 제어하기 위한 보안상황 정보수집 및 분석 엔진에 보안정책과 보안상황 정보를 요구한다. 보안상황 정보수집 및 분석 엔진은 접근제어 및 보안상황 정보와 관련된 보안정책을 제공하며, 수집된 보안상황 정보를 통합, 가공하여 온톨로지 추론 과정을 수행한다.

보안상황 정보수집 및 분석 엔진은 수집한 보안상황 정보를 이용하여 요청한 서비스가 보안 규칙과 상황 조건에 부합한지에 대한 여부를 확인하여 접근을 허가한다. 온톨로지 핸들러는 역할과 보안상황 정보에 따라 접근할 수 있는 모든 정보자원의 위치를 제공한다.

b. 지능형 접근제어 모듈별 기능

보안상황 정보수집 및 분석 엔진은 보안상황 인식 기반의 보안 서비스를 위해 시스템 내 보안상황 정보를 수집 및 관리한다. 보안상황 정보 요청을 받은 후, 정보 수집을 위한 질의 메시지를 만들어 보안상황 정보수집 모듈에 전송하고, 수집된 보안상황 정보를 보안정책에 적용할 수 있는 형식으로 통합한 후, 접근제어 모듈에 전송한다.

접근제어 모듈은 사용자 인증, 접근제어 및 보안상황 정보 등 시스템에서 보안 서비스를 관리한다. 관리 도메인 내에서 보안상황 인식 애플리케이션에 대한 사용 요청이 발생하면, 지능형 접근제어는 보안상황 정보수집 및 분석 엔진을 통해 애플리케이션 사용자의 인증 및 식별, 그리고 접근제어 등과 같은 보안 서비스를 제공하고, 인증과 접근제어 수행을 위해 보안정책과 보안상황 정보수집 및 분석 엔진으로부터 받은 보안상황 정보를 기반으로 보안 서비스를 수행한다.

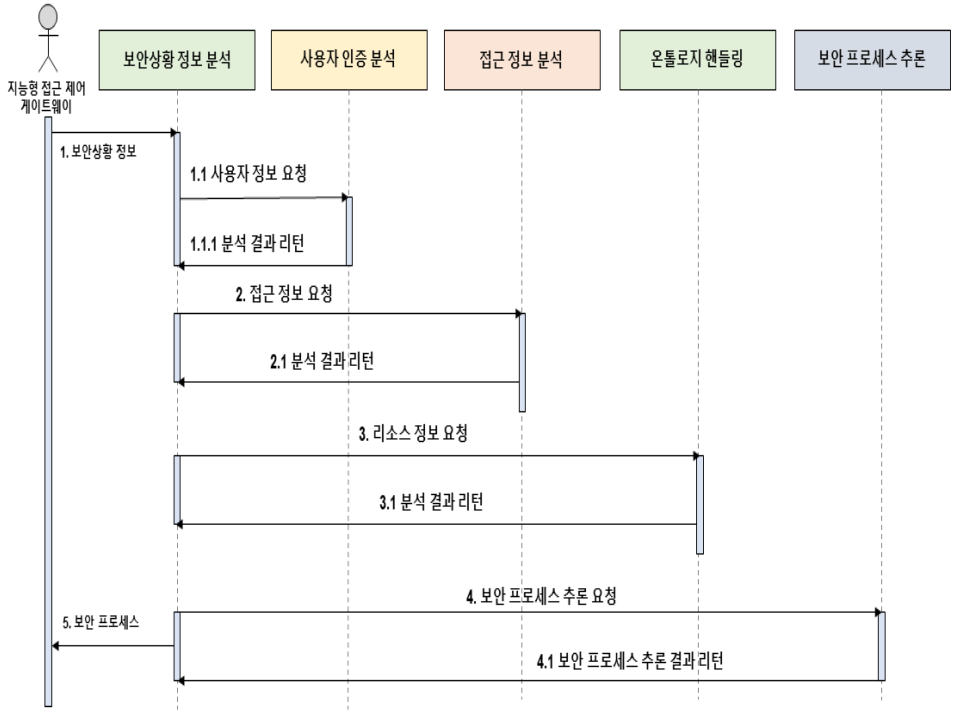
추론 엔진은 보안상황 인식 접근제어 시스템의 핵심인 접근제어 기능을 수행한다. 권한 서비스, 인증서비스, 보안상황 접근제어 온톨로지 저장소 등으로 구성되는데, 각 모듈은 보안정책을 유지 및 관리하고, 사용자의 역할과 상황의 역할을 추론하여 자원 접근 요청에 따라 올바른 접근제어를 수행한다.

보안상황 접근제어 온톨로지 핸들러는 사용자가 접근하려는 리소스의 권한 서비스로부터 데이터 처리를 통하여 분석한 보안상황 정보를 저장하는 보안상황 접근제어 온톨로지를 관리한다. 보안상황 접근제어 온톨로지는 접근 요청이 포함된 트랜잭션 리스트와 트랜잭션의 승인 여부를 나타내는 규칙을 저장한다.

보안정책 관리 모듈은 시스템에 접근하려는 주체에 대한 신원확인과 주체의 보안상황 정보를 관리 및 처리한다. 또한, 리소스에 접근하기 위한 주체의 보안상황 정보에 대하여 접근 위치, 접근시간, 공간적인 영역 등에 대한 추가정보를 획득하고, 접근정책의 분석을 통하여 사용자의 역할을 동적으로 할당하는 서비스를 제공한다. 또한, 현재 활성화된 사용자 역할과 활성화된 보안상황 역할, 그리고 보안정책을 비교하고 분석하여 최적의 접근제어를 결정하는 역할을 한다.

c. 지능형 접근제어 동작 시나리오

[그림 3-7]은 지능형 접근제어 동작 시나리오를 도식화한 것이다.



[그림 3-7] 지능형 접근제어 동작 시나리오

내부 사용자의 리소스 접근을 위한 권한 인증 요청 발생 시 접근제어 모듈에 접근한다. 접근제어 모듈은 내부 사용자 권한 부여를 위해 리소스 접근에 대한 사용자의 보안상황 정보를 보안상황 정보수집 및 분석 엔진이 수집한다. 내부 사용자의 보안상황 정보에 대한 획득정보를 접근제어 모듈에 전달하고, 분석한다.

리소스에 접근하려는 내부 사용자의 보안상황 정보를 획득한 접근제어 모듈은 사용자의 역할할당과 접근제어를 수행하기 위해 보안상황 접근제어 온톨로지 저장소에 접근한다. 보안상황 접근제어 온톨로지 저장소로부터 사용자의 역할 정보와 접근정책 데이터를 요청하고, 접근제어 모듈의 보안정책 관리 모듈은 리소스에 접근하려는 사용자의 역할과 접근정책에 의한 접근 권한을 부여한다. 내부 사용자는 부여받은 역할에 대한 접근 권한을 획득하여 서비스를 요청하고, 서비스를 요청한 접근제어 모듈은 사용자가 요청한 리소스를 부여된 역할과 권한의 수준을 통하여 접근 권한 수준에 적합한 리소스에 접근한다. 보안상황 접근제어 온톨로지는 현재 사용자의 상황과 보안정책 및 권한에 부여된 접근 권한의 수준에 적합한 리소스를 추론을 통해 결정한다.

