8-2021

Multilevel Security Policy Implementation Using OWL Ontology

Ruting Bai

MULTILEVEL SECURITY POLICY IMPLEMENTATION USING OWL ONTOLOGY

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Information Systems and Technology: Cybersecurity

---

by

Ruting Bai

August 2021

MULTILEVEL SECURITY POLICY IMPLEMENTATION USING
OWL ONTOLOGY

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Ruting Bai

August 2021

Approved by:

Joon Son, Ph.D, Committee Chair, Information & Decision Science

Conrad Shayo, Ph.D, Committee Member, Information & Decision Science

Jay Varzandeh, Ph.D, Dept. Chair, Information & Decision Science

ABSTRACT

This project is an experimental implementation of Multi-Level Security (MLS) lattice model by using semantic web technologies (OWL) to create and test Mandatory Access Control (MAC) with Bell-LaPadula (BLP) properties. Semantic web (web of data) is building on top of the World Wide Web (web of documents), aiming to make data machine-readable so that to improve data processing and management. OWL is a semantic web computational logic-base language which is designed to represent complex knowledge in semantic format. With the MLS ontology, we are able to define dominance relationship between variables within the lattice model and perform different queries to verify if the subject (with security clearance) can access (read/write) to the object (with security classification). Moreover, by leveraging BLP properties, the ontology would only allow information to flow from entities with lower classification to entities with higher classification.

ACKNOWLEDGEMENTS

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

I would like to thank my thesis advisor Dr. Joon Son of the JHB College of Business and Public Administration at California State University, San Bernardino. The door to Prof. Son's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work. Without his support, I would not be able to accomplish this project.

I would also like to acknowledge Dr. Conard Shayo of the JHB College of Business and Public Administration at California State University, San Bernardino as the second reader of this project, and I am gratefully indebted to him for his very valuable comments on this project.

Finally, I must express my appreciation to my parents' unfailing support and continuous encouragement throughout my years of. Without their support, I would not accomplish this project and finish my degree.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES viii

LIST OF FIGURES viii

CHAPTER ONE: INTRODUCTION 1

Research Motivation 1

Organization 3

CHAPTER TWO: BACKGROUND 4

Mandatory Access Control 4

Dominance Rule 5

BLP Security Policy (Bell, 2005) 5

Simple Security Policy 6

* (Star) Property 7

Multi-Level Security 8

Intro to the Semantic Web and Technologies 12

RDF 12

OWL 13

Semantic Rule Language (SWRL) 13

Protégé 15

CHAPTER THREE: MODELING MULTI-LEVEL SECURITY IN OWL 16

Building MLS Ontology 16

Step 1. Create Classes 16

Step 2. Create Object Properties and Inverse Properties 18

Step 3. Modeling Classes Expression with Property Restrictions . 22

Step 4. Create Individuals with Property Assertions...................... 25

CHAPTER FOUR: SWRL RULE IMPLEMENTATION FOR MAC AND BLP...... 28

Apply Dominance Rule 28

Testing MLS Ontology with Mandatory Access Control Criteria............... 30

Test Scenario 1 (Comparable Security Labels)............................. 30

Test Scenario 2 (Incomparable Security Labels) 33

SWRL Rules for BLP Implementation within a Single Domain................. 34

Simple Security Property... 35

* (Star) Property 38

CHAPTER FIVE: CONCLUSION.. 41

Future Work 41

REFERENCES 42

LIST OF TABLES

Table 1. Create Classes and Subclasses 17

Table 2. Convert the Knowledge into RDF Triple and Property 18

Table 3. List of Property Domain and Range 21

Table 4. Property and Inverse Property 22

Table 5. Property Restrictions of Each Class 24

Table 6. Individuals with Property Assertion 26

Table 7. Subject and Object Individuals with Assertions 35

LIST OF FIGURES

Figure 2.1. Left side: without BLP properties information can flow from high to low. The simple security condition would prevent low from reading high. The star property would prevent high from writing to low. Right side: with BLP properties information can only flow from low to high........................ 6

Figure 2.2. Lattice structure.. 7

Figure 2.3. Lattice Model 9

Figure 2.4. Information Flow with BLP.. 11

Figure 2.5. The Layers of Semantic Web Technology...................................... 12

Figure 3.1. Apply Transitive Characteristic to isGreaterThan 20

Figure 3.2. Security Label TS_BioNuke.. 23

Figure 3.3. Compartment BioNuke 24

Figure 3.4. Protégé Inconsistent Ontology Explanation..................................... 27

Figure 4.1. List of All Comparable Security Label Pairs..................................... 31

Figure 4.2. List of Comparable Security Labels of _SecurityLabel_TS_Bio........ 32

Figure 4.3. List of Incomparable Security Labels.. 34

Figure 4.4 Query Result of SQ4 37

Figure 4.5. Query Result for SQ5 39

CHAPTER ONE

INTRODUCTION

Research Motivation

Web development has never stopped since the birth of the Internet in 1962. To look back from these days, it requires users to have expert knowledge for accessing information through the Internet. In the 1990s, the founder of the World Wide Web, Sir Tim Berners-Lee, invented the World Wide Web and wrote the three fundamental technologies of the web, HTML, URI and HTTP. In addition, with the invention of search engines to form today's digital world that enables normal people to access the information on the web without any expert knowledge. In the past 20 years, the rapid growth of web technologies upgraded the web to a data centered processing age, in which users become the mainstream in data generation through broadcasting and social networking. Berners-Lee, Hendler and Lassila (2001) first discussed their vision of the web in the future. They discussed that the current web is the foundation of semantic web. It's goal is to apply semantic meaning to the web to make data machine-readable and develop new technologies to better store, process and express knowledge with large volume of data.

Some parts of the vision have already come true. Semantic web technologies have been used in the healthcare industry and artificial intelligence for knowledge modeling. Meanwhile, information security is always a critical topic. Throughout the years, cybersecurity professionals are aware of the challenges brought by new web technologies such as cloud computing, big data, Internet of Things, etc. The security threats are not only coming from the Internet, but also from the internal environment. Case studies such as Marriott Data Breach (Sanger et al., 2018) and US Office of Personnel Management (Thomas, 2019) proved that design and maintaining the security of information systems is the priority for both private and government agencies. Organizations have the obligation to collect, process, store and share sensitive data in a secure manner. For example, health care information of patients, top secret military resources and personal identity information should all be protected because data breach can cause huge financial loss to individuals and organizations as well as increase national security issues. Multi-level security policy (MLS) is prevalent in military systems, and further enforced on their contractors and partners. The increasing security threats from both internal and external environments also lead a lot of organizations to embrace to the MLS in order to raise their security profile. Each uses access control to require pre-authorized user privileges to gain access to the designated information according to the classification of the data.

While the web is extending in a semantic manner, some questions came to mind. Security measures should be implemented in every layer of the web environment. When the data are formalized with semantic meaning, what kind of security measures can be used to protect the data in a semantic environment? Even though no study shows a semantic version of MLS implementation, if it is possible to implement the MLS policy in this environment? Hence, I think there are emerging needs to upgrade the access control policies while adopting new web technologies within the organization. Therefore, the security policies should also make an extension to enforce information security management in the semantic web environment.

Organization

The remainder of the paper is organized as follows. Chapter 2 summarizes the past studies on MLS and provides a brief introduction of the semantic web. Chapter 3 demonstrates how the MLS lattice model is constructed by using Protégé, and Chapter 4 discusses how to use semantic web rule language to apply dominance rules in the ontology. In conclusion, Chapter 5 summarizes the work accomplished in this project and discusses areas for future development.

CHAPTER TWO

BACKGROUND

Mandatory Access Control

Defined by the National Institute of Standards and Technology(NIST), the Mandatory access control (MAC) is a type of nondiscretionary access control that enforces a uniform security level to all subjects and objects in an information system. (“Mandatory Access Control”, n.d.) To prevent the information flow from a subject must be authorized (with security clearance) to access an object (with security classification). Past research shows that MAC is closely related to Multi-Level Security (MLS). MLS is first proposed by the defense community to maximize the protection of sensitive and confidential information. (43.6. Multi-Level Security(MLS), n.d.) It is widely used in the defense industry, especially in the military system and government with higher levels of security than those in private business and organizations. In addition, MLS uses the Bell-LaPadula (BLP) model to prevent confidential information flow from higher level to lower level with the need-to-know requirement. (Kim, 2020) According to Bell (2005), Denning (1976) introduced a lattice structure, Bell-LaPadular (BLP) model, to compare the security levels of user clearance and information classification.

Within a large and complex information system, sensitivity level it is not flexible enough to classify the information sensitivity and user clearance. The BLP model uses additional information known as a compartment (also called category or *need to know*) to specify MLS security labels or levels. An MLS security level or label is a sensitivity level or a pair of a sensitivity level and a set of compartments. In this project, we use a colon to separate a sensitivity level and a set of compartments when defining a security level or label in concept. (Elliott,1990; van Tilborg, Jajodia, 2011) A few examples of security levels are *TopSecret:{bio,chem}*, *Secret:{}*, and *Unclassified:{nuke,bio}*.

<u>Dominance Rule</u>

An MLS system has a dominance rule that defines a partial order (≤) over the MLS security levels. The partial ordering (≤) is always defined such that two securitylevels can be compared for dominance:

Given two security levels l_1 with sensitivity level S_1 and compartment C_1, and l_2 with sensitivity level S_2 and compartment C_2. We write $l_1 \leq l_2$, meaning l_1 is dominated by (is less than) l_2 or l_2 dominates (is greater than) l_1 when

- S_2 is equal to or higher than S_1
- C_1 is a subset of C_2, namely, $C_1 \subseteq C_2$

BLP Security Policy (Bell, 2005)

The BLP security policies enforce that every subject and object must have at least one security label. To block information flow from entities with higher sensitivity level to ones with lower sensitivity level within the information system, two important properties are proposed: simple security property and star property

(Figure 2.1).

