Developing and Building a Multi-Level Security Access Control Policy Using OWL

Ruting Bai
California State University, San Bernardino
Joon Son*
California State University, San Bernardino

Abstract

This paper presents the development of a Multi-Level Security (MLS) lattice model using Semantic Web technologies, specifically the Web Ontology Language (OWL), to design and validate Mandatory Access Control (MAC) systems with Bell-LaPadula (BLP) properties. The Semantic Web, often referred to as the web of data, enhances the existing World Wide Web by facilitating data processing and management through machine-readable formats. OWL, a logic-based language within the Semantic Web framework, is designed to represent complex knowledge effectively in semantic forms. By employing an MLS ontology, this research defines dominance relationships within the lattice model and conducts various queries to determine if a subject with a particular security clearance can access (read/write) an object with a specific security classification. Utilizing BLP properties ensures strict adherence to information flow protocols, permitting data transfer only from entities with lower classification to those with higher. In addition, we propose future research directions which can be extended from our ontology model.

Keywords: semantic web, Mandatory Access Control, OWL, MLS, SWRL
JEL: M15, O30

*contact author: Json@csusb.edu

1 Introduction

1.1 Research Motivation

Web development has continuously evolved since the inception of the Internet in 1962. Initially, navigating the web required specialized knowledge. In the 1990s, Sir Tim Berners-Lee, the founder of the World Wide Web, developed three fundamental technologies—HTML, URI, and HTTP—that transformed how information is accessed, allowing even non-experts to navigate the web easily.

In the last two decades, rapid advancements in web technologies have ushered in an era where data is at the core of processing activities, with users increasingly contributing to data generation through broadcasting and social networking. Berners-Lee, Hendler, and Lassila (Berners-Lee et al., 2001) envisioned a future where the web would serve as the basis for the semantic web, enhancing data's machine readability and fostering the development of new technologies for better data storage, processing, and knowledge expression.

This vision is partially realized as semantic web technologies are now integral in sectors like healthcare and artificial intelligence, particularly for knowledge modeling. However, information security has emerged as a critical challenge, exacerbated by advancements in cloud computing, big data, and the Internet of Things. High-profile breaches, such as those at Marriott (Sanger et al., 2018) and the US Office of Personnel Management (Thomas, 2019), highlight the ongoing need to secure information systems in both the private and public sectors.

The necessity to safeguard sensitive information, such as health records, military data, and personal identity details, is more pressing than ever, given the potential for significant financial and national security repercussions following data breaches. As a result, Multi-Level Security (MLS) policies, prevalent in military operations, are increasingly adopted by various organizations to enhance their security measures. These policies mandate stringent access controls based on data classification.

As the web evolves with semantic capabilities, it poses new security challenges that require robust measures across all its layers. The absence of studies demonstrating the implementation of MLS in a semantic framework point to a critical gap and an urgent need for research in this area. As new technologies are embraced, updating access control policies to safeguard data in a semantically enriched web environment becomes imperative. policies as organizations adopt new web technologies. Therefore, security policies must also evolve to enforce information security management in the semantic web environment.

1.2 Organization

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

2 Background

2.1 Mandatory Access Control

Defined by the National Institute of Standards and Technology (NIST), Mandatory Access Control (MAC) is a type of nondiscretionary access control that imposes a uniform security level across all subjects and objects within an information system (Editor, n.d.) Its primary goal is to restrict information flow from unauthorized subjects (those without necessary security clearance) to objects (those with designated security classifications). This system is pivotal in environments where the integrity and confidentiality of information are paramount.

Historically, MAC has been intricately linked to Multi-Level Security (MLS), a framework initially developed by the defense community to enhance the protection of sensitive and confidential information (Logrippo, 2018). MLS is predominantly utilized within military systems and governmental sectors, which often demand more rigorous security measures than those required in the private sector. A core component of MLS is the Bell-LaPadula (BLP) model, which is designed to prevent the leakage of confidential information from higher to lower security levels, adhering strictly to the need-to-know principle(Bell, 2005; Hyun, 2020; Son & Alves-Foss, 2014).

The BLP model was further detailed by Denning in 1976, who introduced a lattice structure to better represent and compare the security levels of user clearances and information classifications. This lattice model facilitates the management of access controls in complex information systems where a single sensitivity level may not suffice for adequate classification of information sensitivity and user clearance.

In practice, the BLP model incorporates additional dimensions known as compartments—also referred to as categories or need-to-know bases—to refine MLS security labels or levels. In this research, we conceptualize a security level or label using a colon to separate sensitivity levels and their associated set of compartments, as per Elliott & Bell (Elliott Bell, 2011). Examples of how these labels are structured include *TopSecret:{bio, chem}*, *Secret:{}*, and *Unclassified:{nuke, bio}*, which demonstrate the granularity achievable with this method.

2.1.1 Dominance Rule

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

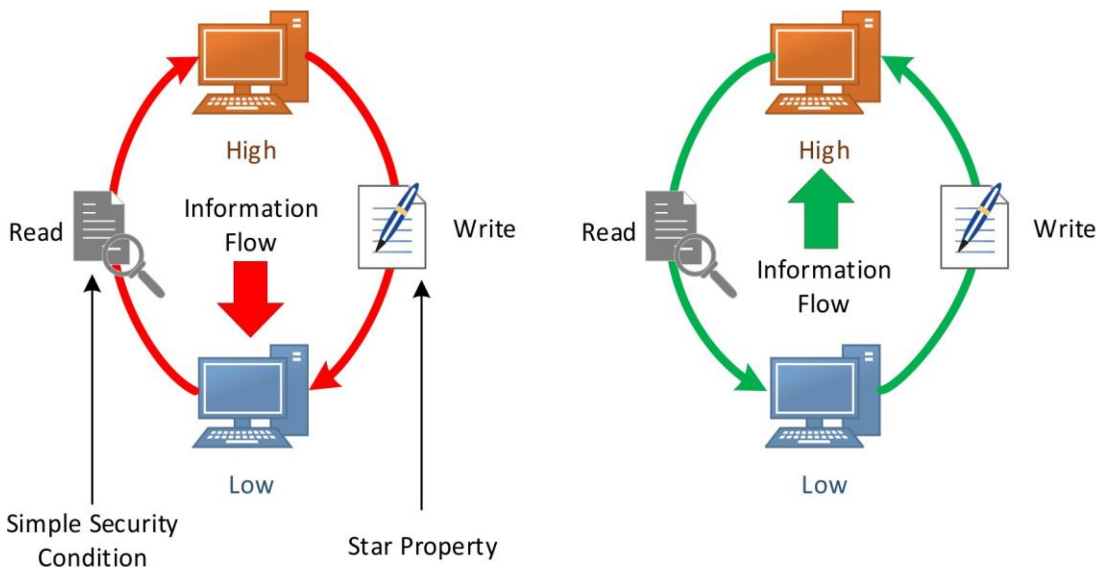
Given two security levels l_1 with sensitivity level S_1 and compartment C_1, and l_2 with sensitivity level S_2 and compartment C_2. We write $l_1 \leq l_2$, meaning l1 is dominated by (is less than) l_2 or l_2 dominates (is greater than) l_1 when

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

2.2 BLP Security Policy

The Bell-LaPadula (BLP) model is fundamental to security protocols within information systems, ensuring that every subject and object is assigned at least one security label. To prevent information flow from entities with higher sensitivity levels to those with lower sensitivity levels within the information system, two pivotal security properties aimed at regulating information flow: the simple security property and the star property (see Figure 1).

Figure 1
Left side: Without BLP properties, information can flow from high to low. The simple security condition prevents low from reading high. The star property prevents high from writing to low. Right side: With BLP properties, information can only flow from low to high.