D. 온톨로지 기반 보안상황 인식서비스 설계

1. 보안상황 온톨로지 추론 및 설계

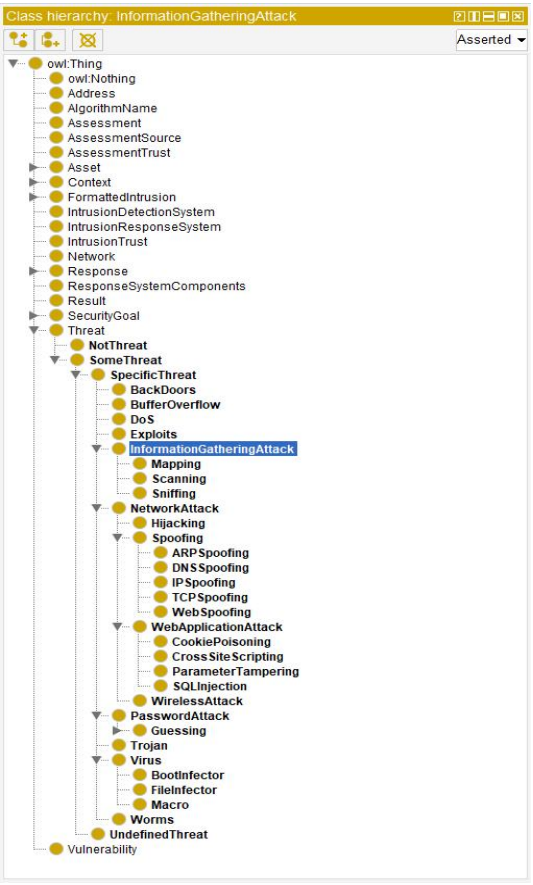
a. 보안상황 온톨로지

시스템에서의 공격 항목은 내부 시스템 침입, 네트워크 및 장치 취약점, 악성코드 감염, 정보 유출 등이 있는데, 이러한 공격은 크게 구조적인 공격, 물리적인 공격, 외부적 공격으로 분류할 수 있다. 구조적인 공격은 시스템의 아키텍처 디자인의 취약점을 이용한 공격으로, 프로토콜, 인증 절차, 시스템 모듈화의 약점을 이용한 공격 등이 있고, 물리적인 공격은 소스코드의 취약점을 이용한 공격으로, SQL 인젝션, 버퍼 오버플로 등을 이용한 공격이고, 외부적인 공격은 공격 대상 이외의 타 프로그램을 이용한 공격으로 바이러스, 웜, 트로이목마 등을 이용한 공격이 여기에 해당된다. 이러한 공격을 분석하여 시스템에서 보안상황 정보를 위한 온톨로지 구축에 활용한다. [표 3-10]는 시스템 보안 요소와 세부 내용이다.

[표 3-10] 시스템 보안 요소와 세부 내용

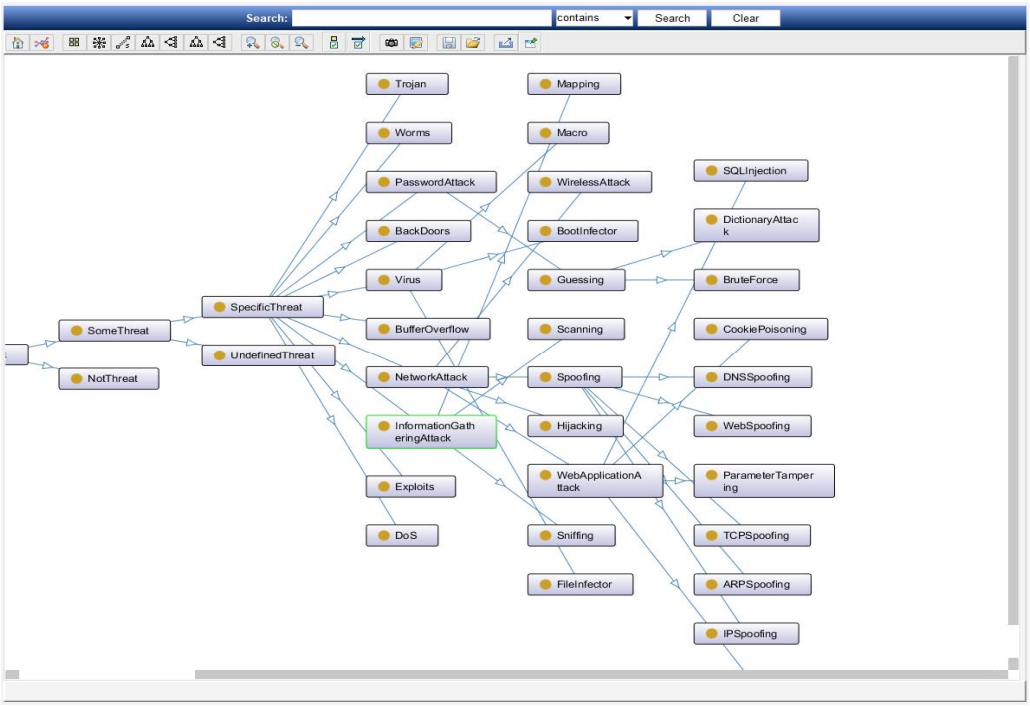
공격 보안 요소	세부 보안 내용
악성 봇 탐지 및 대응	악성 봇 배포서버 차단 및 감염 호스트 치료/격리, 악성 봇 제거, 통신 프로토콜 제어, 행위기반 분석, 좀비 PC 및 C&C서버차단, 이상트래픽 탐지 및 차단, 이상트래픽 관리자 통보
감사 기록	로그 생성 및 접속이력 기록, NTP 서버 및 서버OS 시간정보, 로그 생성 내역별 조회 및 접근 권한 제한, 용량 초과 시 로그 보호 및 관리자 통보, 로그 삭제 및 변경제어
식별 및 인증	관리자 식별 및 인증, 인증 실패 시 일정시간 비활성화 처리, 사용자 인증 실패 시 계정 Lock, 강화된 비밀번호 Role 설정, 비밀번호 입력/변경 시 Masking 처리, 오류정보 제어
전송 데이터 보호	데이터 송·수신 시 암호화 전송, 암호화 관련 프로토콜 안전성 검증
보안관리	보안기능, 보안정책 설정, 관리 서버 접속 IP 제한
에이전트 보호	실행 파일 및 필터 드라이버 무결성, 변조된 정보 복구기능, 부인방지, 무결성 보장

시스템 내 각종 보안시스템의 로그 구조 분석을 통하여 보안상황 정보를 획득한 후, 온톨로지의 규칙기반 추론 엔진이 추론을 수행하는 과정을 거쳐 시스템 보안상황 인식서비스를 제공한다. 정보들의 일반화와 정보들 사이의 관계 정의를 위해 온톨로지의 하위수준의 정보에서 상위수준의 정보를 추출한다. ([그림 3-8], [그림 3-9], [그림 3-10])는 보안상황 온톨로지의 요소들을 클래스로 표현하여 OWL의 계층 구조로 표현한 것이다.