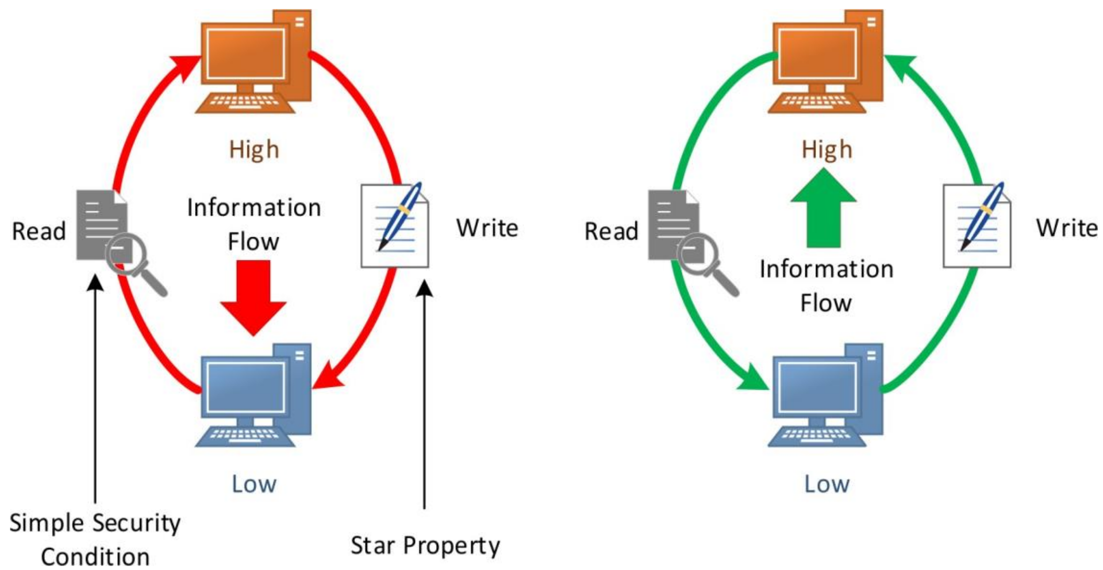


Figure 2.1. Left side: without BLP properties information can flow from high to low. The simple security condition would prevent low from reading high. The star property would prevent high from writing to low. Right side: with BLP properties information can only flow from low to high.

Simple Security Policy

Also known as the "no read-up" policy of the BLP model states that a subject with certain security clearance cannot read an object with a higher classification. Therefore, given the subject's security label *sl(S)* and the object's security label *sl(O)*, the subject can read the object when

$$sl(O) \leq sl(S)$$

Example 1. *Assuming Alice is granted a security clearance TS:{bio}, namely, sl(Alice) =TS:{bio} and the object O1 has the security classification*

TS:{bio, chem}, namely, sl(O1) = TS:{bio, chem}. {bio} is a subset of {bio, chem}. Then, Alice cannot read O1 as sl(Alice) ≤ sl(O1).

* (Star) Property

Also known as the "no write-down" policy states that a subject with certain security clearance cannot write to any object with a lower security classification. Therefore, given the subject's security label *sl(S)* and the object's security label *sl(O)*, the subject can write the object when

$$sl(S) \leq sl(O)$$

Example 2. *Referring the same scenario in* Example 1*, sl(Alice) = TS:{bio} and sl(O1) = TS:{bio, chem}. Then Alice can write to O1 as sl(Alice) ≤ sl(O1).*

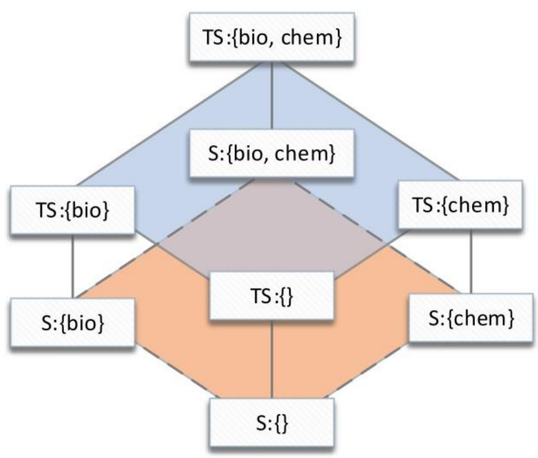


Figure 2.2. Lattice structure (Kim, 2020)

Example 3. *The diagram in Figure 2.2 depicts the partial ordering (≤) over the MLS security levels as a lattice. Assuming Bob is granted a security* *clearance TS:{}, namely, sl(Bob) = TS:{} and Frank is granted a security clearance S:{}, namely, sl(Frank) = S:{}. Two objects, O2 is classified as TS:{}, namely, sl(O2) = TS:{}, and O3 is classified as S:{}, namely, sl(O3) = S:{}. Compare the security labels between the subjects and the objects. Between Bob and O2, sl(Bob) = TS:{} = sl(O2), Bob can read and write O2. Similarly, since sl(Frank) = S:{} = sl(O3), Frank can read and write to O3. As sl(Bob) = TS:{} is higher than sl(O3) = S:{}, Bob can only read O3. Bob will be blocked from writing to O3 because information cannot flow from high to low. As S:{} ≤ TS:{}, Frank can write to O2 but not read O2.*

Example 4. *Attaching compartments to sensitivity level gives more flexibility to information classification in a complex information system. Figure 2.2 shows that there is no partial ordering between TS:{} and S:{bio} (i.e., they are not comparable). This means that no operation such as read or write should be performed between them.*

Multi-Level Security

The lattice structure of MLS with BLP model (Figure 2.3) is formed with vertices connected by edges. The model distinguished two sets of vertices with different colors by their hierarchy levels. Each security label $(SL(s_i, c_i))$ has two components, sensitivity level S_i and compartment C_i. Sensitivity level is hierarchically defined with a range from high to low, “Top Secret” ≥ “Secret” ≥ “Classified” ≥ “Unclassified”. Compartment is defined as {Bio, Nuke} ⊇ {Bio} |

{Nuke} ⊇ {}. Vertices in red area are labels with "Top Secret" clearance (noted as TS) and vertices in orange labels with "Secret" clearance (noted as S). (Kim, 2020)

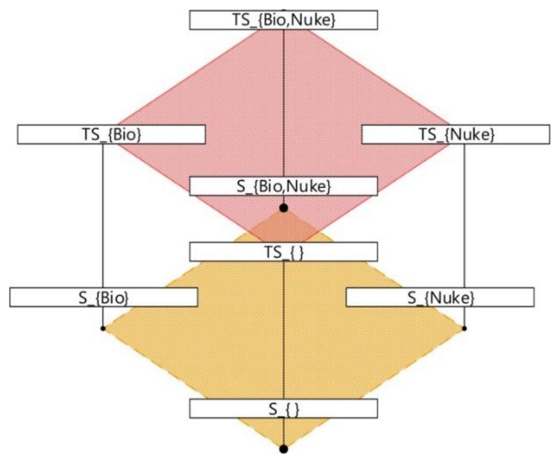


Figure 2.3. Lattice Model (Kim, 2020)

<u>Example 5.</u> *Based on Figure 2.3, "Top Secret" TS_{} is considered a higher classification than "Secret" S_{}. TS_{} can read S_{} because information is allowed to flow from a lower classification ("Secret") to a higher classification ("Top Secret"). Inversely, it prohibits S_{} read up to TS_{} to prevent information leaking from higher classification to lower classification. Meanwhile, S_{} can write up to TS_{} but TS_{} cannot write down to S_{}.*

Moreover, the BLP model does not grant users with "Top Secret" clearance to access all objects. With additional need-to-know restriction, known as compartment (Example 6), to block irrelevant users from accessing confidential information. (Denning, 1976; Panossian, 2019)

Example 6. *Based on Figure 2.3, assuming Mary with security clearance TS_{} is trying to read/write the object file with security classification S_{Nuke}. Mary passes the first criteria because she has a "Top Secret" clearance which is higher than the object file classification. However, she also needs a compartment {Nuke} to meet the second criteria. {} can not grant her access to objects with {Nuke}. This example explains how the need-to-know condition is applied to provide an extra layer of protection to the information system.*

In this project, the mathematical notation used to define a security label such as SL(Si,Cj) is also expressed in terms of SL(TS_{Bio,Chem}) or SL(TS, {Bio,Chem}). To examine if there is a dominance relationship between two security label variables, both dominance rules must be satisfied. Once the dominance relationship exists, the two BLP properties can be easily applied to complete the MLS policies based on this relationship.

In addition, the lattice structure specifies the path of information flow according to the dominance relationship between the vertices through the edges. (Panossian, 2019) To block information leaking from higher classification to lower classification (Figure 2.4), MLS enforces simple security property and star property. Example 7 and Example 8 each will discuss the scenarios how each BLP property ensures the information flow from lower classification to higher classification. These examples will illustrate the rules to identify if a subject (S) can read/write an object (O) based on their security labels.