2.2.1 Simple Security Policy

Also known as the “no read-up” policy, the BLP model states that a subject with a 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

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

<u>Example 1.</u> *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}. Since {bio} is a subset of {bio, chem}, Alice cannot read O1 because sl(Alice) ≤ sl(O1).*

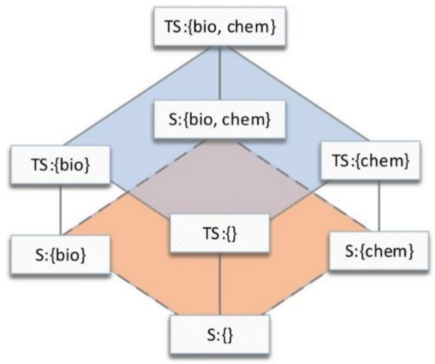
*2.2.2 *(Star) Property*

Also known as the “no write-down” policy, this states that a subject with a 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 to the object when

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

<u>Example 2.</u> *Referring to the same scenario in Example 1, sl(Alice) = TS:{bio} and sl(O1) = TS:{bio, chem}. In this case, Alice can write to O1 as sl(Alice) ≤ sl(O1).*

Figure 2
Lattice Structure (Hyun, 2020)



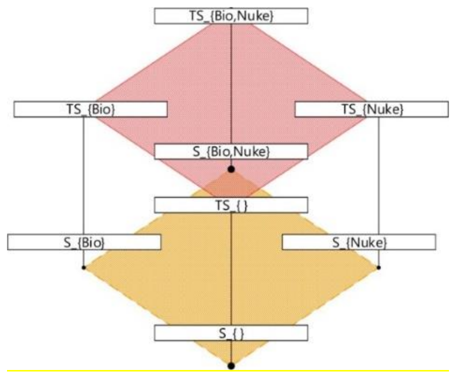
Example 3. *The diagram in Figure 2 depicts the partial ordering (≤) over the MLS security levels as a lattice. Assuming Bob is granted a security clearance TS:{}, namely, sl(Bob) = TS:{}, and Frank is granted a security clearance S:{}, namely, sl(Frank) = S:{}. Two objects, O2 is classified as TS:{}, 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), so Bob can read and write O2. Similarly, since sl(Frank) = S:{} = sl(O3), Frank can read and write 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 cannot read O2.*

Example 4. *Attaching compartments to sensitivity levels provides more flexibility for information classification in a complex information system. Figure 2 shows that there is no partial ordering between TS:{} and S:{bio} because they are not comparable. This means that no operation, such as read or write, should be performed between them.*

2.3 Multi-Level Security

The lattice structure of MLS with the BLP model (Figure 3) is formed with vertices connected by edges. The model distinguishes two sets of vertices with different colors based on their hierarchy levels. Each security label (SL(s_i, c_i)) has two components: sensitivity level s_i and compartment c_i. Sensitivity levels are hierarchically defined, ranging from high to low: "Top Secret" ≥ "Secret" ≥ "Classified" ≥ "Unclassified". Compartments are defined as {Bio, Nuke} ⊇ {Bio} | {Nuke} ⊇ {}. Vertices in the red area represent labels with "Top Secret" clearance (noted as TS), and vertices in the orange area represent labels with "Secret" clearance (noted as S) (Hyun, 2020).

Figure 3
Lattice Model (Hyun, 2020)



Example 5: *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"). Conversely, S_{} is prohibited from reading up to TS_{} to prevent information from leaking from a higher classification to a lower classification. Similarly, S_{} can write up to TS_{}, but TS_{} cannot write down to S_{}.*

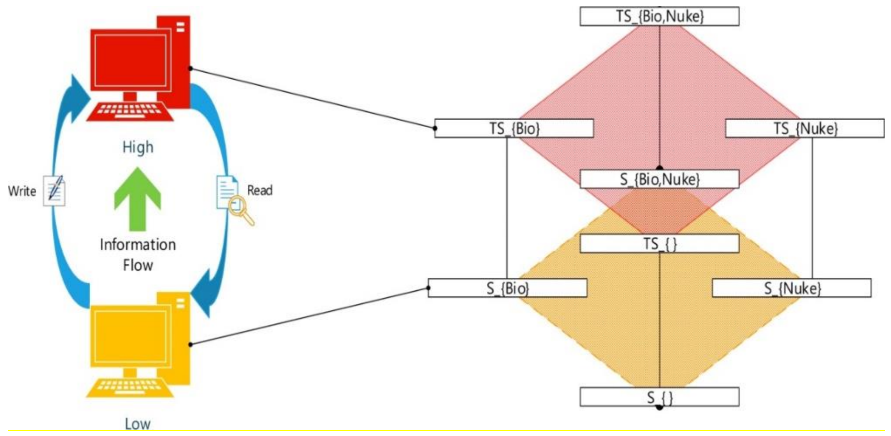
Moreover, the BLP model does not grant users with "Top Secret" clearance access to all objects. An additional need-to-know restriction, known as a compartment, is applied to block irrelevant users from accessing confidential information (Denning, 1976; Panossian, 2019).

Example 6: *Based on Figure.3, assume Mary, with security clearance TS_{}, is trying to read/write an object file with security classification S_{Nuke}. Mary passes the first criterion 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 criterion. Since she does not have the {Nuke} compartment, she cannot be granted access to objects with {Nuke}. This example illustrates 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 is established, 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 prevent information from leaking from higher classification to lower classification (Figure 4), MLS enforces the simple security property and the star property. Example 7 and Example 8 will each discuss scenarios illustrating how each BLP property ensures information flows from lower classification to higher classification. These examples will demonstrate the rules for determining if a subject (S) can read/write an object (O) based on their security labels.

Figure 4
Information Flow with BLP (Hyun, 2020)

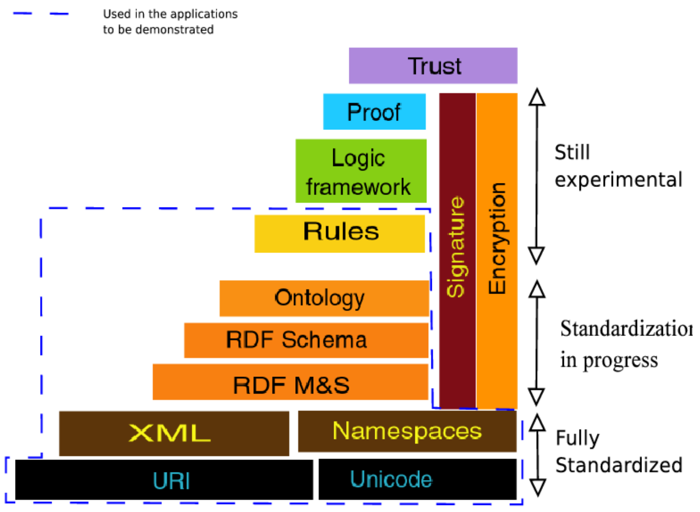


Example 7. *Assuming a person A (s_i) has the security clearance S_{Bio} and an object (o_i) has the classification TS_{Bio, Nuke}, s_i cannot read o_i because $SL(s_i) \leq SL(o_i)$. However, s_i can read any o_j when $SL(s_i) \geq SL(o_j) \geq SL(o_k)$. For instance, $SL(o_j)$ is equal to S_{Bio}, and(o_k) is equal to S_{}. (Hyun, 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 $SL(s_i) \leq SL(o_j)$, allowing information to flow from lower level security clearance to higher level security clearance. However, person A will not be able to write to o_k.* (Hyun, 2020)

2.4 Introduction to Semantic Web and Technologies

Figure 5
The Layers of Semantic Web Technology



The 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 explicitly represented, allowing the data to be machine-readable to improve information retrieval and produce more useful work. Several semantic web technologies are used in this project, including RDF, OWL, SWRL, and Protégé (Figure 5). Each of these technologies will be briefly introduced.

2.4.1 The Resource Description Framework (RDF)

RDF is a fundamental component of the Semantic Web, built on top of HTML, HTTP, and XML to express the semantic meaning of knowledge. A resource can be anything and must be uniquely identified and referenced via an Internationalized Resource Identifier (IRI). Knowledge is expressed in a list of statements called triples, which follow a simple schema with three components: subject, property, and object. In RDF, both the subject and the property must be IRIs, while the object of the triple can be either an IRI or a literal (datatype).