[그림 3-8] 보안상황 공격 분류 클래스

[그림 3-9] 보안상황 공격 Object Property



[그림 3-10] 보안상황 온톨로지 공격 분류 클래스 관계도

명확한 정보를 표현하기 위해 구성요소를 정의하고 구성요소들 사이의 관계를 OWL의 속성(Property)으로 정의하여 표현하였다. OWL의 속성은 관계를 의미하고, 속성의 표현은 두 Individual 사이의 관계를 의미하는 Object Properties와 Individual과 DataValue 사이의 관계를 표현하는 Datatype Properties로 구분된다.

b. 보안상황 온톨로지 기반 추론 설계

시스템의 보안상황 인식서비스의 Class, Property, individual 사이의 규칙기반 추론을 위해 SWRL을 이용한다. 이것은 새로운 지식을 얻기 위해 온톨로지에서 정보의 관계들의 규칙을 정의하는 것으로, 실행에는 Jess 엔진을 활용하였다. 보안상황 인식을 위한 규칙을 추론하기 위해 보안상황 온톨로지를 참고하여 [표 3-11]과 같은 규칙을 정의하였다.

[표 3-11] 보안상황 인식을 위한 온톨로지 추론 규칙(SWRL)

추론 내용	온톨로지 추론규칙 SWRL
서비스 거부 공격탐지	AutomatedIntrusionResponseSystem(?IRS), DoS(?threatdos), NetworkContext(?netcontext), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?netcontext, ?netdate), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), networkAnomaly(?netcontext, ?netan), systemLatency(?syscontext, ?syslatency), equal(?netdate, ?intdate), equal(?sysdate, ?intdate), greaterThan(?netan, 7), greaterThan(?syslatency, 7) -> indicates(?syscontext, ?threatdos)
침임 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), realIntrusionImpact(?intrusion, ?intimpact), responseComplexity(?respon1, ?respcomplex1), responseComplexity(?respon2, ?respcomplex2), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), equal(?aloi, "low"), equal(?respcost1, ?respcost2), greaterThan(?numOpResp, 1), lessThan(?respcomplex1, ?respcomplex2) -> optimumResponse(?intrusion, ?respon1)
침입 경고 신뢰성	NotThreat(?threatcon), SystemContext(?context), indicates(?syscontext, ?threatcon), isGeneratedBy(?intrusion, ?ids), receivedFormattedIntrusion(?IRS, ?intrusion), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "low"), equal(?status, "Pending"), equal(?sysdate, ?intdate) -> intrusionAlertReliability(?intrusion, "low")
백도어 공격탐지	AutomatedIntrusionResponseSystem(?IRS), BackDoors(?threatbackdoors), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemActiveProcesses(?syscontext, ?sysprocesses), systemNumberOfUsersLogged(?syscontext, ?sysUsers), equal(?sysdate, ?intdate), greaterThan(?sysUsers, 5), greaterThan(?sysprocesses, 5) -> indicates(?syscontext, ?threatbackdoors)
침임 탐지 최적 응답	FormattedIntrusion(?intrusion), Passive(?passive), receivedFormattedIntrusion(?irs, ?intrusion), recommendedResponses(?intrusion, ?response), numberOfRecommendedResponses(?intrusion, ?numResp), greaterThan(?numResp, 0), SameAs (?response, ?passive) -> optimumResponse(?intrusion, ?response)
침임 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?component, ?aloi), intrusionSeverity(?intrusion, ?intseverity), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), responseSeverity(?respon1, ?respseverity1), responseSeverity(?respon2, ?respseverity2), equal(?aloi, "medium"), greaterThan(?numOpResp, 1), greaterThan(?respseverity1,

	?intseverity), greaterThan(?respseverity2, ?intseverity), lessThan(?respcost1, ?respcost2) -> optimumResponse(?intrusion, ?respon1)
웜 바이러스 공격탐지	AutomatedIntrusionResponseSystem(?IRS), SystemContext(?syscontext), Trojan(?threattrojan), Virus(?threatvirus), Worms(?threatworms), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemActiveProcesses(?syscontext, ?sysprocesses), equal(?sysdate, ?intdate), greaterThan(?sysprocesses, 7) -> indicates(?syscontext, ?threattrojan), indicates(?syscontext, ?threatvirus), indicates(?syscontext, ?threatworms)
패스워드 탈취 공격	AutomatedIntrusionResponseSystem(?IRS), PasswordAttacks(?threatpassword), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemCPUUsage(?syscontext, ?syscpu), systemLatency(?syscontext, ?syslatency), equal(?sysdate, ?intdate), greaterThan(?syscpu, ?7), greaterThan(?syslatency, 7) -> indicates(?syscontext, ?threatpassword)
침입 경고 신뢰성	Active(?response1), Active(?response2), SpecificThreat(?threatsp), SystemContext(?syscontext), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatCont), isGeneratedBy(?intrusion, ?ids), protects(?response1, ?sgres1), protects(?response2, ?sgres2), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), threatens(?threatCont, ?sgintCont), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "medium"), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres1, ?sgint), SameAs (?sgres2, ?sgintCont), SameAs (?threatCont, ?threatsp), DifferentFrom (?threatCont, ?threat) -> recommendedResponses(?intrusion, ?response1), recommendedResponses(?intrusion, ?response2), intrusionAlertReliability(?intrusion, "medium")
침입 경고 신뢰성	Active(?response), SpecificThreat(?threatsp), SystemContext(?syscontext), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatCont), isGeneratedBy(?intrusion, ?ids), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "high"), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint), SameAs (?threatCont, ?threatsp), DifferentFrom (?threatCont, ?threat) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "medium")
침입 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?component, ?aloi), intrusionSeverity(?intrusion, ?intseverity), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseComplexity(?respon1, ?respcomplex1), responseComplexity(?respon2, ?respcomplex2), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), responseSeverity(?respon1, ?respseverity1), responseSeverity(?respon2,

	?respseverity2), equal(?aloi, "medium"), equal(?respcost1, ?respcost2), greaterThan(?numOpResp, 1), greaterThan(?respseverity1, ?intseverity), greaterThan(?respseverity2, ?intseverity), lessThan(?respcomplex1, ?respcomplex2) -> optimumResponse(?intrusion, ?respon1)
침입 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), intrusionSeverity(?intrusion, ?intseverity), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), responseSeverity(?respon1, ?respseverity1), responseSeverity(?respon2, ?respseverity2), equal(?aloi, "high"), equal(?respseverity1, ?respseverity2), greaterThan(?numOpResp, 1), greaterThanOrEqual(?respseverity1, ?intseverity), greaterThanOrEqual(?respseverity2, ?intseverity), lessThanOrEqual(?respcost1, ?respcost2) -> optimumResponse(?intrusion, ?respon1)
알려지지 않은 공격탐지	AutomatedIntrusionResponseSystem(?IRS), SystemContext(?syscontext), UndefinedThreat(?threatunde), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemStatus(?syscontext, ?sysstatus), equal(?sysdate, ?intdate), greaterThan(?sysstatus, 7) -> indicates(?syscontext, ?threatunde)
침입 경고 신뢰성	Active(?response), AutomatedIntrusionResponseSystem(?IRS), SystemContext(?syscontext), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatCont), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint), SameAs (?threat, ?threatCont) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "high")
침입 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), intrusionSeverity(?intrusion, ?intseverity), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseSeverity(?respon1, ?respseverity1), responseSeverity(?respon2, ?respseverity2), equal(?aloi, "high"), greaterThan(?numOpResp, 1), greaterThan(?respseverity1, ?respseverity2), greaterThanOrEqual(?respseverity1, ?intseverity), greaterThanOrEqual(?respseverity2, ?intseverity) -> optimumResponse(?intrusion, ?respon1)
침입 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), equal(?aloi, "low"), greaterThan(?numOpResp, 1), lessThan(?respcost1, ?respcost2) -> optimumResponse(?intrusion, ?respon1)