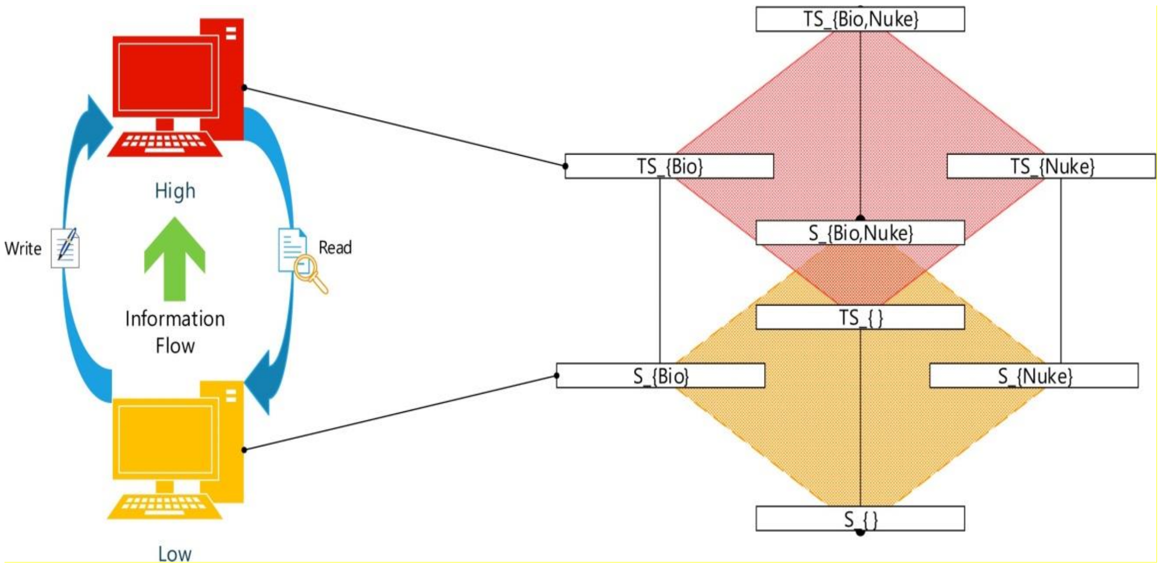


Figure 2.4. Information Flow with BLP (Kim, 2020)

Example 7. *Assuming a person A (s_i) has the security clearance S_{Bio} and an object (o_i) with the classification TS_{Bio, Nuke}, s cannot read o because SL(s_i) ≤ SL(o_i). However, s_i can read any object when SL(s_i) ≥ SL(o_j). For instance, SL(o_j) equal to S_{Bio} and SL(o_k) equal to S_{}. (Kim, 2020)*

Example 8. *Assuming every variable has the same security label as shown in Example 7,person (s_i) can now write to o_i and o_j because SL(s_i) ≤ SL(o_j), which allow information to flow from lower level security clearance to higher level security clearance. However, person A will not be able to write to o_j as well as o_k.(Kim, 2020)*

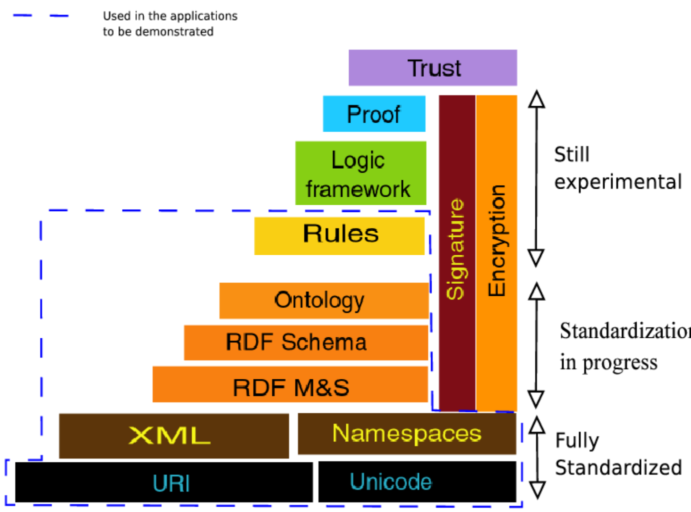


Figure 2.5. The Layers of Semantic Web Technology

Semantic Web is an extension of the current world wide web standardized by the W3C. Its goal is to make the implicit meaning of data to be explicitly represented, so that the data is machine-readable to improve information retrieval and produce more useful work. Some of the semantic web technologies (Figure 2.5), RDF, OWL, SWRL and Protégé, are used in this project and each will be given a brief introduction.

<u>RDF</u>

Resource Description Framework (RDF) is a fundamental block of the semantic web built on top of HTML, HTTP, and XML to express the semantic meaning of knowledge. The resource can be anything and must be uniquely identified and referenced via Internalized Resource Identifier (IRI). Knowledge is expressed in a list of statements called triple, which follows a simple schema with three components, subject, property and object. In RDF, the subject and the property must be IRI, and the object of the triple can be either an IRI or a literal (datatype).

OWL

The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge based on description logics to describe classes, individuals and properties. It transfers the common knowledge of philosophy and mathematics into a formal language in the form of RDF to give semantic meaning, so that the knowledge becomes machine understandable. The goal of building an OWL ontology is to create a model that represents a subject of matter with individual things, kinds of things, and kinds of relationships, as well as support automated reasoning. A class represents things of an interest group, an individual is an instance of a class, and a property defines the relationship between subjects and objects. Description logic separates terminological knowledge base to assertional knowledge base. Terminological knowledge base describes the relationships between classes when defining the model and assertional knowledge describes how individuals are related to each other.

Semantic Rule Language (SWRL)

SWRL combines OWL ontology and DataLog expressions that apply DataLog rules to OWL ontologies in the form of “If…then…” statements. SWRL rules are in the form of “Antecedent -> Consequent”. The term “Antecedent” is also referred to rule body and “Consequent” is referred to rule head. (O’Connor et al., 2005) The body represents the “If…” statement and the head represents the “then…” statement. An example SWRL rule can be:

SecurityLabel(?a) ^ SecurityLabel(?b) ^ sameAs(?a,?b) -> read(?a,?b)

This example explains the rule states that “If two security label a is equal to security label b, then a can read b.” For the implementation of BLP in chapter 4, such rules will be created to apply the read/write relationship between subjects and objects. Each will be discussed and shown output of implementation.

Without SWRL, the ontology can still be implemented by manually created assertions in the editor. However, if an ontology has hundreds of assertions for a small ontology to made to represent the knowledge without using an inference engine, it is very inefficient for manually processing data. SWRL provides automated reasoning functions. The inference engine can finish the work of creating inference assertions in milliseconds. Moreover, modification of an individual can cause modification of several assertions. SWRL can carry the rest of the modification to improve work efficiency. Several studies have shown that using SWRL can improve business process management. According to Abadi, Ben-Azza, Sekkat (2018), SWRL is the only tool which gathers the ontology to model the information and model decision making rules for industrial applications. Matsokis and Kiristsis also suggested using SWRL to extend the OWL models to develop a learnable approach in production management. (2011)

Furthermore, Roy, Dayan and Holla presented that it supports business knowledge management in industrial business processes. (2018)

Protégé

Protégé is an open-source ontology editor developed by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. This tool is widely used by academic, government, and corporate groups. It complies with W3C standards, has visualization support and extensive build-in tools to support ontology construction. According to Rubin et al.(2005),Protégé provides a variety of features to support developers in creating, modifying and managing ontologies:

- Simple and customizable user interface
- Support collaboration work
- Visual support for ontology expressions
- Built-in reasoners for checking consistency and inference engine
- Multiple formats for exporting ontology to other platforms
- Web version compatible to desktop version

CHAPTER THREE

MODELING MULTI-LEVEL SECURITY IN OWL

This chapter will demonstrate the steps of building MLS ontology in Protégé.

The three key components of OWL ontology are classes, properties and individuals. To distinguish each component, this project uses the following naming conventions without spaces:

1. Classes: upper camel cases (e.g., Person, Animal, Food)
2. Properties: lower camel cases (e.g., isGreaterThan, hasPet, movesTo)
3. Individuals: leading underscore (e.g., _JohnSmith, _Dog, _Pizza)

Building MLS Ontology

Step 1. Create Classes

The implementation starts with defining the terminological knowledge. Previously, Chapter two discussed that a security label has two components, sensitivity level and compartment. The first step is to create three classes, *SecurityLabel*, *SensitivityLevel*, *Compartment* and their subclasses. Refer to the lattice structure in Figure 2.2, each node will be a subclass of *SecurityLabel*. A security label has two components, sensitivity level and compartment. *TopSecret* and *Secret* are subclasses of SensitivityLevel; and *BioNuke*, *Bio*, *Nuke*, *Null(represents { })* are subclasses of Compartment. Because OWL uses open world reasoning, it means if two classes are not specified to be different types of things, they are unknown to be different and allow to have intersections. To say that there are no common members in *SecurityLabel*, *SensitivityLevel* and *Compartment*, these three classes are disjoint to each other. It means that one individual cannot be an instance of more than one of the three. Protégé allows users to create a list of classes and indicates disjointness by using the *Create Class Hierarchy* tool. To verify the implementation, select a random class to view in the bottom of the Class Description. All sibling classes of the selected class should be shown in the *Disjoint With* section.

In addition, at the same class hierarchy level as *SecurityLabel*, *SensitivityLevel* and *Compartment*, two more disjoint classes, *Subject* and *Object* are created for implementation in the next chapter. Table 1 shows the full list of classes with class hierarchy levels.