2.4.2 The W3C Web Ontology Language (OWL)

OWL is a Semantic Web language designed to represent rich and complex knowledge based on description logic. It describes classes, individuals, and properties, transforming common knowledge from philosophy and mathematics into a formal language in the form of RDF to impart semantic meaning. This transformation makes the knowledge machine-understandable. The goal of building an OWL ontology is to create a model that represents a subject matter with individual entities, types of entities, and types of relationships, as well as to support automated reasoning.

In OWL, a class represents a group of entities of interest, an individual is an instance of a class, and a property defines the relationship between subjects and objects. Description logic separates the terminological knowledge base (TBox) from the assertional knowledge base (ABox). The TBox describes the relationships between classes when defining the model, while the ABox describes how individuals are related to each other.

Several previous studies have adopted ontology-based approaches to various access control policies (Kayes et al., 2018; Kiran & Nalini, 2022; Li & Zhang, 2019; Veloudis et al., 2019). However, none of these studies have specifically explored access control within Multi-Level Secure (MLS) systems.

2.4.3 The Semantic Rule Language (SWRL)

SWRL combines OWL ontologies and Datalog expressions, applying Datalog rules to OWL ontologies in the form of "If... then..." statements. SWRL rules are structured as "Antecedent -> Consequent," where the term "Antecedent" refers to the rule body and "Consequent" refers to the rule head (O'Connor et al., 2005). The body represents the "If..." statement, and the head represents the "then..." statement. An example of an SWRL rule is:

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

This example illustrates the rule stating that "If security label a is equal to security label b, then a can read b." For the implementation of BLP in Section 4, such rules will be created to establish the read/write relationships between subjects and objects. Each rule will be discussed and the implementation output will be shown.

Without SWRL, the ontology can still be implemented by manually creating assertions in the editor. However, if an ontology has hundreds of assertions, it becomes highly inefficient to process the data manually without using an inference engine. SWRL provides automated reasoning functions, allowing the inference engine to create inference assertions within milliseconds. Moreover, modifying an individual can cause several assertions to change. SWRL can handle the rest of the modifications, significantly improving work efficiency.

Several studies have demonstrated that using SWRL can enhance business process management. According to Abadi, Ben-Azza, and Sekkat (Abadi et al., 2018), SWRL is the only tool that combines ontology to model information and decision-making rules for industrial applications. Matsokis and Kiritsis (Matsokis & Kiritsis, 2011) also suggested using SWRL to extend OWL models and develop a learnable approach in production management. Furthermore, another study presented that SWRL supports business knowledge management in industrial business processes. (Roy et al., 2018)

2.4.2 Protégé

Protégé is an open-source ontology editor developed by the 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 and offers visualization support and extensive built-in tools to facilitate ontology construction. Protégé provides a variety of features to assist developers in creating, modifying, and managing ontologies

(Rubin et al., 2005) such as simple and customizable user interface, support for collaborative work, visual support for ontology expressions, built-in reasoners for checking consistency and inference, multiple formats for exporting ontologies to other platforms, and web version compatible with the desktop version.

3 Modeling Multi-Level Security in OWL

This section will demonstrate the steps for building an MLS ontology in Protégé. The three key components of an OWL ontology are classes, properties, and individuals. To distinguish each component, this project uses the following naming conventions without spaces:

- Classes: Upper camel case (e.g., *Person*, *Animal*, *Food*)
- Properties: Lower camel case (e.g., *isGreaterThan*, *hasPet*, *movesTo*)
- Individuals: Leading underscore (e.g., *_JohnSmith*, *_Dog*, *_Pizza*)

3.1 Building MLS Ontology

3.1.1 Create Classes

The implementation begins 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*, and *Compartment*—and their subclasses. Referring to the lattice structure in Figure 3, 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*, while *BioNuke*, *Bio*, *Nuke*, and *Null* (representing *{ }*) are subclasses of Compartment.

Because OWL uses open world reasoning, if two classes are not specified to be different types of things, they are considered unknown and allowed to have intersections. To specify that there are no common members in *SecurityLabel*, *SensitivityLevel*, and *Compartment*, these three classes must be *disjoint* from each other. This means that one individual cannot be an instance of more than one of these three classes. Protégé allows users to create a list of classes and indicate disjointness using the *Create Class Hierarchy* tool. To verify the implementation, select a random class to view at the bottom of the *Class Description*. All sibling classes of the selected class should appear 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 section. Table 1 shows the full list of classes with their class hierarchy levels.

Table 1
Classes and Subclasses

Class	Subclass
Compartment	*Bio*
	BioNuke
	Nuke
	Null
SensitivityLevel	*TopSecret*
	Secret
	Confidential

Class	Subclass
	Unclassified
SecurityLabel	*TS_BioNuke*
	TS_Bio
	TS_Nuke
	TS_Null
	S_BioNuke
	S_Bio
	S_Nuke
	S_Null
Subject	
Object	

3.1.2 Create Object Properties and Inverse Properties

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

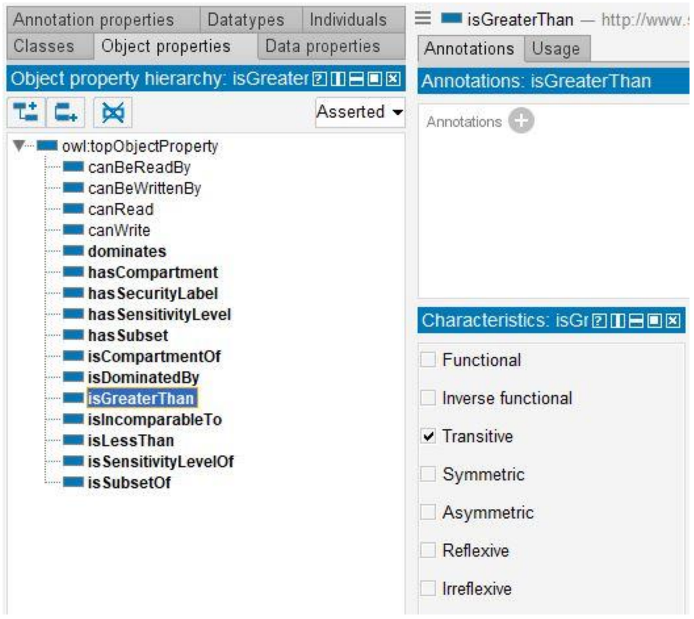
Table 2
Convert the Knowledge into RDF Triple and Property

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 properties. The first type is object properties, which link individuals to other individuals. The second type is datatype properties, which link individuals to RDF datatypes (e.g., string, integer, date, etc.). In this MLS ontology, all properties are object properties.

Properties have characteristics, which can be easily specified in Protégé (Figure 3.1). The transitive characteristic will be specified for three properties: hasSubset, isGreaterThan, and dominates. These properties have the characteristic that if X is related to Y and Y is related to Z, then X is related to Z. It is not necessary to manually add an assertion stating that X is related to Z, as the inference engine can generate the inferred axioms if the property characteristics are specified.

Figure 5
Apply Transitive Characteristic to isGreaterThan



owl:topObjectProperty
- canBeReadBy
- canBeWrittenBy
- canRead
- canWrite
- dominates
- hasCompartment
- hasSecurityLabel
- hasSensitivityLevel
- hasSubset
- isCompartmentOf
- isDominatedBy
- isGreaterThan
- isIncomparableTo
- isLessThan
- isSensitivityLevelOf
- isSubsetOf

Functional [ ]
Inverse functional [ ]
Transitive [✓]
Symmetric [ ]
Asymmetric [ ]
Reflexive [ ]
Irreflexive [ ]

Protégé also provides the option to define the domain and range of properties, similar to their meaning in mathematics. When two individuals are connected by a property in an RDF triple, the domain class specifies that the subject of the triple belongs to the domain class, while the range class specifies that the object of the triple belongs to the range class.