침입 탐지 최적 응답	FormattedIntrusion(?intrusion), hasTarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), equal(?aloi, "low"), greaterThan(?numOpResp, 1), lessThan(?respcost1, ?respcost2) -> optimumResponse(?intrusion, ?respon1)
침입 탐지 최적 응답	FormattedIntrusion(?intrusion), potentialOptimumResponses(?intrusion, ?response), receivedFormattedIntrusion(?irs, ?intrusion), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), equal(?numOpResp, 1) -> optimumResponse(?intrusion, ?response)
침입 경고 신뢰성	Active(?response), NotThreat(?threatcon), SystemContext(?context), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatcon), isGeneratedBy(?intrusion, ?ids), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "medium"), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "low")
침입 경고 신뢰성	Active(?response), NotThreat(?threatcon), SystemContext(?syscontext), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatcon), isGeneratedBy(?intrusion, ?ids), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "high"), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "medium")
침입 경고 신뢰성	Active(?response), AutomatedIntrusionResponseSystem(?IRS), SystemContext(?syscontext), UndefinedThreat(?threatunde), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatCont), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint), SameAs (?threatunde, ?threatCont), DifferentFrom (?trheatCont, ?threat) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "high")
침입 탐지 최적 응답	Active(?respactive), FormattedIntrusion(?intrusion), receivedFormattedIntrusion(?irs, ?intrusion), recommendedResponses(?intrusion, ?response), maximumResponseComplexity(?irs, ?irscomplex), realIntrusionImpact(?intrusion, ?intimpact), responseComplexity(?response, ?rescomplex), responseImpact(?respon1, ?resimpact), lessThanOrEqual(?rescomplex, ?irscomplex), lessThanOrEqual(?resimpact,

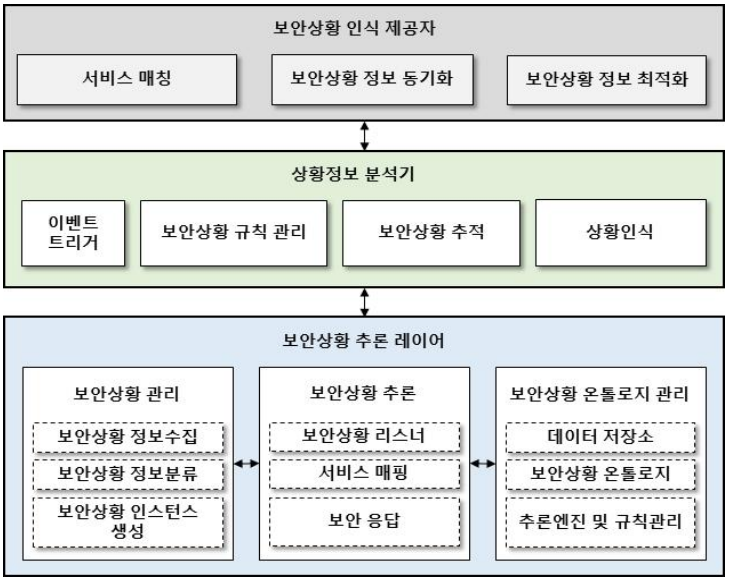
	?intimpact), SameAs (?response, ?respactive) -> potentialOptimumResponses(?intrusion, ?response)
침입 탐지 추천 응답	AutomatedIntrusionResponseSystem(?IRS), Recovery(?response), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemFreeSpace(?syscontext, ?syshard), equal(?sysdate, ?intdate), greaterThan(?syshard, 7) -> recommendedResponses(?intrusion, ?response)
패스워드 탈취 공격	AutomatedIntrusionResponseSystem(?IRS), PasswordAttacks(?threatpassword), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemSSHFailed(?syscontext, ?sysssh), equal(?sysdate, ?intdate), greaterThan(?sysssh, ?7) -> indicates(?syscontext, ?threatpassword)
유사행위 침입탐지	FormattedIntrusion(?intrusion), Result(?result), hasIntrusionType(?intrusion, ?threat), hasIntrusionType(?intrusionres, ?threatres), hasTarget(?intrusion, ?target), hasTarget(?intrusionres, ?targetres), receivedFormattedIntrusion(?irs, ?intrusion), relatedIntrusion(?result, ?intrusionres), assetLevelOfImportance(?target, ?targetimp), assetLevelOfImportance(?targetres, ?targetresimp), numberOfGeneratedResult(?irs, ?numRes), responseStatus(?intrusion, ?intstatus), equal(?intstatus, "Pending"), equal(?targetimp, ?targetresimp), SameAs (?threat, ?threatres) -> hasSimilarResult(?intrusion, ?result)
결과에 대한 권장 추론	Result(?result), hasSimilarResult(?intrusion, ?result), responseResult(?result, ?resresult), responseStatus(?intrusion, ?respStatus), equal(?respStatus, "Pending"), equal(?resresult, "Satisfactory") -> neededRecommendedInference(?intrusion, false)
침입 탐지 추천 응답	AutomatedIntrusionResponseSystem(?IRS), Passive(?response), receivedFormattedIntrusion(?IRS, ?intrus1) -> recommendedResponses(?intrus1, ?response)

상황인식 단계는 다음과 같은 3가지로 구성된다. 먼저 식별(Perception)은 전체 상황을 인지하기 위한 준비 단계로써 여러 시스템에서 발생하는 이벤트를 수신하여 저장하며 발생한 이벤트와 시스템에 대한 정보를 갖고 있다. 다음으로 이해(Comprehension)는 시스템의 전체 상황 표현이 가능한 단계로 발생한 이벤트들의 통합 및 재배치, 연관성 분석 등을 수행하며 피해 시스템, 취약 시스템과 피해 정도를 비교적 직관적으로 표현한다. 예측(Projection)은 시스템 상의 피해 상태나 취약 정도를 기반으로 어떻게 대응해야 하는지에 대한 결정을 돕는 단계이다. 발생한 상황들을 토대로 공격 대상의 가능성이 있는 시스템 파악 및 피해 가능성을 예측하는 동시에 보안 방안에 대한 결정에 도움이 되는 자료를 생성한다.

2. 보안상황 인식서비스 프레임워크

a. 전체 프레임워크

[그림 3-11]는 시스템 보안상황 인식서비스 프레임워크의 전체 구성도이다. 설계한 시스템은 3개의 계층으로 나누어져 있고, 각 계층의 상세한 내용은 다음과 같다.