Table 1. Create Classes and Subclasses

Class	Subclass
Compartment	Bio
	BioNuke
	Nuke
	Null
SensitivityLevel	TopSecret
	Secret
	Confidential
	Unclassfied
SecurityLabel	TS_BioNuke
	TS_Bio
	TS_Nuke
	TS_Null
	S_BioNuke
	S_Bio

Class	Subclass
	S_Nuke
	S_Null
Subject	
Object	

Step 2. Create Object Properties and Inverse Properties

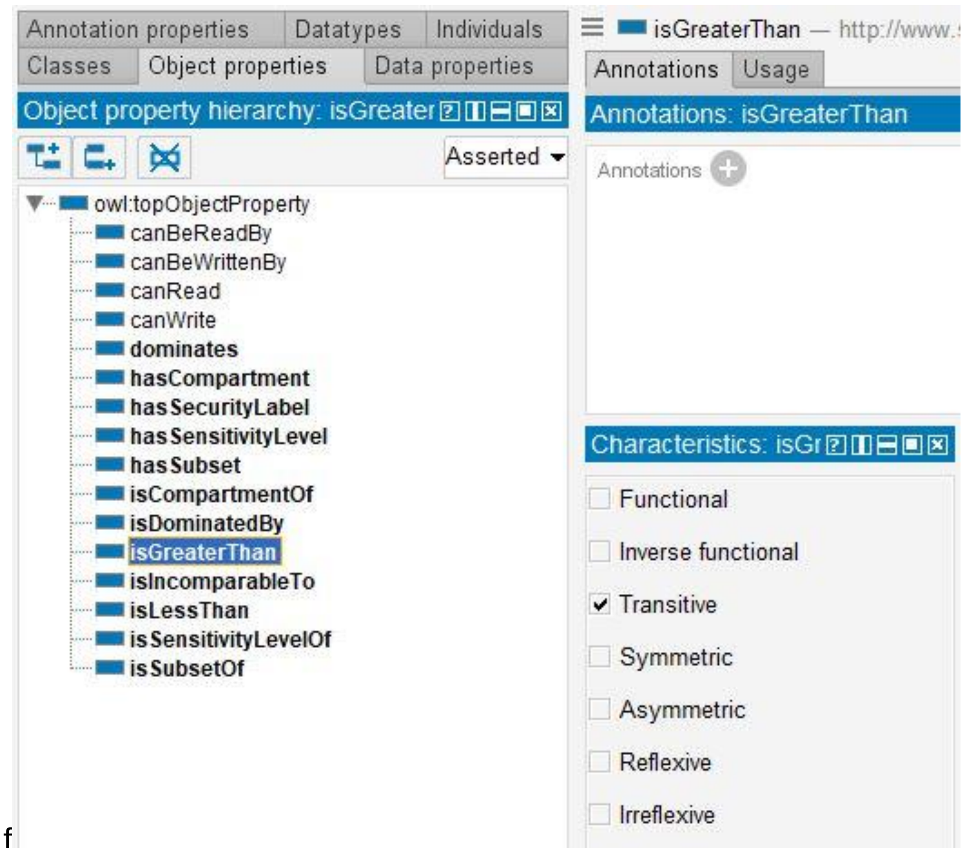
The second step is to define the binary relationships (properties) between entities. Table 2 shows how common knowledge is converted into RDF triple and property for MLS ontology:

Table 2. Convert the Knowledge into RDF Triple and Property

Knowledge	RDF Triple	Property
A security label consists of one sensitivity level.	*SecurityLabel hasSensitivityLevel SensitivityLevel.*	*hasSensitivityLevel*
A security label consists of one compartment	*SecurityLabel hasCompartment Compartment.*	*hasCompartment*
The compartment BioNuke has subset Bio or Nuke.	*BioNuke hasSubset (Bio or Nuke)*	*hasSubset*
The (sensitivity level) Top Secret is greater than (sensitivity level) Secret.	*TopSecret isGreaterThan Secret*	*isGreaterThan*
A Subject has one security label.	*Subject hasSecurityLabel SecurityLabel.*	*hasSecurityLabel*
Security label TS_{Bio} dominates security label S_{Bio}.	*TS_Bio dominates S_Bio.*	*Dominates*
Security label TS_{Bio} cannot compare to security label S_{Nuke}.	*TS_Bio isIncomparableTo S_Nuke.*	*isIncomparableTo*
A Subject can read an Object	*Subject canRead Object.*	*canRead*
A Subject can write to an Object	*Subject canWrite Object*	*CanWrite*

There are two types of RDF property. The first type is object property which links individuals to individuals, and the second type is datatype property which links individuals to RDF datatypes (e.g. string, integer, date, etc.). In this MLS ontology, all properties are object properties.

Properties have characteristics. In Protégé(Figure 3.1), it is very easy to specify the characteristics of the property. The transitive characteristic will be specified in three properties, *hasSubset*, *isGreaterThan* and *dominates*. These properties have the characteristics that if X is related to Y and Y is related to Z, then X is related to Z. It is not necessary to add an assertion to state that X is related to Z. The inference engine can generate the inferred axioms if the property characteristics are specified.



f

Figure 3.1. Apply Transitive Characteristic to isGreaterThan

Protégé also gives the option to define the domain and the range of properties with the same meaning in mathematics. Given two individuals are connected by a property in an RDF triple. The domain class specified that the subject of the triple belongs to the domain class as well as the object of the triple belongs to the range class.

Table 3 lists the domain and range is listed for each property. Take the *hasSensitivityLevel* as an example, the domain of this property is *SecurityLabel*, and the range is *SensitivityLevel*. Whenever a triple assertion contains *hasSensitivityLevel,* the subject of this triple should be an instance of *SecurityLabel*, and the object should be an instance of *SensitivityLevel*.

With the specification of property domain and range as well as class disjointness, the built-in Protégé reasoner Pallet can catch inconsistent assertions which conflict with the description logic expressed in the model. The reasoner can catch inconsistent assertions such as *A* (instance of SecurityLabel) *hasSensitivityLevel B* (instance of Compartment), or *A* (instance of Compartment) *hasSensitivityLevel B* (instance of SensitivityLevel).

Table 3. List of Property Domain and Range

Object Property	Domain	Range
hasSensitivityLevel	*SecurityLabel*	*SensitivityLevel*
hasCompartment	*SecurityLabel*	*Compartment*
hasSubset	*Compartment*	*Compartment*
isGreaterThan	*SensitivityLevel*	*SensitivityLevel*
dominates	*SecurityLabel*	*SecurityLabel*
isIncomparableTo	*SecurityLabel*	*SecurityLabel*
canRead	*Subject*	*Object*
canWrite	*Subject*	*Object*

Each object property can have its inverse property. In an RDF triple, the property links the subject to the object in one direction. Its inverse property applies this relationship from an opposite perspective. For example, if *A* is linked to *B* through property *P*, the inverse way of saying the same thing is that B is linked to A through inverse property P_i. In Protégé, the inverse relationship between P and P_i can be defined in the Property Description panel. To better support the rule inferences in the next chapter, an inverse property is created for each object property (Table 4).

Table 4. Property and Inverse Property

Property (P)	Inverse Property (P_I)
hasSensitivityLevel	*isSensitivityLevelOf*
hasCompartment	*isCompartmentOf*
hasSubset	*isSubsetOf*
isGreaterThan	*isLessThan*
dominates	*isDominateBy*
isIncomparableTo	*N/A*
canRead	*canBeReadBy*
canWrite	*canBeWrittenBy*

Step 3. Modeling Classes Expression with Property Restrictions

The third step is to apply property restrictions to model class expression. Properties describe the relationship between individuals. It can also be used as a special kind of class description to emphasize that all instances of the class must satisfy the restriction. There are four types of property restrictions, existential, universal, cardinality and value restrictions. To model the SecurityLabel class, existential and universal restrictions will be used to define SecurityLabel and its subclasses. Take *TS_BioNuke* (Figure 3.2) as example, the class must qualify for two conditions:

1. The class must have a sensitivity label and the security label must be TopSecret. (existential & universal)

2. The class must have a compartment and the compartment must be BioNuke. (existential & universal)

According to the two conditions, four new property restrictions are applied:

1. hasSensitivityLevel some TopSecret
2. hasSensitivityLevel only TopSecret
3. hasCompartment some BioNuke
4. hasCompartment only BioNuke

User can click the compartment of *TS_BioNuke, BioNuke,* Protégé will redirect to class description of this class(Figure 3.3).