Table 3 lists the domain and range for each property. For example, the domain of the property *hasSensitivityLevel* 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, Pellet, can detect inconsistent assertions that conflict with the description logic expressed in the model. The reasoner can identify inconsistencies such as A (an instance of SecurityLabel) having *hasSensitivityLevel* B (an instance of Compartment), or A (an instance of Compartment) having *hasSensitivityLevel* B (an instance of SensitivityLevel).

Table 3
List of Properties' Domain and Range

Object Property	Domain	Range
hasSensitivityLevel	*SecurityLabel*	*SensitivityLevel*
hasCompartment	*SecurityLabel*	*Compartment*
hasSubset	*Compartment*	*Compartment*
isGreaterThan	*SensitivityLevel*	*SensitivityLevel*

Object Property	Domain	Range
dominates	*SecurityLabel*	*SecurityLabel*
isIncomparableTo	*SecurityLabel*	*SecurityLabel*
canRead	*Subject*	*Object*
canWrite	*Subject*	*Object*

Each object property can have an inverse property. In an RDF triple, the property links the subject to the object in one direction. Its inverse property applies this relationship from the opposite perspective. For example, if *A* is linked to *B* through property *P*, the inverse way of stating the same relationship 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 rule inferences in the next chapter, an inverse property is created for each object property (Table 4).

Table 4
Properties and Inverse Properties

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

3.1.3 Modeling Classes Expression with Property Restrictions

The third step is to apply property restrictions to model class expressions. Properties describe the relationships between individuals and can also be used as a special kind of class description to ensure that all instances of the class 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.

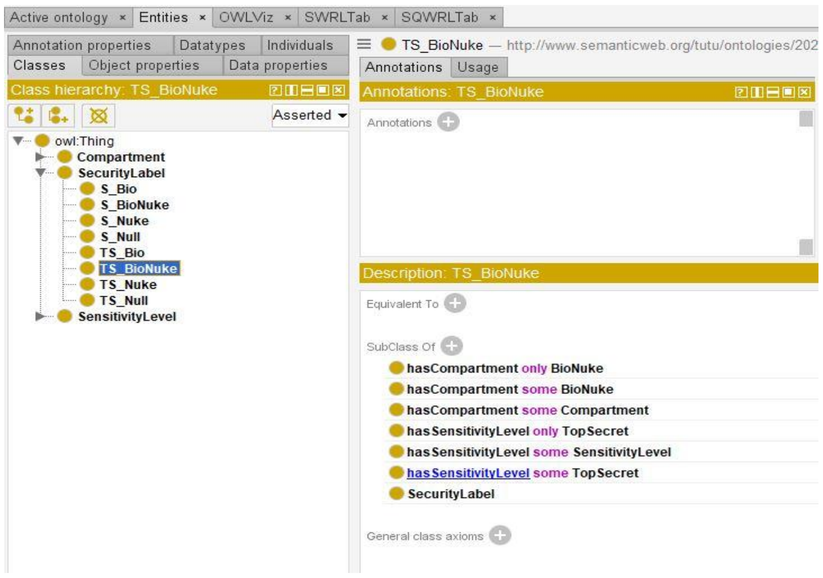
For example, to model the *TS_BioNuke* class (Figure 6), the class must meet two conditions:

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

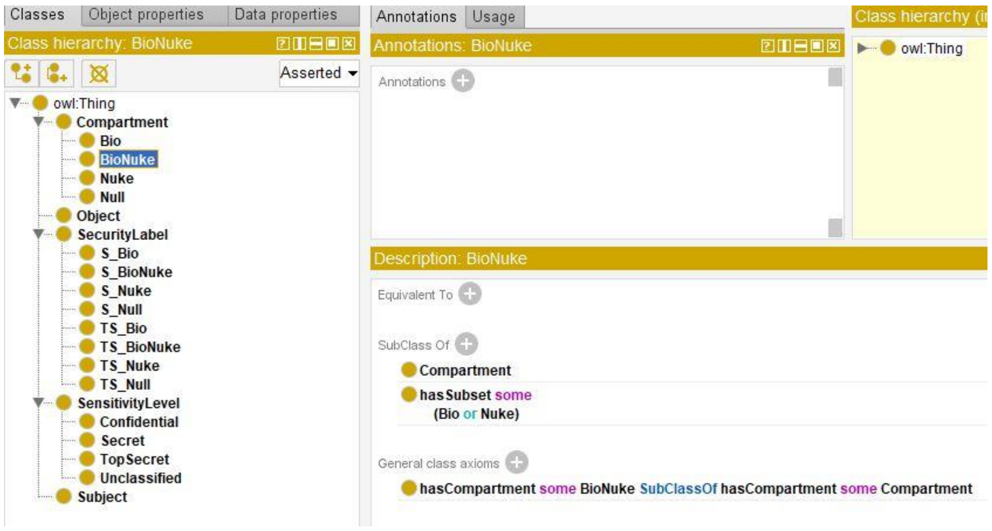
Figure 6
Security Label TS_BioNuke



hasCompartment only BioNuke
hasCompartment some BioNuke
hasCompartment some Compartment
hasSensitivityLevel only TopSecret
hasSensitivityLevel some SensitivityLevel
hasSensitivityLevel some TopSecret
SecurityLabel

Users can click on the compartment of TS_BioNuke, BioNuke, and Protégé will redirect to the class description of this class (Figure 7).

Figure 7
Compartment BioNuke



Compartment
hasSubset some (Bio or Nuke)
hasCompartment some BioNuke SubClassOf hasCompartment some Compartment

Moreover, the following table lists 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*

Class	Subclass	Property Restrictions
	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* *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*

3.1.4 Create Individuals with Property Assertions

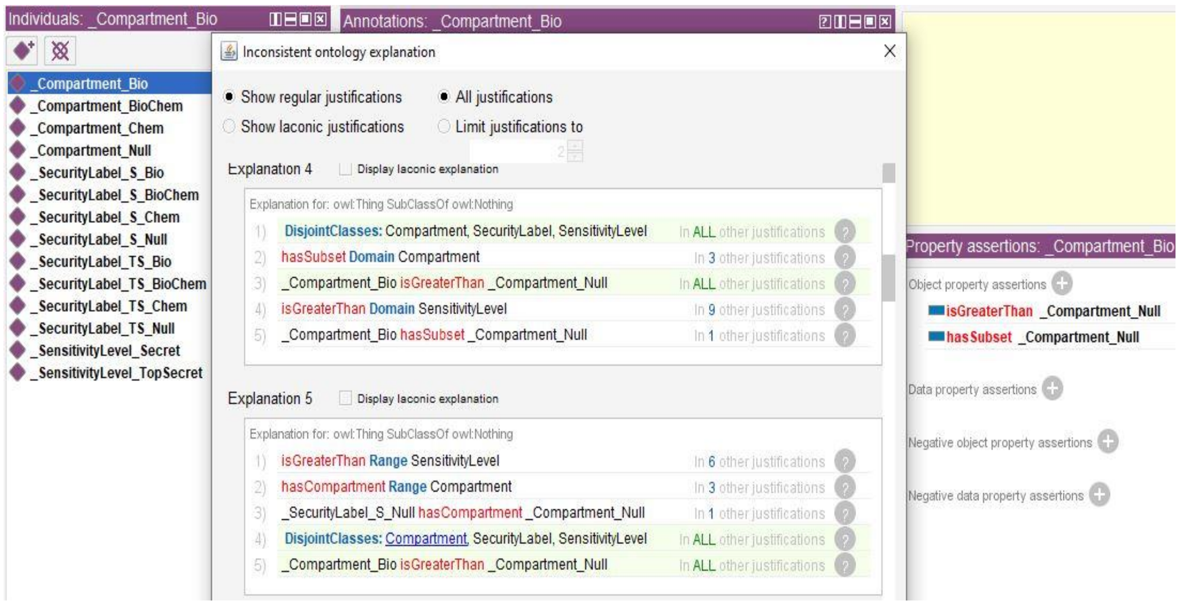
After modeling classes with property restrictions, we can then create instances with property assertions. Table 6 shows a list of individuals with their property assertions for each security label node of the lattice model.