[그림 3-11] 시스템 보안상황 인식서비스 프레임워크

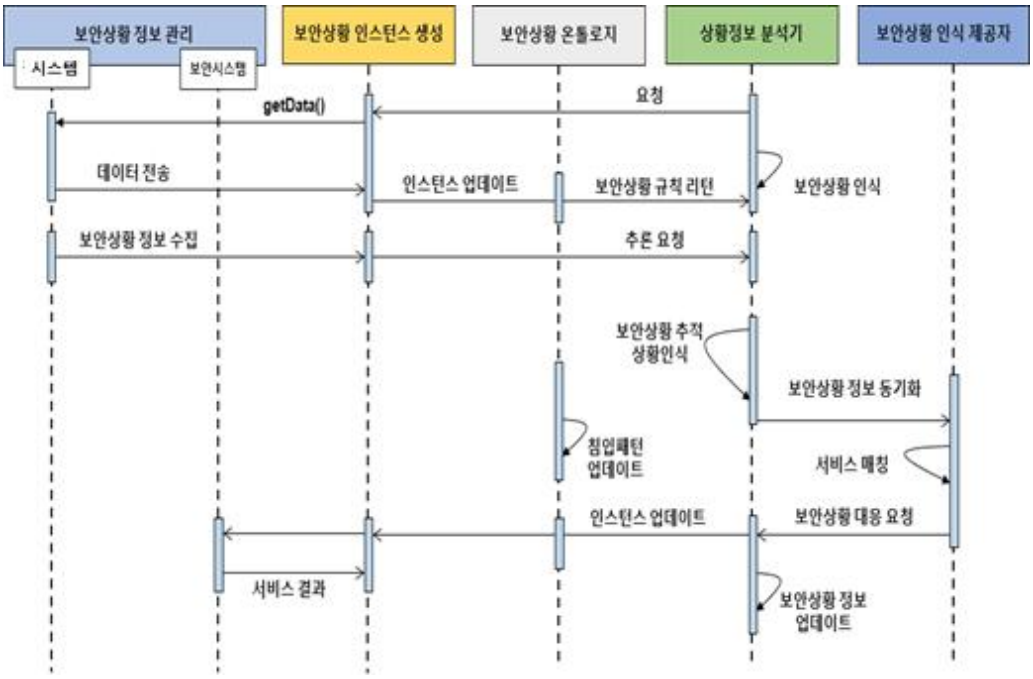
보안상황 추론 레이어는 보안상황 정보를 수집하고, 분류하여, 보안상황 인스턴스 정보를 생성한다. 전달된 보안상황 정보를 활용하여 보안상황을 추론하는 역할을 한다. 이를 위해 보안상황 추론 모듈을 사용한다. 보안상황 추론 모듈은 보안상황을 판단하기 위해 온톨로지에 정의된 정보를 활용한다. 보안상황 온톨로지 관리 모듈은 보안상황 정보를 수집하는 모듈로써, 보안상황 관리 모듈에서 수집한 보안상황 정보를 저장한 후, 보안상황 추론 모듈에서 보안상황 이벤트를 발생시킨다.

추론 엔진 및 규칙관리 모듈은 보안상황 온톨로지를 추론하기 위한 쿼리 생성과 보안상황 온톨로지를 OWL 언어로 변환한다. 보안상황 추론 모듈은 전송받은 보안상황 정보를 이용하여 온톨로지 추론이 가능한 쿼리 형태로 변환하여 보안상황을 추론한다. 보안상황 정보를 추론한 후, 보안상황을 보안상황 인식 제공자 레이어에 제공하여 보안상황에 대한 대응 방법을 제공한다.

보안상황 인스턴스 생성 모듈은 보안상황 정보 수지 및 부류 모듈에 의해 생성된 하위 레벨의 보안상황 정보를 보안상황 온톨로지의 인스턴스로 변환한다. 생성한 인스턴스를 온톨로지 저장소에 저장한 후, 보안상황 추론 모듈에 의해 상황 인식 추론이 진행된다. 보안상황 인스턴스 생성 모듈은 보안상황 서비스 매핑 모듈을 통해 상황에 적합한 도메인 및 상위 온톨로지 모델을 적용하고, 보안상황 인스턴스 생성 모듈을 통해 적용한 보안상황 인스턴스를 생성한다. 보안상황 온톨로지 관리 모듈은 데이터저장소에 보안상황 온톨로지 모델 스키마 정보와 인스턴스, 보안상황 규칙 등이 저장된다.

b. 보안상황 인식서비스 동작 시나리오

[그림 3-12]은 시스템 보안상황 인식서비스 프레임워크의 컴포넌트 간 동작 시나리오다. 내부 시스템 및 보안시스템에서 수집된 보안상황 정보를 보안상황 추론 레이어의 보안상황 관리 모듈에 전달한다. 보안상황 관리는 수집된 보안상황 정보를 보안상황 인스턴스 생성 모듈에서 사용할 수 있는 형태로 변환한다.



[그림 3-12] 시스템 보안상황 인식서비스 동작 시나리오

[표 3-12] 보안상황 인스턴스 모듈 중 서브 네트워크 정보를 추출 저장하는 모듈

```
public String obtainSubNetworkInfo(String hostIP) throws SQLException, DAOException {
        connectDataSource();
        String hostIPaddress=hostIP;
        String subnetworkName = null;

        String select = "SELECT location FROM "+network_assets_table+ " WHERE
ipAddress='"+hostIPaddress+"'";
        System.out.println(select);
        try {
            Statement stmt = conNet.createStatement();
            ResultSet rs = stmt.executeQuery(select);
            while (rs.next()) {
                subnetworkName = rs.getString("location");
            }
            rs.close();
            stmt.close();
        } catch (SQLException e) {
            throw new DAOException(e);
        }
        return subnetworkName;
    }
```

보안상황 정보수집 및 분류 모듈로부터 전달받은 보안상황 정보는 보안상황 인스턴스 생성 모듈을 통해 새로운 보안상황 인스턴스를 생성한다. 변환된 보안상황 인스턴스는 상황 추론을 위해 보안상황 추론 모듈로 전달되고, 동시에 내부 보안상황 온톨로지 관리 모듈에 의해 업데이트된다.

[표 3-13] 보안상황 추론 중 유사침입을 추론 및 탐지하는 모듈

```
boolean checkSimilarIntrusion(IntrusionAlert map){
        synchronized (ontology_infered_instances) {

System.out.println("_____________________________________________________________________");
            System.out.println( Thread.currentThread() + "-------유사 침입 판단 여부
체크-----------");
            String sameAlertDate= null;
            String existingIntrusionDate = null;
            ArrayList existingIntrusionIndi = new ArrayList();
            try{
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
                long time =(simpleDateFormat.parse(map.getIntDetectionTime())).getTime();
                DateToXsdDatetimeFormatter xdf = new DateToXsdDatetimeFormatter();
                sameAlertDate = xdf.format(new Date(timesamealert));
                existingIntrusionDate = xdf.format(new Date(timeexistingalert));
            }
            catch(Exception e){
            }
```

```
String sameAlert = "PREFIX individual: <"+PREFIXIND+">" +
        " PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
        " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"+
        " PREFIX owl: <http://www.w3.org/2002/07/owl#>"+
        " PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"+
        " SELECT DISTINCT ?formattedintrusion" +
        " WHERE {" +
          "?threat a individual:"+map.getIntType()+ "."+
          "?component a individual:SystemComponent ."+
          "?component individual:hasAddress ?address ."+
          "?address individual:addressIP ?ip ."+
          "FILTER (?ip = \""+map.getIntrusionTarget().getAddressIP()+"\")."+
          "?formattedintrusion individual:intrusionSeverity ?severity."+
          "FILTER (?severity ="+map.getIntSeverity()+") ."+
          "?formattedintrusion individual:portDest ?portd ."+
          "FILTER (?portd = "+map.getPortDest()+ " )."+
          "?formattedintrusion individual:portSrc ?ports ."+
          "FILTER (?ports = "+map.getPortSrc()+ ") ."+
          "?formattedintrusion individual:sourceOfIntrusionID ?source ."+
          "FILTER (?source =  \""+map.getSourceIP()+"\") ."+
          "?formattedintrusion individual:intrusionDetectionTime ?idt ."+
          "FILTER (xsd:dateTime(?idt) >= \""+sameAlertDate+"\"^^xsd:dateTime)
."+
          "?formattedintrusion individual:hasIntrusionType ?threat;"+
          "individual:hasTarget ?component ."+
        " } ";
String existingIntrusion = "PREFIX individual: <"+PREFIXIND+">" +
         " PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"+
         " SELECT DISTINCT ?formattedintrusion" +
         " WHERE {" +
           "?threat a individual:"+map.getIntType()+ "."+
           "?component a individual:SystemComponent ."+
           "?component individual:hasAddress ?address ."+
           "?address individual:addressIP ?ip ."+
           "FILTER (?ip = \""+map.getIntrusionTarget().getAddressIP()+"\")."+
           "?formattedintrusion individual:intrusionDetectionTime ?idt ."+
           "FILTER (xsd:dateTime(?idt) >=
\""+existingIntrusionDate+"\"^^xsd:dateTime) ."+
           "?formattedintrusion individual:hasIntrusionType ?threat;"+
           "individual:hasTarget ?component ."+
         " } ";
inferedindividualModel =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
inferedindividualModel.read(ontology_infered_instances);
```