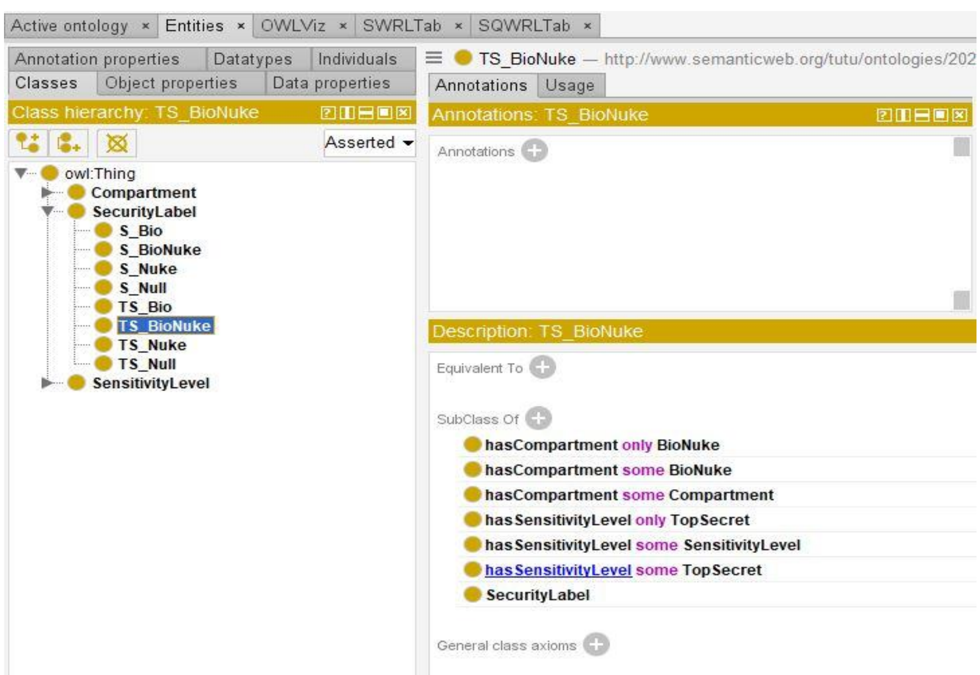


Figure 3.2. Security Label TS_BioNuke

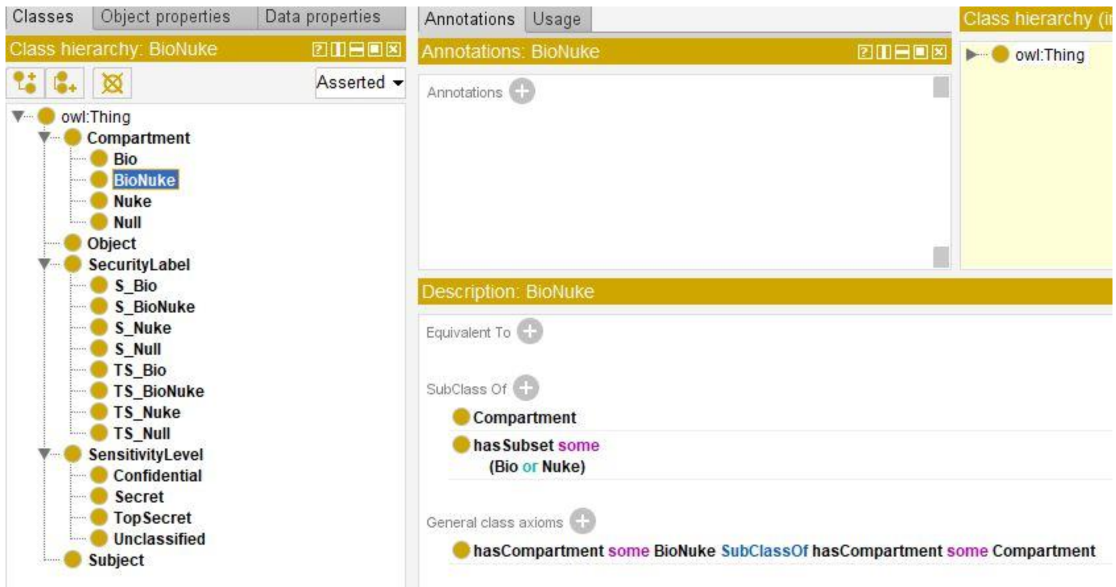


Figure 3.3. Compartment BioNuke

Moreover, the bellowing table is a list of the property restrictions applied to each class.

Table 5. Property Restrictions of Each Class

Class	Subclass	Property Restrictions
Compartment	BioNuke	hasSubset some (Bio or Nuke)
	Bio	hasSubset some Null
	Nuke	hasSubset some Null
SensitivityLevel	*TopSecret*	*isGreaterThan some Secret*
	Secret	*isGreaterThan some Confidential*
	Confidential	*isGreaterThan some Unclassified*
SecurityLabel		*hasSensitivityLevel some SensitivityLevel* *hasCompartment some Compartment*
	TS_BioNuke	*hasSensitivityLevel some TopSecret* *hasSensitivityLevel only TopSecret* *hasCompartment some BioNuke* *hasCompartment only BioNuke*
	TS_Bio	*hasSensitivityLevel some TopSecret* *hasSensitivityLevel only TopSecret*

Class	Subclass	Property Restrictions
		hasCompartment some Bio *hasCompartment only Bio*
	TS_Nuke	*hasSensitivityLevel some TopSecret* *hasSensitivityLevel only TopSecret* *hasCompartment some Nuke* *hasCompartment only Nuke*
	TS_Null	*hasSensitivityLevel some TopSecret* *hasSensitivityLevel only TopSecret* *hasCompartment some Null* *hasCompartment only Null*
	S_BioNuke	*hasSensitivityLevel some Secret* *hasSensitivityLevel only Secret* *hasCompartment some BioNuke* *hasCompartment only BioNuke*
	S_Bio	*hasSensitivityLevel some Secret* *hasSensitivityLevel only Secret* *hasCompartment some Bio* *hasCompartment only Bio*
	S_Nuke	*hasSensitivityLevel some Secret* *hasSensitivityLevel only Secret* *hasCompartment some Nuke* *hasCompartment only Nuke*
	S_Null	*hasSensitivityLevel some Secret* *hasSensitivityLevel only Secret* *hasCompartment some Null* *hasCompartment only Null*
Subject		*hasSecurityLabel some SecurityLabel*
Object		*hasSecurityLabel some SecurityLabel*

Step 4. Create Individuals with Property Assertions

After modeling classes with property restrictions, we can then create instances with property assertions. Table 6 shows a list of individuals with their property assertions for each security label node of the lattice model.

Table 6. Individuals with Property Assertion

Class	Individual	Property Assertions
BioNuke	_Compartment_BioNuke	hasSubset _Compartment_Bio hasSubset _Compartment_Nuke
Bio	_Compartment_Bio	hasSubset _Compartment_Null
Nuke	_Compartment_Nuke	hasSubset _Compartment_Null
Null	_Compartment_Null	
TopSecret	_SensitivityLevel_TopSecret	isGreaterThan _SensitivityLevel_Secret
Secret	_SensitivityLevel_Secret	
TS_BioNuke	_SecurityLabel_TS_BioNuke	hasSensitivityLevel _SensitivityLevel_TopSecret hasCompartment _Compartment_BioNuke
TS_Bio	_SecurityLabel_TS_Bio	hasSensitivityLevel _SensitivityLevel_TopSecret hasCompartment _Compartment_Bio
TS_Nuke	_SecurityLabel_TS_Nuke	hasSensitivityLevel _SensitivityLevel_TopSecret hasCompartment _Compartment_Nuke
TS_Null	_SecurityLabel_TS_Null	hasSensitivityLevel _SensitivityLevel_TopSecret hasCompartment _Compartment_Null
S_BioNuke	_SecurityLabel_S_BioNuke	hasSensitivityLevel _SensitivityLevel_Secret hasCompartment _Compartment_BioNuke
S_Bio	_SecurityLabel_S_Bio	hasSensitivityLevel _SensitivityLevel_Secret hasCompartment _Compartment_Bio
S_Nuke	_SecurityLabel_S_Nuke	hasSensitivityLevel _SensitivityLevel_Secret hasCompartment _Compartment_Nuke
S_Null	_SecurityLabel_S_Null	hasSensitivityLevel _SensitivityLevel_Secret hasCompartment _Compartment_Null

Till this step, the security label modeling has completed. The ontology modeling constructs terminology assertions are applied to classes with property restrictions. Assertional knowledge is represented with individuals. For testing purposes, select *Compartment* individual *_Compartment_Bio* and add an object property assertion to represent *_Compartment_Bio* isGreaterThan *_Compartment_Null*. Running Pellet reasoner, an *inconsistentOntologyException* error message popped up because Protégé explains (Figure 3.2) that the domain and range of *isGreaterThan* are limited to *SensitivityLevel*, which is disjoint to *Compartment*. The test assertion conflicts with the specified domain and range classes of *isGreaterThan*. This test shows the reasoner's capability of catching inconsistency errors. Reasoner can be used to detect the modeling errors at any step.

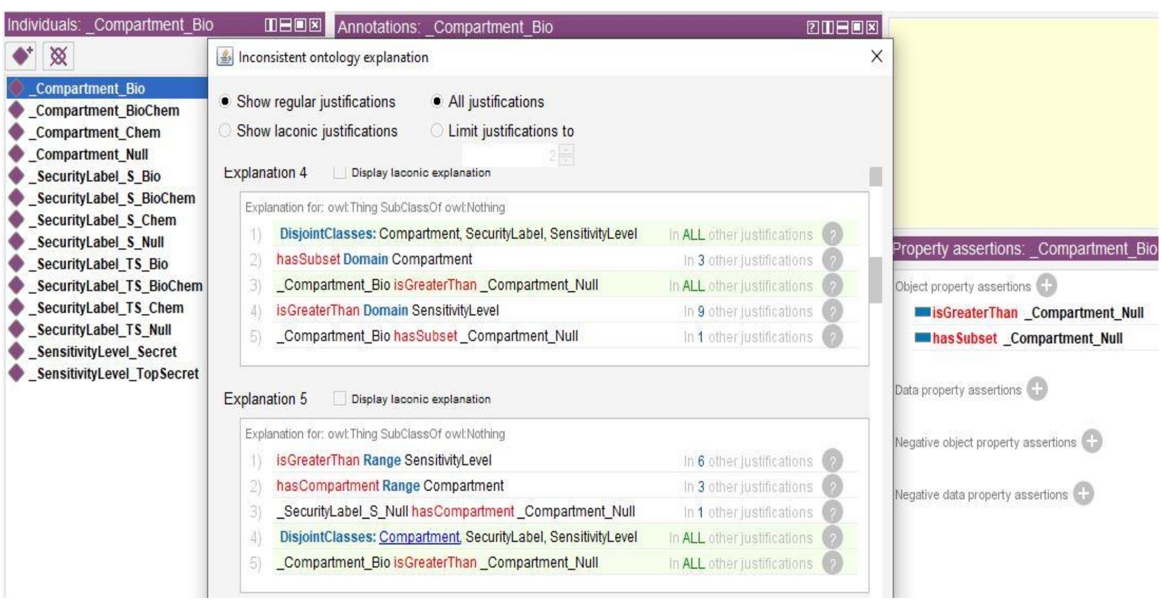


Figure 3.4. Protégé Inconsistent Ontology Explanation

CHAPTER FOUR

SWRL RULE IMPLEMENTATION FOR MAC AND BLP

Apply Dominance Rule

This section uses a Semantic Web Rule Language (SWRL) to apply dominance rules to the MLS ontology. SWRL combines OWL and DataLog expressions in the form of Horn-like rules to express "*If …, then …*" statements. The SWRL inference engine checks the set of predefined rules to apply the relationship to the matching variables. Therefore, any modification of the ontology will automatically update the inferred axioms by SWRL. The purpose of using SWRL is not only to use it as an inference engine, but also SWRL can transfer the inferred axioms to the OWL model to make them explicitly represented. The ontology (with inferred axioms made by inference engine) can be exported to be reviewed in simple text editor or other semantic tools.