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

At this step, the security label modeling is complete. The ontology modeling constructs terminological assertions that are applied to classes with property restrictions, while assertional knowledge is represented with individuals. For testing purposes, select the *Compartment* individual *_Compartment_Bio* and add an object property assertion to represent *_Compartment_Bio isGreaterThan _Compartment_Null*. When running the Pellet reasoner, an inconsistentOntologyException error message appears because Protégé explains (Figure 3.2) that the domain and range of *isGreaterThan* are limited to *SensitivityLevel*, which is disjoint from *Compartment*. The test assertion conflicts with the specified domain and range classes of *isGreaterThan*. This test demonstrates the reasoner's capability of catching inconsistency errors. The reasoner can be used to detect modeling errors at any step.

Figure 8
Protégé Inconsistent Ontology Explanation


Individuals: _Compartment_Bio

_Compartment_Bio
_Compartment_BioChem
_Compartment_Chem
_Compartment_Null
_SecurityLabel_S_Bio
_SecurityLabel_S_BioChem
_SecurityLabel_S_Chem
_SecurityLabel_S_Null
_SecurityLabel_TS_Bio
_SecurityLabel_TS_BioChem
_SecurityLabel_TS_Chem
_SecurityLabel_TS_Null
_SensitivityLevel_Secret
_SensitivityLevel_TopSecret

Annotations: _Compartment_Bio

Inconsistent ontology explanation

● Show regular justifications ● All justifications
○ Show laconic justifications ○ Limit justifications to 2

Explanation 4 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1) DisjointClasses: Compartment, SecurityLabel, SensitivityLevel — In ALL other justifications
2) hasSubset Domain Compartment — In 3 other justifications
3) _Compartment_Bio isGreaterThan _Compartment_Null — In ALL other justifications
4) isGreaterThan Domain SensitivityLevel — In 9 other justifications
5) _Compartment_Bio hasSubset _Compartment_Null — In 1 other justifications

Explanation 5 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1) isGreaterThan Range SensitivityLevel — In 6 other justifications
2) hasCompartment Range Compartment — In 3 other justifications
3) _SecurityLabel_S_Null hasCompartment _Compartment_Null — In 1 other justifications
4) DisjointClasses: Compartment, SecurityLabel, SensitivityLevel — In ALL other justifications
5) _Compartment_Bio isGreaterThan _Compartment_Null — In ALL other justifications

Property assertions: _Compartment_Bio

Object property assertions
isGreaterThan _Compartment_Null
hasSubset _Compartment_Null

Data property assertions

Negative object property assertions

Negative data property assertions

4 SWRL Rule Implementation for MAC And BLP

4.1 Apply Dominance Rule

This section utilizes the Semantic Web Rule Language (SWRL) to enforce dominance rules within the MLS ontology. SWRL integrates OWL and Datalog expressions into Horn-like rules, allowing for *"If …, then …"* statements. The SWRL inference engine processes a set of predefined rules to apply relationships to the matching variables. Consequently, any modification to the ontology will automatically update the inferred axioms using SWRL. SWRL not only functions as an inference engine but also transfers the inferred axioms to the OWL model, making them explicitly represented. The ontology, along with the inferred axioms generated by the inference engine, can be exported for review in a simple text editor or other semantic tools.

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

An MLS system has a dominance rule that defines a partial order (≤) over the MLS security levels. This partial ordering (≤) is defined to ensure that two security levels can always 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:

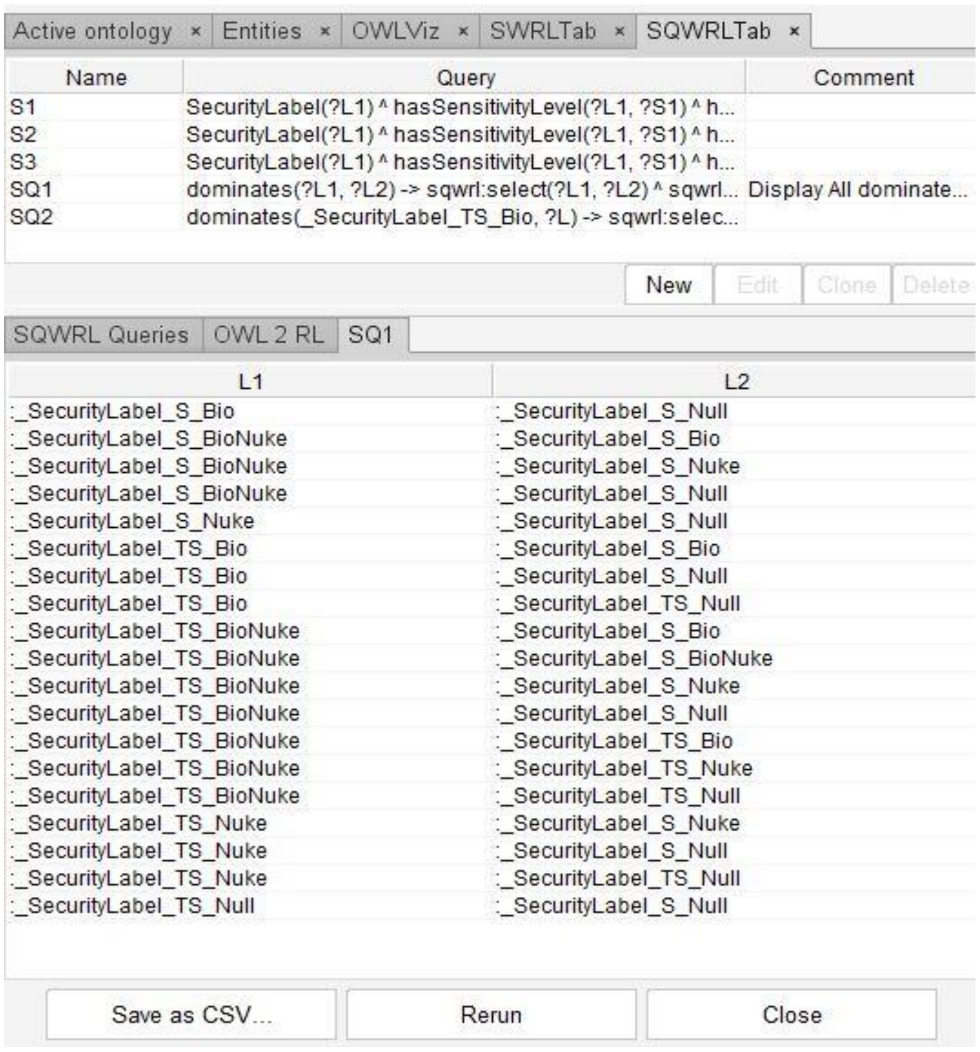
S3 – Compare two security labels $L_1 = S_1{:}C_1$ and $L_2 = S_2{:}C_2$, if S_2 is greater than S_1, $C_2 = C_1$, then L_2 dominates L_1.

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)

4.2 Testing MLS Ontology with Mandatory Access Control Criteria

This section demonstrates scenario tests using SWRL queries to detect comparable security label pairs (Figure 9) and incomparable security label pairs (Figure 10) to verify the correct implementation of the MLS lattice model. 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.

Figure 9
List of All Comparable Security Label Pairs


Active ontology × Entities × OWLViz × SWRLTab × SQWRLTab ×

Name	Query	Comment
S1	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S2	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S3	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
SQ1	dominates(?L1, ?L2) -> sqwrl:select(?L1, ?L2) ^ sqwrl...	Display All dominate...
SQ2	dominates(_SecurityLabel_TS_Bio, ?L) -> sqwrl:selec...	