보안상황 추론 모듈 보안상황 인스턴스 생성 계층에서 전달된 보안상황 인스턴스와 패턴을 기반으로 시스템 내 보안상황을 추론하고, 상황에 맞는 적절한 서비스를 결정한다. 보안상황 인식 제공자 계층은 상황 정보 분석기 계층을 통해 전달된 보안상황 추론 결과를 통해 시스템의 보안상황을 모니터링하고, 결과를 상황 정보 분석기 계층에게 전달하여 해당 보안상황 정책에 따라 보안상황 대응을 요청한다.

Ⅳ. 실험 및 결과

본 장에서는 보안 상황인식을 위한 지능형 접근제어를 위해 실험의 성능평가 항목 및 지표, 데이터셋 그리고 보안상황 접근제어 온톨로지 기반 접근제어 추론, 보안상황 인식서비스의 공격탐지 추론 마지막으로 지능형 통합 접근제어 시스템의 공격 대응 실험내용 및 결과에 대해 설명한다.

A. 성능평가 항목 및 지표

본 연구의 성과목표 평가를 위해 평가항목을 다음과 같이 설정하였다.

보안상황 온톨로지 기반 접근제어 추론 정확률은 정의된 접근제어 추론 규칙을 기반으로 시스템 접근 권한획득 정확률을 평가한다. 보안상황 인식서비스의 공격탐지 추론 정확률은 보안상황 인식서비스를 이용한 시스템 공격탐지 추론능력을 평가한다. 지능형 통합 접근제어 시스템의 공격 대응 정확률은 시스템 보안공격 발생 시 지능형 통합 접근제어 시스템에서 최적의 대응방안을 제시할 수 있는 추론능력을 평가하였다.

추론 정확성과 공격 대응 정확성을 측정하기 위해 결과에 대해 정확도, 정밀도, 재현율을 사용하여 평가에 활용하였다.

정확도는 옳게 예측한 비율로 (수식 1)로 정의됐다. TP(True Positive)는 양성(Positive) 데이터를 양성으로 올바르게 판별한 경우의 수이며, TN(True Negative)는 음성(Negative) 데이터를 음성으로 올바르게 판별한 경우의 수이다. FP(False Positive)는 양성 데이터를 음성 데이터라고 부정확하게 판별한 경우의 수이며, FN(False Negative)는 양성 데이터를 음성 데이터라고 판별한 경우의 수를 말한다.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad \text{(수식 1)}$$

정밀도는 양성으로 예측한 데이터 중에 실제 양성 데이터가 포함된 비율로 분류 모델이 얼마나 정확하게 양성 데이터를 분류했는지에 대한 지표로 (수식 2)로 정의된다.

$$Precision = \frac{TP}{TP+FP} \quad \text{(수식 2)}$$

재현율은 민감도(Sensitivity)라고도 하며 실제 양성 데이터 중에 양성으로 예측한 비율로 분류 모델이 얼마나 민감하게 양성 데이터를 분류했는지 보여주는 지표로 (수식 3)으로 정의된다.

$$Recall = \frac{TP}{TP+FN} \quad \text{(수식 3)}$$

B. 데이터 셋

다양한 보안이벤트가 발생할 때, 종합적인 보안 상황을 판단할 수 있는지에 대한 모의실험은 보안상황 정보 레코드 200개를 이용하여 공격 이벤트를 발생시키는 것으로 진행하였다. 공격 이벤트는 Win32 API 행위 정보를 기반으로 악성코드와 정상 파일이 실행되는 동안에 발생하는 행위 패턴 데이터를 기반으로 수집하였다. 이를 위해 악성코드 생성기에서 생성한 500개의 악성코드와 400개의 정상 파일을 실행하여 특정 이벤트가 발생할 때마다 Win32 API 행위 패턴을 수집하였다. 이 방법을 선택한 이유는 본 연구에서 제시한 백도어 공격탐지, 패스워드 탈취 공격, 서비스 거부 공격탐지 등 다양한 보안 상황을 발생시키기 위함이다.

[표 4-1] 실험 데이터 셋

Dataset	정상 행위 시퀀스 집합 크기	악성 행위 시퀀스 집합 크기	악성 코드 포함 비율	전제 레코드 수
data-0.1	8000	120	0.1	9200
data-0.2	8000	120	0.2	10400
data-0.3	8000	120	0.3	11600
data-0.4	8000	120	0.4	12800
data-0.5	8000	120	0.5	14000

API 호출 함수를 기반으로 각 프로세스가 수행하는 행위 패턴의 기본 요소를 기반으로 해당 시스템 내에서 행위 패턴에 해당하는 프로세스가 발생하면 호스트 ID, 사용자 ID, 프로세스 ID, 발생 시간, 파일이름 등의 정보가 수집되고, [표 4-2]와 같은 구조로 저장하였다.

[표 4-2] 프로세스 수집 정보 예

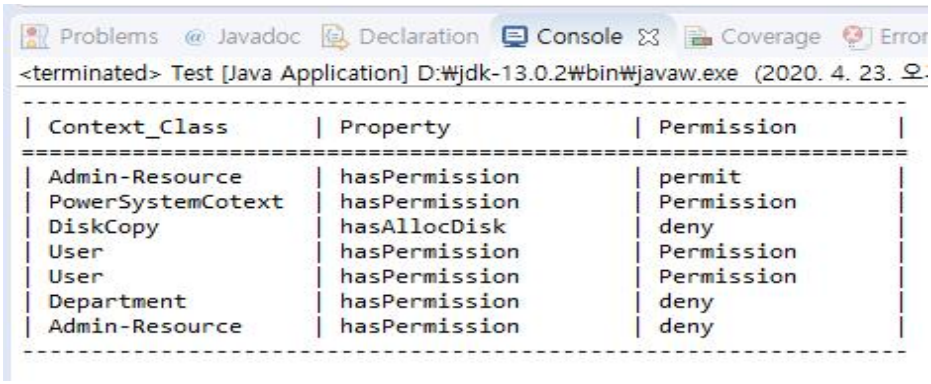
index	HostID	UserID	PID	Time	Filename	Feature Index	Result	Parameter
20343	1	admin	1233	2020-02-22 18:34:54,319	chrome.exe	12 (OpenThread)	SUCCESS	Thread ID : 5212
21012	1	admin	1233	2020-02-22 18:34:54,468	chrome.exe	4 (OpenProcess)	SUCCESS	
…	…	…	…	…	…	…	…	…
129332	1	admin	2631	2020-02-22 18:35:21,461	iexplore.exe	1 (CreateProcess)	SUCCESS	
129333	1	admin	3217	2020-02-22 18:35:21,487	iexplore.exe	13 (SuspendThread)	SUCCESS	

이러한, 과정을 통해 백도어 공격탐지, 패스워드 탈취 공격, 서비스 거부 공격탐지 등의 공격 이벤트를 발생시키고, 보안 상황별로 정의된 추론 규칙을 확장하여 실험하였다.

