Distributed Multilevel Secure Data Access Using SELinux-Enhanced Linux over Cloud Environments

Haklin Kimm*, Eun-Joo Lee, and Hanke Kimm

***Abstract*—Cloud computing is an emerging technology, which provides a spectrum of services such as database, networks, software, storage over the cloud to many organizations on the internet; there is no need of keeping local servers for the organizations that adapt cloud computing for storing and accessing data and services, provided by remote servers. However, there are organization still hesitant to accommodate a cloud system due to security issues and risks. In this work, Security-Enhanced Linux (SELinux) with MultiLevel Security (MLS) framework is applied as a cloud security model; SELinux runs based on sensitivity and category labels of subject and objects − mostly users and data files respectively. The proposed MLS data access is a framework that employs data classification that works on subsequent changes in the sensitivity levels of subjects and objects to meet the dynamic changes of cloud security. To implement the proposed secure data access framework over the cloud system, the Fedora 35 system with SELinux is used as a testbed. We also present that data modification history can be retrieved on a MLS system by using user sensitivity and data information labels.**

***Index Terms*—Security-enhanced Linux, multilevel security, data access, cloud computing**

I. Introduction

Cloud computing is an emerging technology, which provides a spectrum of services such as database, networks, software, storage, over the cloud to many organizations on the Internet, and has evolved along with computer networks and information technologies, distributed file systems, cluster applications. With cloud computing, there is no need of keeping local servers for the organizations that adapt cloud computing for storing and accessing data and services, provided by remote servers. A large number of various types of storage devices are well collected and organized in cloud computing whose service products are provided by Google Cloud [1, 2], Amazon Web Services [3, 4], Microsoft Azure [5], iCloud [6], etc. Cloud computing is defined by the National Institute of Standard and Technology (NIST) such as: a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [7,8].

The essential aspects of cloud computing are on demand self-service, broad network access, measured service, rapid elasticity, and resource pooling. Cloud service offerings divide into three broad categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Note that there has been a recent proliferation of other "as-a-Service" assortments such as Business Integration-as-a-Service, Cloud-Based Analytics-as-a-Service (CLAaaS), Data-as-a-Service (DaaS) [9, 10]. Cloud computing provides many advantages to many organizations; however, there are some organizations still reluctant to accommodate their work on the cloud due to security issues and risks.

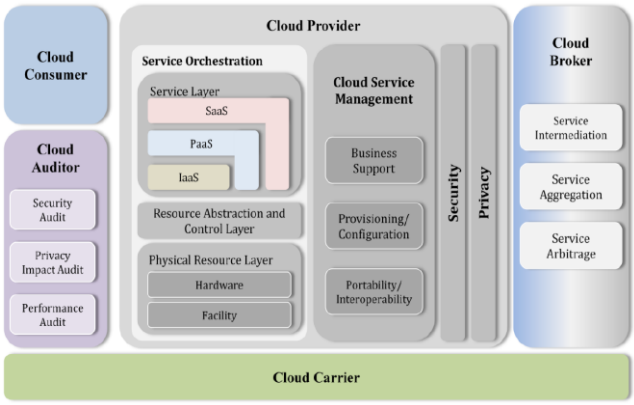

Fig. 1. NIST cloud computing model

Thus, it is important to address the security issues and problems in cloud systems, and the security requirements for data and services on cloud systems are getting more attention nowadays [11].

Security is the main huddle to accommodating a cloud system as stated in Cloud Security Alliance's survey report [12]. Other hindrances from encompassing cloud services are the loss of control over IT services, concern over compromised accounts, insider threats, business continuity, and disaster recovery [13]. In cloud systems, data owner does not have full authority over their own data stored on remote servers since internal and external users, who might misuse their access to harm or disclosure the valuable sensitive data, access and share the data stored on cloud storage. Hence, the data owners seek to develop or deploy strict security in order to protect their sensitive data. These security challenges on cloud systems focused through a robust security management initiative that outlines the role and responsibility of the service provider and the data owner [14-17].