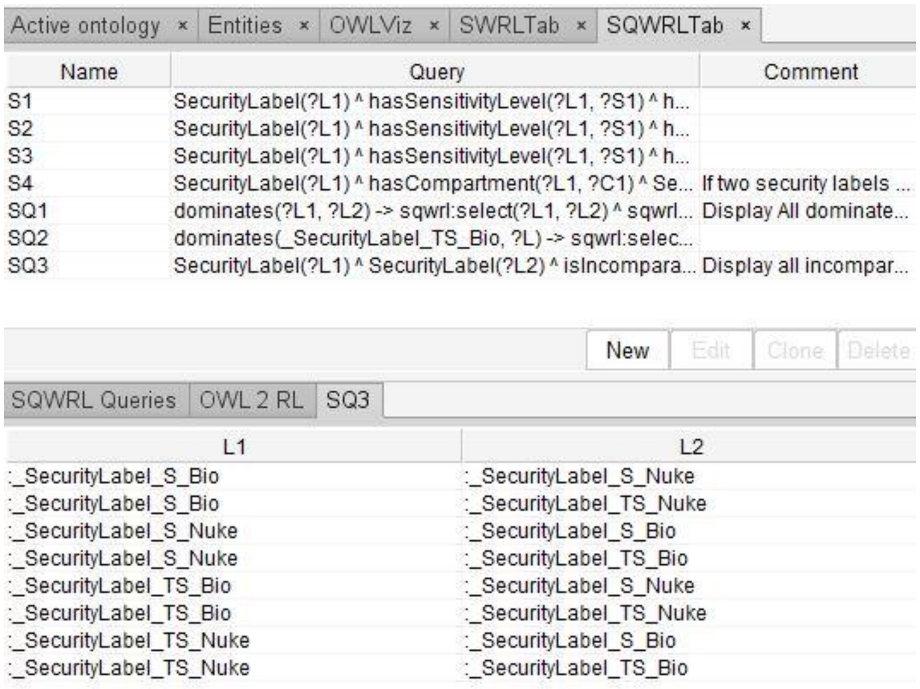
New Edit Clone Delete

SQWRL Queries OWL 2 RL SQ1

L1	L2
:_SecurityLabel_S_Bio	:_SecurityLabel_S_Null
:_SecurityLabel_S_BioNuke	:_SecurityLabel_S_Bio
:_SecurityLabel_S_BioNuke	:_SecurityLabel_S_Nuke
:_SecurityLabel_S_BioNuke	:_SecurityLabel_S_Null
:_SecurityLabel_S_Nuke	:_SecurityLabel_S_Null
:_SecurityLabel_TS_Bio	:_SecurityLabel_S_Bio
:_SecurityLabel_TS_Bio	:_SecurityLabel_S_Null
:_SecurityLabel_TS_Bio	:_SecurityLabel_TS_Null
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_S_Bio
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_S_BioNuke
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_S_Nuke
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_S_Null
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_TS_Bio
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_TS_Nuke
:_SecurityLabel_TS_BioNuke	:_SecurityLabel_TS_Null
:_SecurityLabel_TS_Nuke	:_SecurityLabel_S_Nuke
:_SecurityLabel_TS_Nuke	:_SecurityLabel_S_Null
:_SecurityLabel_TS_Nuke	:_SecurityLabel_TS_Null
:_SecurityLabel_TS_Null	:_SecurityLabel_S_Null

Save as CSV... Rerun Close

Figure 10
List of All Incomparable Security Label Pairs



Name	Query	Comment
S1	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S2	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S3	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S4	SecurityLabel(?L1) ^ hasCompartment(?L1, ?C1) ^ Se...	If two security labels ...
SQ1	dominates(?L1, ?L2) -> sqwrl:select(?L1, ?L2) ^ sqwrl...	Display All dominate...
SQ2	dominates(_SecurityLabel_TS_Bio, ?L) -> sqwrl:selec...	
SQ3	SecurityLabel(?L1) ^ SecurityLabel(?L2) ^ isIncompara...	Display all incompar...

L1	L2
:_SecurityLabel_S_Bio	:_SecurityLabel_S_Nuke
:_SecurityLabel_S_Bio	:_SecurityLabel_TS_Nuke
:_SecurityLabel_S_Nuke	:_SecurityLabel_S_Bio
:_SecurityLabel_S_Nuke	:_SecurityLabel_TS_Bio
:_SecurityLabel_TS_Bio	:_SecurityLabel_S_Nuke
:_SecurityLabel_TS_Bio	:_SecurityLabel_TS_Nuke
:_SecurityLabel_TS_Nuke	:_SecurityLabel_S_Bio
:_SecurityLabel_TS_Nuke	:_SecurityLabel_TS_Bio

4.2.1 Test Scenario 1 (Comparable Security Labels)

Query 1:

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 identifies pairs of variables L1 and L2 where a *dominates* relationship exists. It selects all matching pairs from the database and outputs L1 and then L2 in ascending order. The domain and range of the *dominates* property are pre-defined, so the *dominates* relationship only exists between pairs of SecurityLabel instances. Running the query produces the results shown in Figure 9.

To see how the *dominates* relationship applies to a specific security label, such as *_SecurityLabel_TS_Bio*, a test query (SQ2) can be used to show all security label instances that are dominated by it.

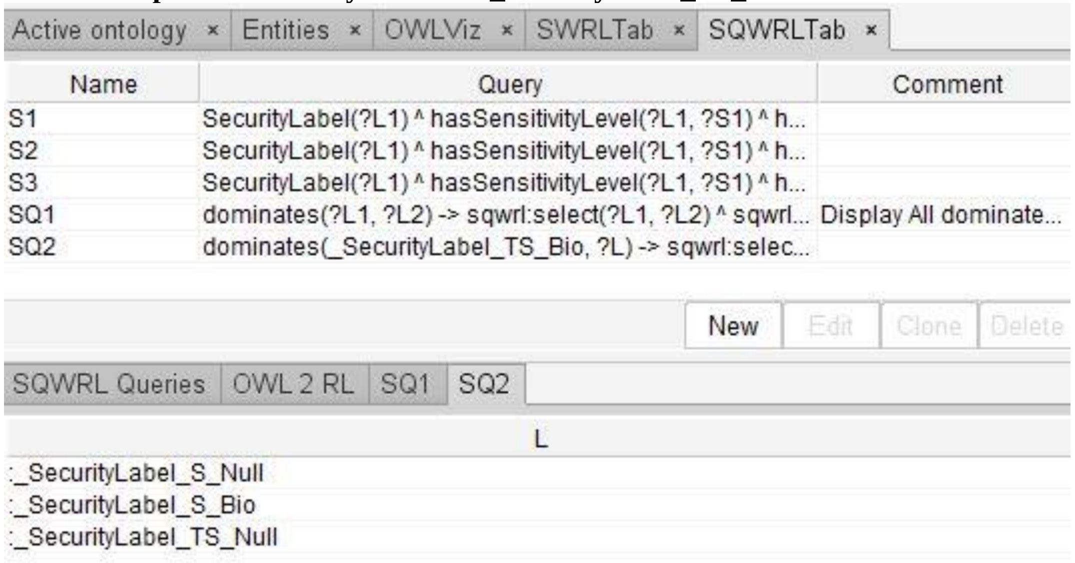
Query 2:

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

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

In Figure 11, three security label instances are returned. In lattice model (Figure 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 an inferred axiom that *TS{Bio} dominates S{Null}*.

Figure 11
List of Comparable Security Labels of _SecurityLabel_TS_Bio



Name	Query	Comment
S1	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S2	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
S3	SecurityLabel(?L1) ^ hasSensitivityLevel(?L1, ?S1) ^ h...	
SQ1	dominates(?L1, ?L2) -> sqwrl:select(?L1, ?L2) ^ sqwrl...	Display All dominate...
SQ2	dominates(_SecurityLabel_TS_Bio, ?L) -> sqwrl:selec...	

L
:_SecurityLabel_S_Null
:_SecurityLabel_S_Bio
:_SecurityLabel_TS_Null

4.2.2 Test Scenario 2 (Incomparable Security Labels)

In the lattice model, even though the compartments {Bio} and {Nuke} are both subsets of the compartment {Bio, Nuke}, an object property isIncomparableTo represents the incomparable relationship between _Compartment_Bio and _Compartment_Nuke. Rule S4 is used to create an incomparable relationship between two security labels if their compartments are incomparable. The SQ3 query shows all security label pairs that have an incomparable relationship. The result of SQ3 is shown in Figure 10.