For a pair of security labels, the *dominates* relationship is not directly asserted. Refer to the dominance rule discussed in Chapter 2, two security labels can be compared for dominance:

An MLS system has a dominance rule that defines a partial order (≤) over the MLS security levels. The partial ordering (≤) is always defined such that two securitylevels can be compared for dominance:

Given two security levels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, we write $L_1 \leq L_2$, meaning L_1 is dominated by (less than) L_2 or L_2 dominates (is greater than) L_1

when

- S_2 is a higher sensitivity level than S
- C_1 is a subset of C_2, namely, $C_1 \subseteq C_2$

Property *dominates* and its inverse property *isDominatedBy* are used to represent the dominance relationship between the security labels. Convert the mathematical notation into SWRL, the following rules are created:

Rule 1:

S1 - Compare two security labels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, if $S_1 = S_2$, C_2 has subset C_1, then L_2 dominates L_1.

```
SecurityLabel(?L1) ^ hasSensitivityLevel(?L1,?S1) ^ hasCompartment(?L1,?C1) ^
SecurityLabel(?L2) ^ hasSensitivityLevel(?L2,?S2) ^ hasCompartment(?L2,?C2) ^
sameAs(?S1,?S2) ^ hasSubset(?C1,?C2) -> dominates(?L1,?L2)
```

Rule 2:

S2- Compare two security labels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, if S_2 is greater than S_1, C_2 has subset C_1, then L_2 dominates L_1.

```
SecurityLabel(?L1) ^ hasSensitivityLevel(?L1,?S1) ^ hasCompartment(?L1,?C1)  ^
SecurityLabel(?L2) ^ hasSensitivityLevel(?L2,?S2) ^ hasCompartment(?L2,?C2) ^
isGreaterThan(?S1,?S2) ^ hasSubset(?C1,?C2) -> dominates(?L1,?L2)
```

Rule 3:

S3. Compare two security labels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, if S_2 is greater than S_1, $C_2 = C_1$, then L_2 dominates L_1.

```
SecurityLabel(?L1) ^ hasSensitivityLevel(?L1,?S1) ^ hasCompartment(?L1,?C1) ^
SecurityLabel(?L2) ^ hasSensitivityLevel(?L2,?S2) ^ hasCompartment(?L2,?C2) ^
isGreaterThan(?S2,?S1) ^ sameAs(?C1,?C2) -> dominates(?L2,?L1)
```

Testing MLS Ontology with Mandatory Access Control Criteria

This section demonstrates scenario tests to use SWRL queries to detect comparable security label pairs (Figure 4.1) and incomparable security label pairs (Figure 4.3) to verify if the MLS lattice model is correctly implemented. The SWRL queries can be executed in the *SQWRLTab* in Protégé to extract information from both asserted and inferred axioms generated by the SWRL inference engine.

<u>Test Scenario 1 (Comparable Security Labels)</u>

<u>Query 1:</u>

SQ1 - Show all pairs of security labels with dominates relationships by ascending order.

```
SecurityLabel(?L1) ^ SecurityLabel(?L2) ^ dominates(?L1,?L2) ->
sqwrl:select(?L1,?L2) ^ sqwrl:orderBy(?L1,?L2)
```

The *SQ1* query represents that if there exists a *dominates* relationship between two variables L1 and L2, then select all matching pairs from the database and output L1 then L2 in ascending order. The domain and range of property dominates are pre-defined, therefore, the *dominates* relationship only exists in pairs of *SecurityLabel* instances. Run the query and the result is shown in Figure 4.1.

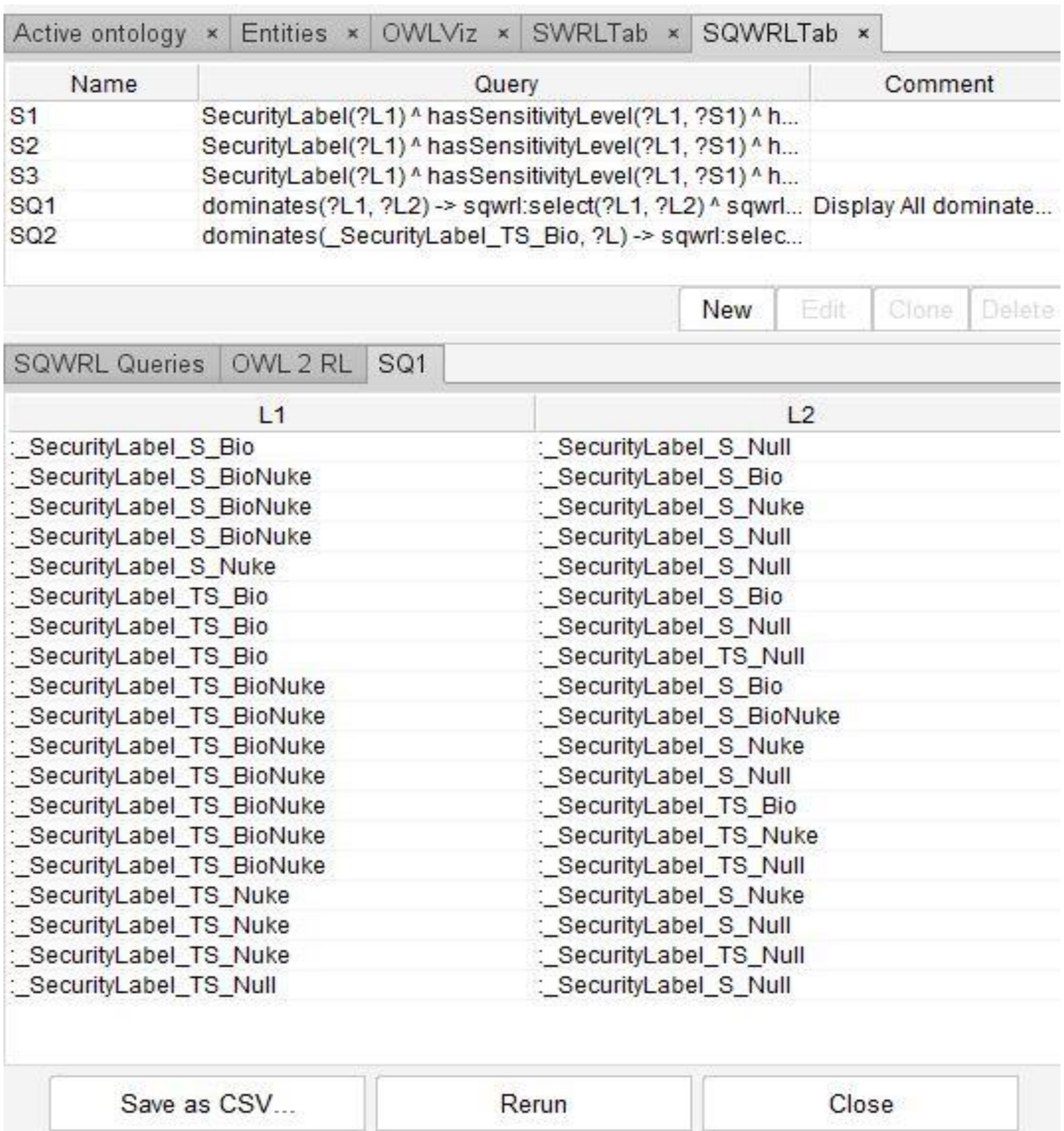


Figure 4.1. List of All Comparable Security Label Pairs

To see the *dominates* relationship applies to a specific security label, for example, *_SecurityLabel_TS_Bio*, a test query SQ2 below can show all security label instances which are dominated by it.

<u>Query 2:</u>

SQ2 - Show All Comparable Security Labels which are dominated by *_SecurityLabel_TS_Bio*

dominates(_SecurityLabel_TS_Bio, ?L) -> sqwrl:select(?L)

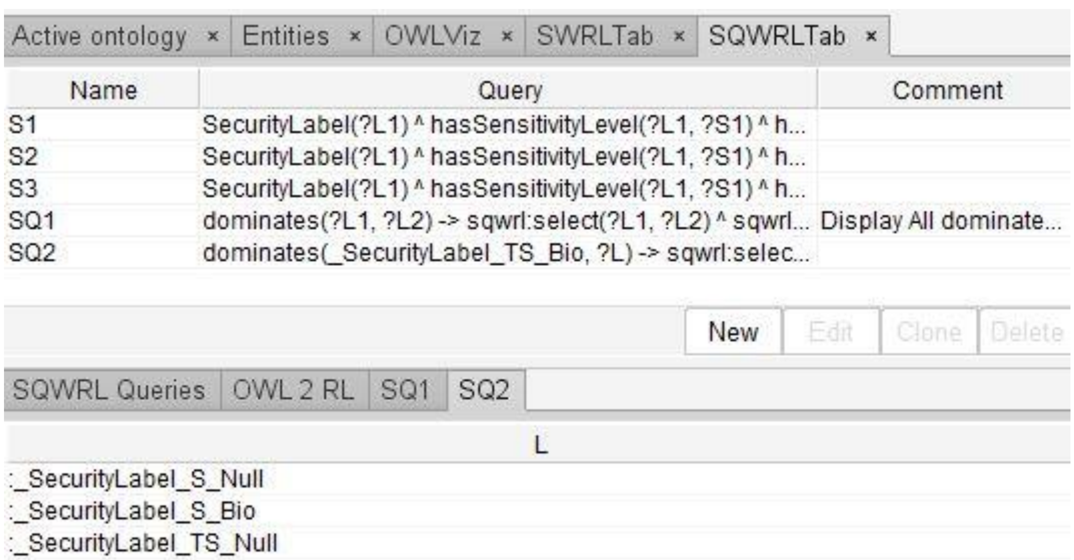


Figure 4.2. List of Comparable Security Labels of _SecurityLabel_TS_Bio

In Figure 4.2, three security label instances are returned. In lattice model (Figure 2.3), even the node *TS{Bio}* is not directly linked to the node *S{Null}*, but it dominates nodes *TS{Null}* and *S{Bio}*, which both dominate *S{Null}*. The inference engine refers to the *dominates* property's transitivity characteristics to make a inferred axiom that *TS{Bio} dominates S{Null}*.