C. 실험 평가 및 분석

1. 보안상황 온톨로지 기반 접근제어 추론 정확률

보안상황 온톨로지 기반 접근제어의 접근제어 추론 정확성을 검증하기 위해 악성코드 탐지 시 지능형 접근제어 모델이 적절한 권한 설정을 부여했는지에 대한 결과를 확인하였다. 확인을 위해 내부 등급별 요청 권한 설정을 통해 제안된 지능형 접근제어 모델을 적용했다. 보안상황 접근제어 온톨로지의 관계 설정을 통하여 추론을 통해 시스템의 리소스 권한 정보를 [그림 4-1]과 같이 확인하였다.



[그림 4-1] 추론규칙 적용 결과

[표 4-3]은 제안된 추론 기반 지능형 접근 제어모델을 적용한 악성코드 탐지 결과에 따른 올바른 리소스 권한을 설정한 결과이다. 악성코드 탐지 시 정보 유출 행위 탐지 수행 결과, 보안상황별로 보안상황 접근제어 온톨로지 추론을 통해 적합한 접근 권한 설정을 적절하게 수행하였음을 확인하였다. 악성코드 탐지 시 접근제어 추론을 이용한 올바른 권한 설정률은 평균 87%이다.

[표 4-3] 악성 행위 탐지 시 지능형 접근제어의 접근 권한 설정 결과

악성행위 개수	악성코드 스캐너 분석도구	탐지 비율	올바른 접근 권한 설정률
50	42	74%	82%
100	83	83%	83%
500	437	87.4%	89.4%
1000	873	87.3%	91.7%

2. 보안상황 인식서비스의 공격탐지 추론 정확률

모의 해킹을 통해 보안상황 인식서비스의 보안 공격탐지 추론 정확률을 측정하였다. 보안상황 온톨로지와 추론 엔진을 통해 공격탐지 추론 정확률을 검증하기 위해서 임의의 보안상황 정보 레코드 200개를 이용하여 공격탐지의 정확률, 재현율, 신뢰도를 평가하였다. [표 4-4]은 질의 종류와 사용된 추론 규칙 개수이다.

[표 4-4] 보안상황별 공격탐지를 위한 질의 종류

보안상황	질의	추론규칙 개수
백도어 공격탐지	AutomatedIntrusionResponseSystem(?IRS), BackDoors(?threatbackdoors), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemActiveProcesses(?syscontext, ?sysprocesses), systemNumberOfUsersLogged(?syscontext, ?sysUsers), equal(?sysdate, ?intdate), greaterThan(?sysUsers, 5), greaterThan(?sysprocesses, 5) -> indicates(?syscontext, ?threatbackdoors)	4
패스워드 탈취 공격	AutomatedIntrusionResponseSystem(?IRS), PasswordAttacks(?threatpassword), SystemContext(?syscontext), receivedFormattedIntrusion(?IRS, ?intrusion), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), systemCPUUsage(?syscontext, ?syscpu), systemLatency(?syscontext, ?syslatency), equal(?sysdate, ?intdate), greaterThan(?syscpu, ?7), greaterThan(?syslatency, 7) -> indicates(?syscontext, ?threatpassword)	10
유사행위 침입탐지	FormattedIntrusion(?intrusion), Result(?result), hasIntrusionType(?intrusion, ?threat), hasIntrusionType(?intrusionres, ?threatres), hasTarget(?intrusion, ?target), hasTarget(?intrusionres, ?targetres), receivedFormattedIntrusion(?irs, ?intrusion), relatedIntrusion(?result, ?intrusionres), assetLevelOfImportance(?target, ?targetimp), assetLevelOfImportance(?targetres, ?targetresimp), numberOfGeneratedResult(?irs, ?numRes), responseStatus(?intrusion, ?intstatus), equal(?intstatus, "Pending"), equal(?targetimp, ?targetresimp), SameAs (?threat, ?threatres) -> hasSimilarResult(?intrusion, ?result)	15

[표 4-5]은 보안상황별 공격탐지 질의의 실험 결과를 나타낸다. TP(True Positive)는 추론 결과가 올바르게 나온 값이고, FN(False Negative)는 추론에 실패한 값을 의미하며 FP(False Positive)는 추론은 됐지만 잘못 추론된 값이다.

[표 4-5] 보안상황별 공격탐지 실험 결과 1

보안상황	추론 개수	TP	FN	FP
백도어 공격탐지	797	165	182	313
패스워드 탈취 공격	454	287	87	78
유사행위 침입탐지	212	207	4	7

정확도를 평가하는 방법으로는 전체 추론된 값 중에서 정확하게 판별한 비율인 정확률, 모 집단의 모든 데이터 중 정확히 판별한 비율인 재현율, 정확률과 재현율을 합한 단위인 신뢰도를 이용한다. F-measure는 추론 결과에 대한 신뢰도를 의미하며 (수식 4)와 같이 정의한다.

$$F-measure = 2 \times \frac{\mathrm{P}recision \times Recall}{\mathrm{P}recision + Recall} \quad \text{(수식 4)}$$

(수식 4)를 이용하여 보안상황 정보 레코드에 대한 보안상황별 정확률과 재현율 그리고 신뢰도를 비교 평가한 결과는 [표 4-6]와 같다. 보안상황별 공격탐지율은 평균 70%이다.

[표 4-6] 보안상황별 공격탐지 실험 결과 2

보안상황	Precision	Recall	F-measure
백도어 공격탐지	0.34519	0.33333	0.33916
패스워드 탈취 공격	0.78630	1.73939	1.08302
유사행위 침입탐지	0.96729	18.81818	1.84000

3. 지능형 통합 접근제어 시스템의 공격 대응률

보안상황 인식서비스의 공격탐지 추론 정확률 측정과 같은 방식으로 지능형 통합 접근제어 시스템의 공격 대응률을 측정한다. [표 4-7]은 질의 종류와 사용된 추론 규칙 개수이다.