Sensitive information on cloud platforms are required to extend with technical measures and organizational policies to avoid security breaches that might result in enormous damages. Sensitive information in the context of cloud computing encompasses data from a wide range of different areas and disciplines. Cloud computing platforms that support the deployment of security policies to data, owners,

Manuscript received November 20, 2022; revised February 2, 2023; accepted March 29, 2023.

Haklin Kimm and Eun-Joo Lee are with Department of Computer Science, East Stroudsburg University, PA, 18301, USA

Hanke Kimm is with Department of Computer Science, Stony Brook University, Stony Brook, NY, 11794, USA

*Correspondence: hkimm@esu.edu

doi: 10.18178/jacn.2023.11.1.285

and users have been evolving recently, and these security policies apply to control when user data flows at runtime. In this paper we contribute to a multilevel security framework based on data sensitivity and security–providing adequate level of data security with various classifications and categories. The proposed multilevel security embedded information retrieval tool encompasses suitable access control, which combine with Security Enhanced Linux (SELinux) that facilitates classification of data based on subsequent changes in the sensitivity levels and the security measures meeting the dynamic changes in cloud security threats [16-20].

The rest of this paper is organized as follows. Section II demonstrates some previous and related work about cloud computing and security. Section III illustrates our distributed MLS model in detail along with SELiux. Section IV introduces the main idea and details of our proposed MLS information retrieval framework that follows the experimental details using SELinux over Fedora Linux. Finally, we conclude with further research directions in Section V.

II. Background and Related Work

Cloud Computing is defined as a model for facilitating ubiquitous, convenient, pay-as-go access to a shared pool of configurable computing resources, providing five major actors as shown in Fig. 1: Cloud consumers, cloud providers, cloud carriers, cloud auditors and cloud brokers. Each of these actors is an entity (either a person or an organization) that participates in a cloud computing transaction or process, and/or performs cloud computing tasks.

A. Cloud Computing and Security

Cloud computing model is a layered architecture providing different services with levels. As seen in Fig. 1 and Table I, each layer of the cloud computing is classified with its own security problems based on a shared cloud responsibility model mostly processing as cloud consumers and providers. The cloud security risk and problems, shared by cloud customers and providers, are delineated as IasS, PasS, and SasS. It notices that SaaS customers are responsible for securing their data and user access; PaaS customers for securing their data, user access, and applications; IaaS customers for securing data, user access, application, operating systems, and virtual network traffic [27].

TABLE I: Cloud Security Classification by McAfee

Shared Responsibility Model for Security in the Cloud			
On-Premises (for reference)	IaaS (infrastructure-as-a-service)	PaaS (platform-as-a-service)	SaaS (software-as-a-service)
User Access	User Access	User Access	User Access
Data	Data	Data	Data
Applications	Applications	Applications	Applications
Operating System	Operating System	Operating System	Operating System
Network Traffic	Network Traffic	Network Traffic	Network Traffic
Hypervisor	Hypervisor	Hypervisor	Hypervisor
Infrastructure	Infrastructure	Infrastructure	Infrastructure
Physical	Physical	Physical	Physical
Customer Responsibility		Cloud Provider Responsibility	

Infrastructure as a Service (IaaS) Clouds: IaaS provides a hardware platform as a service such as virtual machines, processing power, storage, networks, and database services as shown in Table I. IaaS was the first widely available commercial cloud type, initiated by Amazon's launch of their EC2 service [13], and made possible by the widespread availability of efficient open-source hardware virtualization such as Google Cloud, and Microsoft Azure [22, 23]. IaaS resources are usually provided to cloud consumers over the internet as virtualized resources such as virtual storage, virtual network, virtual machine (VM). Hence, IaaS is more flexible and scalable than on physical dedicated hardware. This platform virtualization concerns on security on hardware and software virtualizations.