Test Scenario 2 (Incomparable Security Labels)

In lattice model, even though the compartment *{Bio}* and *{Nuke}* are both subset of compartment *{Bio,Nuke}*. In this test, an object property *isIncomparableTo* represents the incomparable relationship between *_Compartment_Bio* and *_Compartment_Nuke*. Rule S4 will be used to create incomparable relationship between two security labels if their compartments are incomparable, and SQ3 is the query to show all security label pairs which has incomparable relationship. The result of SQ3 is shown in Figure 4.3.

Rutle 4:

S4 - Compare two security labels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, if C_1 and C_2 are incomparable, then *L1* and L_2 are incomparable.

```
SecurityLabel(?L1) ^ hasCompartment(?L1, ?C1) ^ SecurityLabel(?L2) ^
hasCompartment(?L2, ?C2) ^ isIncomparableTo(?C1, ?C2) ->
isIncomparableTo(?L1, ?L2)
```

Query 3:

SQ3 - Show all incomparable security label pairs.

```
SecurityLabel(?L1) ^ SecurityLabel(?L2) ^ isIncomparableTo(?L1, ?L2) ->
sqwrl:select(?L1, ?L2) ^ sqwrl:orderBy(?L1, ?L2)
```

Compare the result of SQ3 (Figure 4.3) to the result of SQ1 (Figure 4.1). There is no same pair of security labels in both queries' results. Hence, the implementation shows that no MAC criteria are violated. The assumption can be made that if two security labels are not comparable, then no *dominates* relationship exists between them.

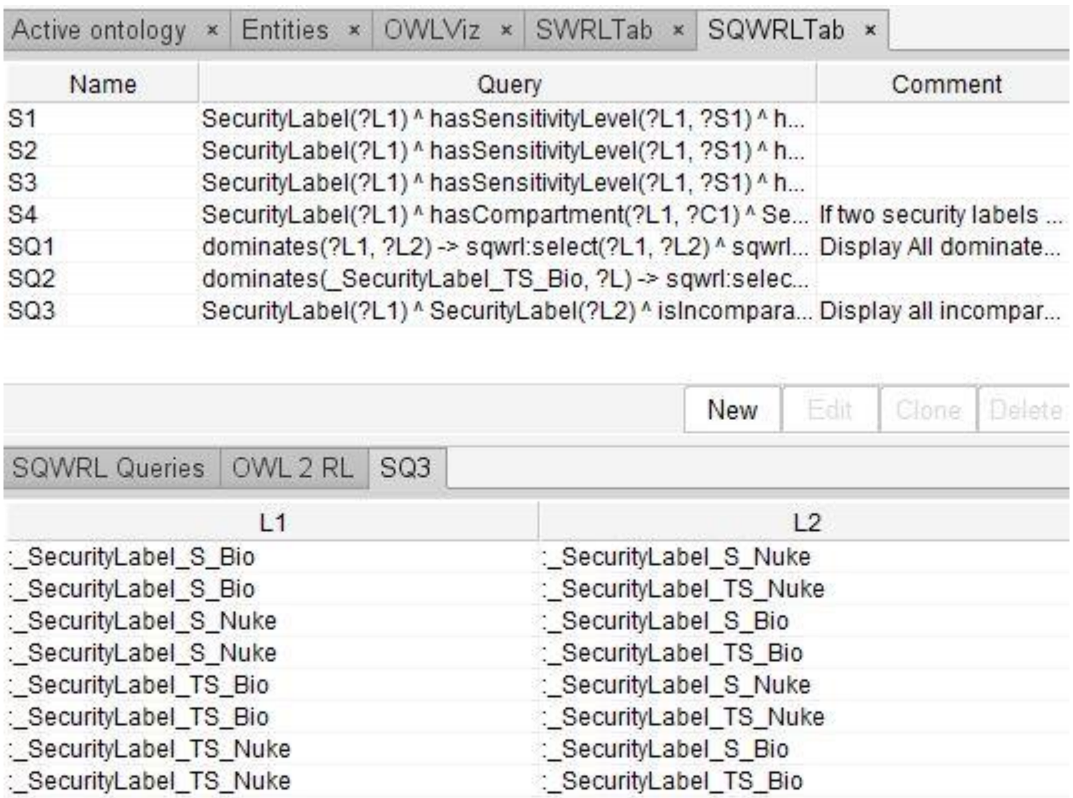


Figure 4.3. List of Incomparable Security Labels

SWRL Rules for BLP Implementation within a Single Domain

This section demonstrates the BLP models to apply the simple security property and the star property to subjects (S) and objects (O), each with its own security label. In Protégé, create a list of new Individuals with Assertions shown in Table 7.

Table 7. Subject and Object Individuals with Assertions

Class	Individual	Assertion
Subject	_Subject_1	hasSecurityLabel _SecurityLabel_S_BioNuke
	_Subject_2	hasSecurityLabel _SecurityLabel_TS_Null
	_Subject_3	hasSecurityLabel _SecurityLabel_S_Bio
	_Subject_4	hasSecurityLabel _SecurityLabel_TS_Bio
	_Subject_5	hasSecurityLabel _SecurityLabel_TS_Nuke
	_Subject_6	hasSecurityLabel _SecurityLabel_S_Null
	_Subject_7	hasSecurityLabel _SecurityLabel_TS_BioNuke
	_Subject_8	hasSecurityLabel _SecurityLabel_S_Nuke
Object	_Object_1	hasSecurityLabel _SecurityLabel_S_Nuke
	_Object_2	hasSecurityLabel _SecurityLabel_S_Null
	_Object_3	hasSecurityLabel _SecurityLabel_TS_Bio
	_Object_4	hasSecurityLabel _SecurityLabel_TS_Null
	_Object_5	hasSecurityLabel _SecurityLabel_S_BioNuke
	_Object_6	hasSecurityLabel _SecurityLabel_S_Bio
	_Object_7	hasSecurityLabel _SecurityLabel_TS_BioNuke
	_Object_8	hasSecurityLabel _SecurityLabel_TS_Nuke

Simple Security Property

The "no read up" policy states that a subject (*S*) at a security level (*sl(S)*) may not read an object (*O*) if the security level (*sl(O)*) of the object is higher than the security level(*sl(S)*) of the subject. So the subject can read the object when:

$$sl(O) \leq sl(S)$$

Therefore, *canRead* can utilize the pre-defined *dominates* relationship between security labels. R5 defines that if the security label of the subject *SL* dominates the security label of the object, then the subject can read the object. In addition, R6 defines that if the subject's security label is equal to object's security label, then they exist *canRead* relationship, and RQ4 queries a complete list of canRead relationships in this ontology(Figure 3.7).

Rule 5:

S5 - If *sl(S) dominates sl(O), then sl(S) canRead sl(O).* This rule expresses that if the subject has higher classification than the object, then apply the *canRead* relationship between these two variables.

> *Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S,?SL) ^*
> *hasSecurityLabel(?S,?OL) ^ dominates(?SL,?OL) -> canRead(?S,?O)*

Rule 6:

S6. If *sl(S) = sl(O), then sl(S) canRead sl(O).* This rule expresses that if the subject and the object have the same classification, then apply *canRead* relationship to these two variables.

> *Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S,?SL) ^*
> *hasSecurityLabel(?O,?OL) ^ sameAs(?SL,?OL) -> canRead(?S,?O)*

Query 4:

SQ4 - Show the list of c*anRead Objects* of each *Subject* , both with their security labels in order of the Subject, then by the Object (Figure 4.4).

> *Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S, ?SL) ^*
> *hasSecurityLabel(?O, ?OL) ^ canRead(?S, ?O) -> sqwrl:select(?S, ?SL, ?O,*
> *?OL) ^ sqwrl:orderBy(?S,?O)*

SQWRL Queries | OWL 2 RL | SQ4

S	SL	O	OL
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_2	:_SecurityLabel_S_Null
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_2	:_SecurityLabel_S_Null
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_2	:_SecurityLabel_S_Null
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_2	:_SecurityLabel_S_Null
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_2	:_SecurityLabel_S_Null
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_8	:_SecurityLabel_TS_Nuke
:_Subject_6	:_SecurityLabel_S_Null	:_Object_2	:_SecurityLabel_S_Null
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_2	:_SecurityLabel_S_Null
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_8	:_SecurityLabel_TS_Nuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_2	:_SecurityLabel_S_Null

Figure 4.4 Query Result of SQ4

To verify the implementation, the Example 1 in Chapter 2 states that a subject with security clearance *TS:{bio}* cannot read the object with security classification *TS:{bio,chem}* because they both have top secret sensitivity level ,but the compartment of the object is higher than (*hasSubset*) the subject's. In the ontology, the minor difference is that this project uses *{bio, nuke}* instead of *{bio,chem}*. The consumption is verified that the subject with *_SecurityLabel_TS_Bio* can only read the objects with four types of security clearances: *_SecurityLabel_TS_Bio, _SecurityLabel_TS_Null, _SecurityLabel_S_Bio and _SecurityLabel_S_Null.*

Example 5 also discussed the scenario that a subject with clearance *TS_{}* can read the object with classification *S_{}.* There is a matching record in Figure 4.4 shows that *_Subject_2 (hasSecurityLabel _SecurityLabel_TS_Null) canRead _Object_2 (hasSecurityLabel _SecurityLabel_S_Null)*.