Rule 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)
```

Comparing the result of SQ3 (Figure 10) to the result of SQ1 (Figure 9), we find that there are no identical pairs of security labels in both queries' results. Hence, the implementation shows that no MAC criteria are violated. We can assume that if two security labels are not comparable, then no dominates relationship exists between them.

4.3 SWRL Rules for BLP Implementation within a Single Domain

This section demonstrates the application of the BLP model's simple security property and star property to subjects (S) and objects (O), each with its own security label. In Protégé, create a list of new individuals with the 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

4.3.1 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. Therefore, the subject can read the object when:

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

Therefore, *canRead* can utilize the pre-defined *dominates* relationship between security labels. Rule 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. Additionally, Rule R6 defines that if the subject's security label is equal to the object's security label, then a *canRead* relationship exists. The query RQ4 generates a complete list of *canRead* relationships in this ontology (Figure 12).

<u>Rule 5:</u>

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

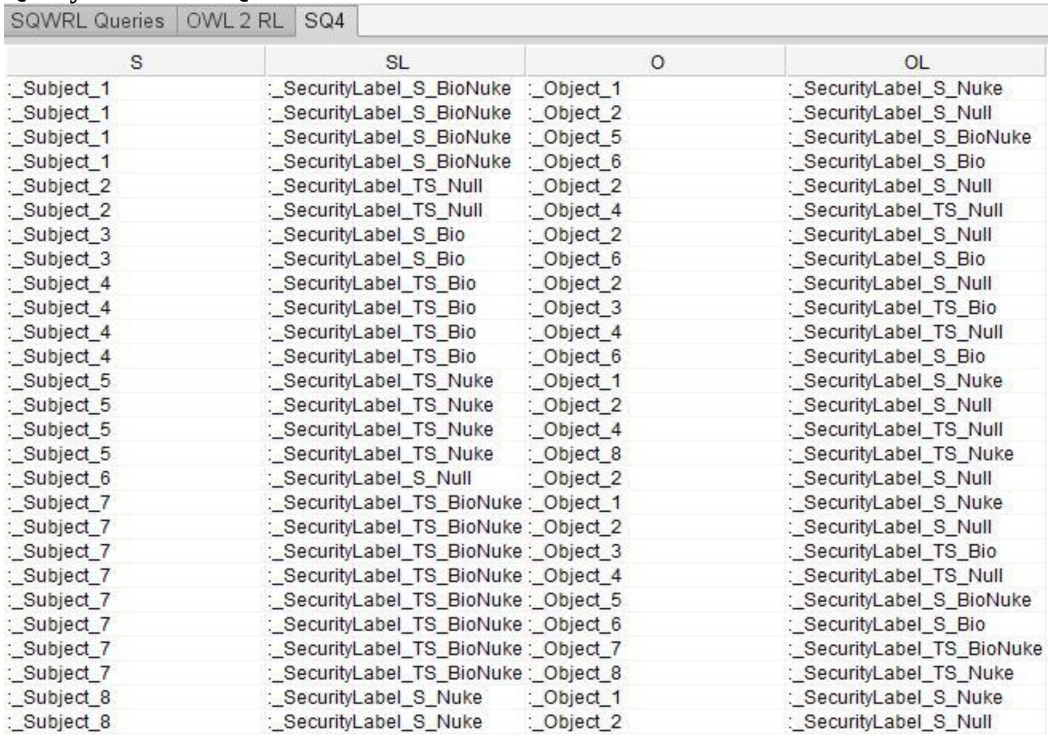
Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S,?SL) ^ hasSecurityLabel(?O,?OL) ^ sameAs(?SL,?OL) -> canRead(?S,?O)

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 12).

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

Figure 12
Query Result of 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

To verify the implementation, Example 1 in Section 2 states that a subject with security clearance *TS:_bio}* cannot read an object with security classification *TS_{bio, chem}* because, although they both have Top Secret sensitivity levels, the compartment of the object is higher (*hasSubset*) than the subject's. In the ontology, the minor difference is that this project uses *{bio,nuke}* instead of *{bio,chem}*. The assumption is verified that the subject with *_SecurityLabel_TS_Bio* can only read 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 where a subject with clearance *TS_{}* can read an object with classification *S_{}*. There is a matching record in Figure 4.4 that shows *_Subject_2* (*hasSecurityLabel _SecurityLabel_TS_Null*) can read *_Object_2* (*hasSecurityLabel _SecurityLabel_S_Null*).

*4.3.2 * (Star) Property*

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* utilizes the *dominates* in the inverse way of *canRead*:

Rule 7:

S7 - If *sl(O) dominates sl(S)*, then *sl(S) canWrite sl(O).* This rule expresses that if the classification of the object *dominates* (higher 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)

Rule 8:

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.

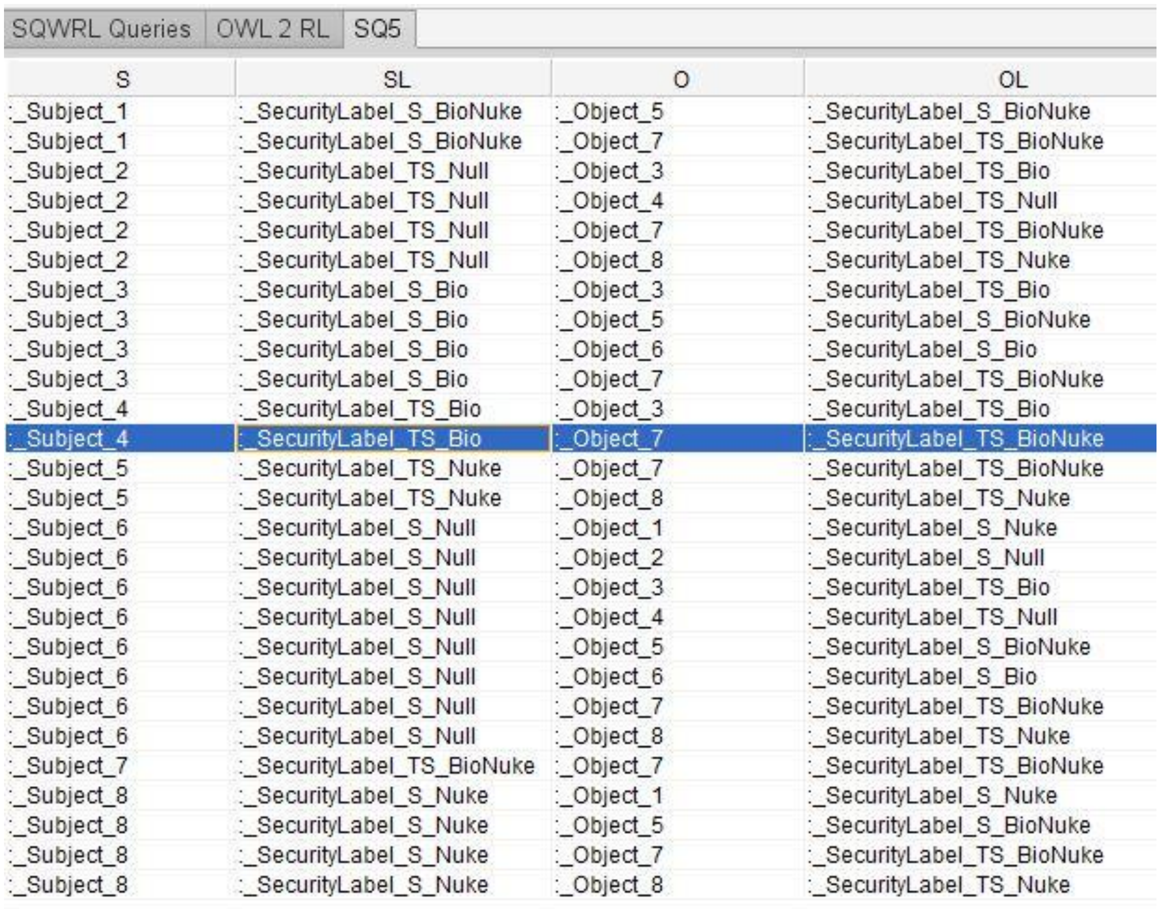
Subject(?S) ^ Object(?O) ^ hasSecurityLabel(?S,?SL) ^ hasSecurityLabel(?O,?OL) ^ sameAs(?SL,?OL) -> canWrite(?S,?O)

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).

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

Figure 13
Query Result for SQ5


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 13 shows all pairs of *canWrite* relationships applied to the combination of subject and object variables. Each record indicates that a subject with a lower or equal clearance can write to an object with a higher or equal classification. The following three records support the hypotheses discussed in Examples 2, 3, and 7:

- *_Subject_4 with _SecurityLabel_TS_Bio canWrite _Object_7 with _SecurityLabel_TS_BioNuke.*
- *_Subject_6 (hasSecurityLabel _SecurityLabel_S_Null) canWrite _object_4 (hasSecurityLabel __SecurityLabel_TS_Null)*
- *_Subject_3 (hasSecurityLabel _SecurityLabel_S_Bio) canWrite _object_7 (hasSecurityLabel __SecurityLabel_TS_BioNuke)*

4.4 Additional Notes for Implementation

Unlike other query languages in Protégé, SWRL queries only extract information from assertional knowledge (relationships between individuals). It is crucial to ensure that actual assertions are made for each individual. In OWL, it is not incorrect to leave object property assertions blank, but the inference engine cannot generate any inferred assertions 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 a sensitivity level and compartment. If L1 does not have a clear classification of its compartment

C1, even if it has a compartment instance *Bio* on the terminological side, the rule's two conditions, hasCompartment(?L1, ?C1) and hasSubset(?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)
```