Platform as a Service (PaaS) Clouds: PaaS customers develop their applications upon IaaS using languages and service APIs provided by the cloud provider without buying and maintaining hardware, software and hosting severs. PaaS providers have their own specialties such as Microsoft Azure PaaS, AWS Lambda, Google App Engine, Apache Stratos, etc. PaaS customers can deploy cloud resources upon their need and capability, utilizing infrastructure scalability of IaaS. Securing PaaS infrastructure is not easy when considering that the PaaS services are mostly heterogeneous, and varying in their security offerings, which usually share with multiple users simultaneously.

Software as a Service (SaaS) Clouds: SaaS is an on-demand cloud model that allows cloud users to subscribe to the application and/or data hosted by the cloud provider upon their need and capability. The data on SaaS manipulate and remain within the cloud, and there is no need for in-house hardware and on-site IT staff to maintain and support the application. Unlike IaaS or PaaS offerings, SaaS users need little technical knowledge. Individual users are unlikely to distinguish SaaS from other types of web-based service.

B. Multilevel Security

The proposed approaches to cloud security are identifying sensitive data, accessing sensitive data and sharing, uncovering misuse of resources including malicious behaviors of cloud users and providers [27]. Any malicious insider may intentionally or accidently misuses the sensitive data in the cloud system even though there are efforts trying to protect the sensitive data stored in their servers. Cloud providers and consumers are in an imperative need to have their own security schemes to protect their valuable data in the cloud. Hence, a MLS management framework for the valuable and sensitive data is suggested, which is more flexible and reliable allowing data owners to choose the appropriate schemes for different data in different classes along with standard authentication and access control mechanisms. Multilevel security or multiple levels of security is to classify data and protect the classified data by providing security at various levels, starting with access permission based on security clearance [21].

Multilevel Security (MLS) policy is an implementation of Mandatary Access Control (MAC), which focuses on confidentiality. A Multilevel Security operating system can enforce the separation of multiple classes of information and manage multiple users with varying levels of information clearance. MLS is based on a formal model called the Bell-LaPadula model designed by David E Bell and Leonard Padula [24, 28]. MLS is designed to control the flow of information between subjects. Subjects and objects in a system are given with hierarchical sensitivity labels, and the MLS policy controls the flow of information between the designated sensitivities [25, 26].

SELinux, maintained and recommended by National Security Agency (NSA), seeks to specify fully the principle

of least privilege on modern operating systems using a Mandatory Access Control (MAC) security policy. SELinux provides several mechanisms to protect against attacks and exploitation of software vulnerabilities. SELinux carries out role-based access control and sandboxing or combination of role-based access control, type enforcement, and MLS models [29-31].

III. MLS Information Retrieval

TABLE II: Global Data Dictionary for Secure Retrieval

Metadatabase with security and category labels
jan ndp001r3:january:r:7.2:/tmp/stc/2021/budget/:s2:c1
feb ndp001r3:february:r:7.2:/tmp/stc/2021/budget/:s4:c1
mar ndp001r3:march:r:7.2:/tmp/stc/2021/budget/:s2:c1
apr ndp001r3:april:r:7.2:/tmp/stc/2021/budget/:s0:c3
may ndp001r3:may:r:7.2:/tmp/stc/2021/budget/:s0:c3
jun ndp001r3:june:r:7.2:/tmp/stc/2021/budget/:s4:c3
jul ndp001r3:july:r:7.2:/tmp/stc/2021/budget/:s0:c4
aug ndp001r3:august:r:7.2:/tmp/stc/2021/budget/:s2:c4
sep ndp001r3:september:r:7.2:/tmp/stc/2021/budget/:s4:c4
oct ndp001r3:october:r:7.2:/tmp/stc/2021/budget/:s4:c4
nov ndp001r3:november:r:7.2:/tmp/stc/2021/budget/:s2:c5
dec ndp001r3:december:r:7.2:/tmp/stc/2021/budget/:s4:c4

In our framework design, SELinux system is used as a testbed to retrieve information securely. At any time, SELinux users or processes are allowed to read data with lower-or-equal sensitivity levels to their current process sensitivity level. In a SELinux system, all user interactions are initiated through the default command-driven terminals. SELinux users are allowed to perform multiple tasks, and to operate at several different sensitivity levels in a single login session [31]. A global data dictionary as shown in Table II, which we call a metadatabase is provided in order to provide name and location transparency to our information retrieval tool. The users do not need to provide information about their own location, file name or file types with security to retrieve information from their local sites because name and location transparency are being kept by the metadatabase as shown in Table II above.