<u>* (Star) Property</u>

The "no write-down" policy states that a subject at a given security level may not write to any object at a lower security level. The *canWrite* relationship exists when *sl(S) ≤ sl(O)*. *canWrite* utilize the *dominates* in the inverse way of *canRead*:

<u>Rule 7:</u>

S7 - If *sl(O) dominates sl(S),* then *sl(S) canWrite sl(O).* This rule expresses that if the classification of the object *dominates* (lower than) the clearance of the subject, then apply the *canWrite* relationship to these two variables.

Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S, ?SL) ^ hasSecurityLabel(?O, ?OL) ^ isDominatedBy(?SL, ?OL) -> canWrite(?S, ?O)

<u>Rule 8:</u>

S8 - If *sl(S) = sl(O),* then *sl(S) canWrite sl(O).* This rule expresses that if the subject and the object have equal classification, then apply the *canWrite* relationship to these two variables.

Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S,?SL) ^ hasSecurityLabel(?O,?OL) ^ sameAs(?SL,?OL) -> canWrite(?S,?O)

Query 5:

SQ5 - Show the list of *canWrite Objects* of each *Subject* , both with their security labels in order of the Subject, then by the Object (Figure 4.5).

SQWRL Queries | OWL 2 RL | SQ5

S	SL	O	OL
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_1	:_SecurityLabel_S_BioNuke	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_2	:_SecurityLabel_TS_Null	:_Object_8	:_SecurityLabel_TS_Nuke
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_3	:_SecurityLabel_S_Bio	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_4	:_SecurityLabel_TS_Bio	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_5	:_SecurityLabel_TS_Nuke	:_Object_8	:_SecurityLabel_TS_Nuke
:_Subject_6	:_SecurityLabel_S_Null	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_6	:_SecurityLabel_S_Null	:_Object_2	:_SecurityLabel_S_Null
:_Subject_6	:_SecurityLabel_S_Null	:_Object_3	:_SecurityLabel_TS_Bio
:_Subject_6	:_SecurityLabel_S_Null	:_Object_4	:_SecurityLabel_TS_Null
:_Subject_6	:_SecurityLabel_S_Null	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_6	:_SecurityLabel_S_Null	:_Object_6	:_SecurityLabel_S_Bio
:_Subject_6	:_SecurityLabel_S_Null	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_6	:_SecurityLabel_S_Null	:_Object_8	:_SecurityLabel_TS_Nuke
:_Subject_7	:_SecurityLabel_TS_BioNuke	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_1	:_SecurityLabel_S_Nuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_5	:_SecurityLabel_S_BioNuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_7	:_SecurityLabel_TS_BioNuke
:_Subject_8	:_SecurityLabel_S_Nuke	:_Object_8	:_SecurityLabel_TS_Nuke

Figure 4.5. Query Result for SQ5

Look at Figure 4.5, shows all pairs of *canWrite* relationships which apply to the combination of subject and object variables. Each record shows a subject with a lower or equal clearance *canWrite* the object with a higher or equal classification. The following three records improve the hypotheses discussed in Example 2., Example 3. and Example 7:

1. _Subject_4 with *_SecurityLabel_TS_Bio canWrite _Object_7* with *_SecurityLabel_TS_BioNuke.*

2. _Subject_6 (hasSecurityLabel _SecurityLabel_S_Null) canWrite _object_4 (hasSecurityLabel __SecurityLabel_TS_Null)

3. _Subject_3 (hasSecurityLabel _SecurityLabel_S_Bio) canWrite _object_7 (hasSecurityLabel __SecurityLabel_TS_BioNuke)

Additional Notes for Implementation

Unlike other query languages of Protégé, SWRL queries only extract the information from assertional knowledge (relationships between individuals). It is very important to make sure the actual assertions are made for each individual. In OWL, it's not wrong to leave the object property assertions blank, but the inference engine cannot make any inferred assertion without assertional knowledge input. For example, to apply dominance rule S1 with two given variables L1 *(_SecurityLabel_TS_Bio*) and L2(*_SecurityLabel_TS_Null*). Each must be explicitly defined with sensitivity level and compartment. If L1 does not have a clear classification of its compartment C1, even it has a compartment instance *Bio* on terminology side, but in the rule the two conditions - hasCompartment(*?L1*,*?C1*) and has Subset(?C1,?C2) are not fulfilled.

SecurityLabel(?L1) ^ hasSensitivityLevel(?L1,?S1) ^ hasCompartment(?L1,?C1) ^ SecurityLabel(?L2) ^ hasSensitivityLevel(?L2,?S2) ^ hasCompartment(?L2,?C2) ^ sameAs(?S1,?S2) ^ hasSubset(?C1,?C2) -> dominates(?L1,?L2)

CHAPTER FIVE

CONCLUSION

This project set an experimental solution for MLS policy in OWL by leveraging semantic web technologies and concepts. The proposed methodology consists of three stages. The first stage is modeling security level follows the MLS concepts. The second stage uses semantic web rule language to apply dominance rules adhering to MAC criteria. The third stage implements the ontology with BLP properties within a single domain. Test queries verify that classified information can only be accessed by authorized users. The results indicate that the MLS policy can be adopted within semantic web infrastructure.

According to the Semantic Scholar, this ontology is the first MLS practice in research studies. It has potentials for organizations to apply this security policy to protect sensitive data.

Future Work

Semantic web also allows connection to multiple ontologies in different domains. The future work can extend the current implementation to MLS multi-domain access control with trust agreement. This will build an extra layer of protection when sharing data across the organizations.

REFERENCES

43.6. Multi-Level Security(MLS) (n.d.) Red Hat Documentation. Retrieved from https://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/sec-mls-ov.html

Abadi, A., Ben-Azza, H., & Sekkat, S. (2018). *Improving integrated product design using SWRL rules expression and ontology-based reasoning.* Procedia Computer Science, *127*, 416–425. https://doi.org/10.1016/j.procs.2018.01.139

Bell, D. E. (2005). *Looking back at the bell-la padula model.* In Computer security applications conference, 21st annual. IEEE.

Berners-Lee, T., Hendler, J & Lassila, O. (2001). *The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities.* ScientificAmerican.com. Retrieved from https://www.researchgate.net/publication/225070375_The_Semantic_Web_A_New_Form_of_Web_Content_That_is_Meaningful_to_Computers_Will_Unleash_a_Revolution_of_New_Possibilities

Denning, D. E. (1976). *A lattice model of secure information flow.* Communications of the ACM, 19(5), 236–243. https://dl.acm.org/doi/pdf/10.1145/360051.360056

Elliott Bell D. (2011) *Bell–La Padula Model.* In: van Tilborg H.C.A., Jajodia S. (eds) *Encyclopedia of Cryptography and Security*. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_811

Focardi, R., & Gorrieri, R. (2003). *Foundations of Security Analysis and Design: Tutorial Lectures (Vol. 2171)*. SPRINGER.

Mandatory Access Control.(n.d.). NIST. Retrieved from https://csrc.nist.gov/glossary/term/mandatory_access_control

Matsokis, A., & Kiritsis, D. (2011). *Ontology applications in PLM.* International Journal of Product Lifecycle Management, 5(1), 84. https://doi.org/10.1504/ijplm.2011.038104

Musen, M.A. *The Protégé project: A look back and a look forward.* AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.

O'Connor, M., Knublauch, H., Tu, S., & Musen, M. (2005). *Writing Rules for the Semantic Web Using SWRL and Jess.* Research Gate. https://www.researchgate.net/profile/Martin-Oconnor-7/publication/239616230_Writing_Rules_for_the_Semantic_Web_using_SWRL_and_Jess/links/53d08ce10cf25dc05cfe4861/Writing-Rules-for-the-Semantic-Web-using-SWRL-and-Jess.pdf

Panossian, G. (2019). *Multi-level secure data dissemination* (Master's thesis). California State University, San Bernardino. https://scholarworks.lib.csusb.edu/etd/946/

Roy, S., Dayan, G.S., & Holla, V. (2018). *Modeling Industrial Business Processes for Querying and Retrieving Using OWL+SWRL*. OTM Conferences.

Rubin, D. L., Knublauch, H., Fergerson, R. W., Dameron, O., & Musen, M. A. (2005). *Protégé-OWL: Creating Ontology-Driven Reasoning Applications with the Web Ontology Language.* AMIA Annual Symposium Proceedings, 2005, 1179.

Sanger, D. E., Perlroth, N., Thrush, G., & Rappeport, A. (2018). *Marriott data breach is traced to chinese hackers as u.s. readies crackdown on beijing,* The New York Times. Retrieved from https://www.nytimes.com/2018/12/11/us/politics/trumpchina-trade.html

Son, Joon. (2008). *Covert timing channel analysis in MLS real-time systems.* Theses and Dissertations Collection, Digital Initiatives, University of Idaho Library. https://www.lib.uidaho.edu/digital/etd/items/etd_237.html

Thomas, Jason. (2019). *A Case Study Analysis of the U.S. Office of Personnel Management Data Breech.* 10.13140/RG.2.2.36670.23360.