[표 4-7] 지능형 통합 접근제어 시스템의 공격 대응을 위한 질의 종류

보안상황	질의	추론규칙 개수
침입 경고 신뢰성	Active(?response), NotThreat(?threatcon), SystemContext(?syscontext), hasIntrusionType(?intrusion, ?threat), indicates(?syscontext, ?threatcon), isGeneratedBy(?intrusion, ?ids), protects(?response, ?sgres), receivedFormattedIntrusion(?IRS, ?intrusion), threatens(?threat, ?sgint), IDSconfidence(?ids, ?idsconf), contextInformationDate(?syscontext, ?sysdate), intrusionDetectionTime(?intrusion, ?intdate), neededRecommendedInference(?intrusion, ?nri), responseStatus(?intrusion, ?status), booleanNot(?nri, false), equal(?idsconf, "high"), equal(?status, "Pending"), equal(?sysdate, ?intdate), SameAs (?sgres, ?sgint) -> recommendedResponses(?intrusion, ?response), intrusionAlertReliability(?intrusion, "medium")	8
침임 탐지 최적 응답	FormattedIntrusion(?intrusion), hasarget(?intrusion, ?target), potentialOptimumResponses(?intrusion, ?respon1), potentialOptimumResponses(?intrusion, ?respon2), receivedFormattedIntrusion(?irs, ?intrusion), assetLevelOfImportance(?target, ?aloi), numberOfPotentialOptimumResponses(?intrusion, ?numOpResp), responseCost(?respon1, ?respcost1), responseCost(?respon2, ?respcost2), equal(?aloi, "low"), greaterThan(?numOpResp, 1), lessThan(?respcost1, ?respcost2) -> optimumResponse(?intrusion, ?respon1)	14
결과에 대한 권장 추론	Result(?result), hasSimilarResult(?intrusion, ?result), responseResult(?result, ?resresult), responseStatus(?intrusion, ?respStatus), equal(?respStatus, "Pending"), equal(?resresult, "Satisfactory") -> neededRecommendedInference(?intrusion, false)	6

[표 4-8]은 지능형 통합 접근제어 시스템의 공격 대응 질의의 실험 결과이다.

[표 4-8] 지능형 통합 접근제어 시스템의 공격 대응 실험 결과 1

보안상황	추론 개수	TP	FN	FP
침입 경고 신뢰성	165	76	33	56
침임 탐지 최적 응답	89	54	18	17
결과에 대한 권장 추론	65	45	12	8

지능형 통합 접근제어 시스템의 공격 대응 정확률 평가방법으로는 위의 실험과 동일하게 정확률, 재현율, 신뢰도를 이용한다.

$$F-measure = 2 \times \frac{\mathrm{P}recision \times Recall}{\mathrm{P}recision + Recall} \quad \text{(수식 4)}$$

(수식 4)를 이용하여 보안상황 정보 레코드에 대해 보안상황별로 비교 평가한 결과는 [표 4-9]와 같다. 공격 대응률은 평균 72.8%이다.

[표 4-9] 공격 대응 실험 결과 2

보안상황	Precision	Recall	F-measure
침입 경고 신뢰성	0.5758	0.6972	0.6307
침임 탐지 최적 응답	0.7606	0.7500	0.7552
결과에 대한 권장 추론	0.8491	0.7895	0.8182

Ⅴ. 결론 및 제언

본 논문은 시스템의 취약점을 수집 및 분석하고 보안 공격 대응을 위한 추론 기반 접근제어 프레임워크 설계를 목적으로 하였다. 먼저 온톨로지 기반 추론에 대해 정리하였고 기존 연구를 통해 상황인식과 접근제어에 대해 정리하였다. 보안상황 정보수집 및 추론규칙 설계 단계에서는 보안취약점을 분석하고 대응방안을 정리하였고, 정보수집 방법에 정의하였다. 그리고 보안정책 관리를 위한 지능형 접근 제어설계 단계에서는 목표를 설정하고 전체 시스템 프레임워크를 정의하고 각 모듈별 기능과 동작 시나리오를 설명하였다. 마지막으로 보안상황 온톨로지 기반 상황인식 서비스 설계 및 개발 단계를 통해 실험을 준비하였다.

실험은 보안상황 정보 레코드 200개를 이용하여 공격 이벤트를 발생시키는 모의실험으로 진행하였다. 공격 이벤트는 정상 파일이 실행되는 동안에 발생하는 행위 패턴 데이터를 기반으로 수집하였고 이를 위해 악성코드 생성기에서 생성한 500개의 악성코드와 400개의 정상 파일을 실행하여 특정 이벤트가 발생할 때마다 시스템의 행위 패턴을 수집하였다. 변칙적이고 다양한 공격방법이 발생되는 실제 상황에서 어떠한 확률로 추론할 수 있는지에 대한 객관적으로 성능 비교를 하는 데 어려움이 있었지만 실험을 통해 논문에서 제안하는 추론기반 접근제어 프레임워크를 적용하면 접근제어에 높은 정확률을 보이는 것으로 확인되었다.

향후 연구로는 시스템의 보안상황 인식서비스 기반 지능형 통합 접근 시스템을 개발하는 것을 목표로 하고 최종적으로는 시스템의 보안상황 인식서비스 기반 지능형 통합 접근 제어시스템을 구현하는 것을 목표로 한다.

나아가 일반적인 시스템의 대표적인 취약점들을 추가 조사하여 추론규칙을 재정립하여 추론식의 다양성을 갖추고 이번 논문에 설계한 지능형 접근제어 시스템과 연계하여 다양한 보안상황을 인식하고 대응하는 시스템을 개발하는 것을 목표로 한다.

참고문헌

[1] e-나라지표, 해킹사고 건수,
http://www.index.go.kr/potal/main/EachDtlPageDetail.do?idx_cd=1363

[2] 위키백과, 대한민국의 정보 보안 사고 목록,
https://ko.wikipedia.org/wiki/대한민국의_정보_보안_사고_목록

[3] 미래창조과학부, KT개인정보 유출사고 중간조사 결과 발표, 2014.3.26

[4] 방송통신위원회, 방통위, 뽐뿌 개인정보 유출 엄정 제재, 2015.11.20

[5] 방송통신위원회, 인터파크 개인정보 유출 침해사고 조사 결과,2016.8.31

[6] 방송통신위원회, 가상통화 거래사이트 <빗썸> 개인정보 유출사고, 과징금 4,350만원, 과태료 1,500만원, 시정명령 처분, 2017.12.12.

[7] 2018 신기술 용어 (한국전파기술협회)

[8] 엄정호, 박선호, 정태명. “내부자의 불법적 정보 유출 차단을 위한 접근통제 모델 설계”, 한국정보보안학회 vol.20, no.5,pp. 59-67, 2010.

[9] 네이버 지식백과, 상황 정보 (IT용어사전, 한국정보통신기술협회)

[10] 최승훈. “시맨틱 웹 기술을 이용한 확장된 특성 모델의 제한조건 검증”, 한국정보기술학회논문지 Vol.9, No.10, pp.229-236, 2011.

[11] 이완곤, 방성형, 박영택. “분산처리 환경에서 SWRL 규칙을 이용한 대용량 점증적 추론 방법”, 정보과학회논문지, Vol.44, No.4, pp.383-391, 2017.

[12] 이범기, 김미선, 서재현. “IoT에서 Capability 토큰 기반 접근제어 시스템 설계 및 구현”, 정보보호학회논문지, Vol.25, No.2, pp.439-448, 2015.

[13] 이승우, 정한민, 김평, 서동민, “추론기술연구동향” 정보통신산업진흥원 2010.

[14] 홍충성 "상황인식 프레임워크를 위한 온톨로지 기반 상황정보 모델링 방법론", 홍익대학교 박사학위논문, 2007.

[15] R. Sandu and P. Samarati, “Access Control: Principles and Practice”, IEEE Communication Magazine, pages40~48, 1994.

[16] R. S. Sandhu, “Role-Based Access Control”, In Advances in Computers, volume 46, Academic Press, 1998.

[17] Muhammad Nabeel Tahir, “C-RBAC: Contextual Role-Based Access Control Model”, UBICC Journal, Volume 2 No. 3, 2007.

[18] 이승철, 김치수, 임재현, “의도추론의 모허성 해결을 위한 온톨로지 기반 상황 해석 구조의 설계 및 구현”, 한국컴퓨터종합학술대회 논문집, 제 34권, 제 1호, pp. 208-213, 2007