A. MLS Information Platform

Here we present a MLS information retrieval tool that runs user queries for retrieving and integrating data. Since our retrieval tool uses the SELinux system, the security level of a global query is the same as the security level of the current user's MLS security context − *user:role:type:mls/mcs* such as *staff_u:system_r:unconfined_t:s0-s5:c1.c5*. Data can be retrieved from the distributed secure database system shown in Table II, satisfying up to a query's security level [32] from the current user. Each individual data of *jan, feb, … dec* has been saved under */tmp/stc/2021/budget/* folder, which can be replaced with actual hostname and its corresponding file folder when the proposed MLS information platform is implemented on a real cloud system. For example, the global query with a confidential sensitivity label, which is the sensitivity label of the current terminal window or user, may not retrieve the data with a Secret label even if the user logins with Top Secret label if the category context of the user and the data are not matched; the user is equipped with category c1 and the data is with category c2. The proposed SELinux Information Retrieval (SELIR) algorithm evaluates the current user's sensitivity levels and category contexts first, comparing with the current sensitivity levels and category context of each data in the global data dictionary: *jan, feb, … dec*. Then the user types in data they like to retrieve not knowing the location, security level, and category context of data. All the allowed data for the user to access are displayed on the screen, satisfying security level and category context. The proposed SELIR algorithm is shown in the following.

Algorithm SELIR
Input: user request for information retrieval.
Output: reply to user request.

Method:

1. *Global query access the global data dictionary (Metadatabase and Metathesaurus) to get the corresponding local objects, where* $query_{security\text{-}level} >= Object_{security\_level}$ *(i.e., minimum security level of all the corresponding objects)*
2. *Generate subquries and send them to each corresponding database if step 1 is satisfied.*
3. *At each local database:*
   a. *Retrieves the records (tuples) from the corresponding table.*
   b. *Send the tuples back to the user.*
4. *At the user query location:*
   a. *If all the requests are received from the corresponding databases,*
      merge all the records received into a results file.
      else:
      generate error messages and repeat the query.
   b. *The merged results file (snapshot file) is processed to create an answer based on the user request.*
   c. *Display the answer to the user.*
   d. *If there is a request for data history, go to step b again.*
5. *Go to step 1, if there is a second global query.*

SELinux labels (information label and sensitivity label) are used to implement the mandatory access control (MAC) policy; hence, the steps 1 and 2 of SELIR algorithm are performed with ease. Therefore, in our SELIR algorithm, we do not need to check or compare security levels of subjects and/or objects to retrieve information manually.

B. Tracking Information Modification History

The following section shows the correctness of using information labels of a MLS system to track the data modification history. There are two security labels which are related to each subject and object in SELinux: a sensitivity label (SL) and an information label (IL). MLS Security labels are represented by the SL and IL together. A sensitivity label (SL) indicates the security level with which an entity such as a file is protected by the mandatory access control (MAC) policy. An information label (IL) indicates the security level of the data actually contained within a subject or object, an entity such as a file. The IL gives an indication of how the data should be handled. The IL can also help users decide whether to downgrade the SL of a file. A user might choose to downgrade the SL of a file if the IL is below the SL of the file that contains the data. An SL consists of a classification and a set of compartments (categories). An IL also consists of a classification and a set of compartments, plus a set of markings.