5 Future Work and Conclusion

This research has successfully demonstrated the feasibility and effectiveness of implementing Multi-Level Security (MLS) policies using semantic web technologies, specifically through the application of the Web Ontology Language (OWL). By integrating the principles of Mandatory Access Control (MAC) and the Bell-LaPadula (BLP) model within a semantic framework, we have established a robust system capable of enforcing stringent security protocols that ensure the flow of information is strictly from lower to higher classified entities.

Throughout this project, the construction and validation of an MLS ontology were explored, highlighting the capability of semantic technologies to enhance the traditional security measures. The use of OWL facilitated a detailed representation of security levels, dominance rules, and access control mechanisms which were dynamically applied through the Semantic Web Rule Language (SWRL). This approach not only supported the enforcement of complex security policies but also contributed to the flexibility and scalability of the security system, accommodating various security scenarios in a highly controlled environment.

Furthermore, the project underscored the importance of adapting security measures to accommodate the evolving nature of the web and its technologies. As the digital landscape becomes increasingly complex, the integration of semantic web technologies with traditional security frameworks presents a promising avenue for enhancing data security and integrity across various domains.

The implications of this research are significant for sectors that require high levels of information security, such as military, government, and healthcare. The methodology and findings discussed provide a foundation for future studies to explore and expand upon, particularly in areas involving the integration of more advanced semantic tools and the exploration of new security scenarios that may arise from the ever-evolving web technologies.

In the near future, the development of Multi-Level Security (MLS) within a semantic web framework is poised for significant advancements. As semantic technologies evolve, integrating MLS more deeply into these frameworks will allow for more nuanced and context-aware security policies that dynamically adjust based on real-time data interpretation and threat analysis. Anticipated innovations include the use of advanced artificial intelligence algorithms to enhance decision-making processes within MLS systems, enabling them to automatically adapt security levels and access controls based on changing environmental cues and user behaviors. Additionally, there is potential for leveraging semantic relationships and ontologies to develop more sophisticated data access protocols that can understand and react to complex user queries with enhanced security precision. This integration will not only streamline security management but also increase its efficacy, ensuring that sensitive information remains protected as it becomes increasingly interconnected across diverse and expansive digital ecosystems.

References

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. *21st Annual Computer Security Applications Conference (ACSAC'05)*, 15 pp. – 351. https://doi.org/10.1109/CSAC.2005.37

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*.

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

Editor, C. C. (n.d.). *mandatory access control (MAC)—Glossary | CSRC*. Retrieved July 3, 2024, from https://csrc.nist.gov/glossary/term/mandatory_access_control

Elliott Bell, D. (2011). Bell–La Padula Model. In H. C. A. van Tilborg & S. Jajodia (Eds.), *Encyclopedia of Cryptography and Security* (pp. 74–79). Springer US. https://doi.org/10.1007/978-1-4419-5906-5_811

Hyun, J. (2020). MULTI-LEVEL SECURITY (MLS) POLICY IMPLEMENTATION USING GRAPH DATABASE. *Electronic Theses, Projects, and Dissertations*. https://scholarworks.lib.csusb.edu/etd/1126

Kayes, A. S. M., Rahayu, W., & Dillon, T. (2018). An ontology-based approach to dynamic contextual role for pervasive access control. *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 601–608.

Kiran, G. M., & Nalini, N. (2022). Ontology-based data access control model supported with grid computing for improving security in healthcare data. *Transactions on Emerging Telecommunications Technologies*. https://doi.org/10.1002/ett.4589

Li, J., & Zhang, B. (2019). An ontology-based approach to improve access policy administration of attribute-based access control. *International Journal of Information and Computer Security*, *11*(4–5), 391–412. https://doi.org/10.1504/IJICS.2019.101928

Logrippo, L. (2018). Multi-level Access Control, Directed Graphs and Partial Orders in Flow Control for Data Secrecy and Privacy. In A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, & J. Garcia-Alfaro (Eds.), *Foundations and Practice of Security* (pp. 111–123). Springer International Publishing.

Matsokis, A., & Kiritsis, D. (2011). Ontology applications in PLM. *International Journal of Product Lifecycle Management*, *5*(1), 84–97. https://doi.org/10.1504/IJPLM.2011.038104

O'Connor, M., Knublauch, H., Tu, S., & Musen, M. (2005). *Writing Rules for the Semantic Web Using SWRL and Jess*. https://www.semanticscholar.org/paper/Writing-Rules-for-the-Semantic-Web-Using-SWRL-and-O%27Connor-Knublauch/4e1defebddc2e017767ba1768cb9694de5a72fe1

Panossian, G. (2019). MULTI-LEVEL SECURE DATA DISSEMINATION. *Electronic Theses, Projects, and Dissertations*. https://scholarworks.lib.csusb.edu/etd/946

Roy, S., Dayan, G. S., & Devaraja Holla, V. (2018). Modeling Industrial Business Processes for Querying and Retrieving Using OWL+SWRL. In H. Panetto, C. Debruyne, H. A. Proper,

C. A. Ardagna, D. Roman, & R. Meersman (Eds.), *On the Move to Meaningful Internet Systems. OTM 2018 Conferences* (Vol. 11230, pp. 516–536). Springer International Publishing. https://doi.org/10.1007/978-3-030-02671-4_31

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, December 11). Marriott Data Breach Is Traced to Chinese Hackers as U.S. Readies Crackdown on Beijing. *The New York Times*. https://www.nytimes.com/2018/12/11/us/politics/trump-china-trade.html

Son, J., & Alves-Foss, J. (2014). Security/Performance Tradeoffs in Hybrid Real-Time Scheduling Algorithms. *Pan Pacific Journal of Business Research*, *5*(1), 2–25.

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

Veloudis, S., Paraskakis, I., Petsos, C., Verginadis, Y., Patiniotakis, I., Gouvas, P., & Mentzas, G. (2019). Achieving security-by-design through ontology-driven attribute-based access control in cloud environments. *Future Generation Computer Systems*, *93*, 373–391. https://doi.org/10.1016/j.future.2018.08.042