As mentioned in [22], the encodings of security labels control the translation between the human-readable and internal formats of information labels, sensitivity labels, and clearance labels. Human-readable labels consist of a classification followed by a set of words. The words can represent compartments: in information labels, sensitivity label, clearances, and markings: information labels only. The word "compartments" will be used throughout this paper for

consistency with other intelligence community documentation, but conceptually means the same as "categories." When two pieces of data with separate information labels are merged or combined, the system of [9] automatically determines the combination of the two information labels, setting the single information label that properly represents the merged data. This process of adjudicating two information labels is also called combining the labels or floating one label with the second one. The values assigned to classifications and the internal category, and marking bit representations assigned to information label words determine how the MLS system will adjudicate information labels [22].

TABLE III: SAMPLE INFORMATION LABELS FOR ENCODINGS FILE

CLASSIFICATIONS:
name= NOT; sname= N; value= 1;
name= LOW; sname= L; value= 4;
initial compartments= 4-5 100-127;
initial markings= 11 12 17 100-127;
name= MEDIUM; sname= M; value=5;
initial compartments= 4-5 100-127;
initial markings= 11 12 17 100-127;
name= HIGH; sname= H; value =6;
initial compartments= 4-5 100-127;
initial markings= 11 12 17 100-127;
INFORMATION LABELS:
name= GSTU; minclass= L;compartments= 1 3;markings= 7;
name= STU;minclass= L;compartments= 1; markings= 7;
name= PSTA;minclass= L;compartments= 0 2; markings= 7;
name= STA; minclass= L; compartments= 0; markings= 7;
name= D/E; minclass= L; markings= 16;
name= NOFORN; minclass= L; compartments= 4-5; markings= 11 13;
SENSITIVITY LABELS;
name= GSTU; minclass= L; compartments= 0;
name= STU; minclass= L; compartments= 1;
name= PSTA; minclass= L; compartments= 2;
name= STA; minclass= L; compartments= 3-5;

In SELinux systems, information labels track the flow of information from objects, through subjects, and onto other objects. An objects's IL reveals the current sensitivity level of the information of the object, whereas an object's SL represents the maximum limit of sensitivity level of information that can ever be written onto the object. In the same way, a subject's IL represents the current sensitivity level of the information in the subject's address space, whereas a subject's SL represents the maximum limit of the sensitivity level of information that can ever be read by the subject. The IL of a subject or object shows the *high-water mark* of the sensitivity of information processed by them, respectively. Whenever information flows from one entity to another, the information label of a subject or object will be updated, which is called *IL floating*. The IL floating is done by using the operation *conjoin* which is consisted of two operations:

(i) $Classification_{New} = MAX(classification_{IL1}, classification_{IL2})$

(ii) $Other\_bits_{New} = Bitwise\_OR\ (Other\_bits_{IL1}, Other\_bit_{IL2})$

Assume that IL1 is the IL of the object and IL2 is the original IL of the subject. When a subject reads information from an object, the IL of the subject is floated to the conjoin of IL of the object and the original IL of the subject; the IL of the object is not changed. When a subject writes information onto an object, then the IL of the object is raised to the conjoin of its original IL and the IL of the subject. The IL of the subject is not updated. This updating of the IL can be written:

$(IL_{subject}, IL_{object})$ ---read---> $(Updated\_IL_{subject}, IL_{object})$

$(IL_{subject}, IL_{object})$ ---write--> $(IL_{subject}, Updated\_IL_{object})$

To retrieve the previous IL of the current object, the IL of the current subject and the IL of the current object are required. The backtracking of the previous IL will be done by using the operation de_conjoin which is also consisted of two operations. In this paper the de_conjoin operations are defined as follows:

(i) $Classification_{Old} = MIN(classification_{IL1}, classification_{IL2})$

(ii) $Other\_bits_{Old} = Bitwise\_XOR\ (Other\_bits_{IL1}, Other\_bit_{IL2})$

The bitwise exclusive or (XOR) operation is performed to the ILs of the current object and subject to retrieve the previous IL of the current entity. The example of bitwise OR and XOR operations for the compartment and marking bits is shown as follows:

Subject	*1 0 1 0*	*New*	*1 1 1 0*
Object(OR)	*0 1 0 0*	*Object(XOR)*	*0 1 0 0*
New	*1 1 1 0*	*Subject*	*1 0 1 0*

The new IL will be the IL of the current subject if the read operation has done, which will be IL of the current object if the write operation has done. As shown above, the bitwise exclusive or (XOR) is able to retrieve the previous compartment and marking bits of the IL of the current entity. This bitwise XOR is used to retrieve the previous ILs when the compartment and marking bits are defined as non-hierarchical words. However, when the compartment and marking bits are defined as hierarchical words, there needs more attention.

The example of conjoin and de_conjoin operations is as follows: A user with an information label (Low Stu) copies an object with an information label (Low Sta), where the classification value of Low is 4, and the compartment bits of Stu is 0010 and Sta is 1100. What will be an information label of the newly copied object? The conjoin operation of the given information labels will be done to get an IL of the copied object as follows:

(i) $Classification_{New\_obj} = MAX(classification_{sub} = 4, classification_{obj} = 4) = 4$

(ii) $Other\_bits_{New\_obj} = Bitwise\_OR\ (Other\_bits_{sub} = 0010, Other\_bit_{obj} = 1100) = 1110,$
where classification 4 = Low, and Compartment 1110 = Stu Sta

The information label of the copied object by a user with an information label (Low Stu) will be (Low Stu Sta). If the user wants to know the history of the copied data such as source of the data copied, then the de_conjoin operation will be performed to retrieve the previous information of copied data.

(i) $Classification_{Old\_obj} = MIN(classification_{sub} = 4, classification_{obj} = 4) = 4$

(ii) $Other\_bits_{Old\_obj} = Bitwise\_XOR\ (Other\_bits_{sub} = 0010, Other\_bit_{obj} = 1110) = 1100,$
where classification 4 = Low, and Compartment 1100 = Sta

Now we know that the previous object has an information label (Low Sta). The previous object (file) also will be retrieved from the audit file of MLS system by using the IL (Low Sta).

A further study on the method of getting a previous object's information labels needs to be implemented on SELinux.

IV. Information Retrieval on SELinux

A. MLS SELinux Using Fedora Linux

In order to implement the proposed framework, shell scripts have been implemented on a computer running Fedora 35 enforcing SELinux MLS. However, at this time of developing this framework Korn Shell Script (ksh) has been used – not using a Linux default Born Again Shell (bash); this *ksh* script demonstrates how user and resource labels are created to enforce access control on resources on varying MLS labels and categories. These shell scripts have been implemented after setting up the */etc/selinux/config* file with the options of *permissive* and *mls*. In order to implement the SELIR script properly, temporary text files were created for storing information about user sensitivity labels and category labels. These files allow the user to access and check if each user has the proper right to access, and retrieve the sample data containing a month of the corresponding budget amount from the secure text files shown in Table II, which are stored under /tmp/stc/2021/budget. For example, the user with sensitivity level with **=<** ***s2*** and category **=** ***c1*** as seen in Table III is not allowed to access *january* data in Table II, which carries the information label of *s2:c2* since the categories are not matched. The following is strongly recommended for setting up SELinux MLS on Fedora 35 to implement the proposed MLS framework.

```
# Set up root user
    sudo –i
    passwd
    logout (cntl + d)
#dnf install the following packages
    selinux-policy-targeted, selinux-policy, selinux-policy-mls,
    libselinux, libselinux-utils, policycoreutils, libselinux-python,
    policycoreutils-python, setroubleshoot,
    setroubleshoot-server, setroubleshoot-plugins
#Add SELinux users
    Useradd –Z staff_u nina
    Passwd nina
    Chcon –R –l s2:c3.c5 /home/nina

#Uncomment the last 3 lines of /etc/security/namespace.conf
```

This MLS security framework is vital because the layered nature of cloud computing services is scattered across different storage devices or servers that build the proper security policy and category of an organization upon the user security as well as the sensitivity of data or information. By implementing MLS with sensitive and security levels, a business may also insulate themselves from any malicious insider's intent on harming data or stealing it due to lack of clearance and log monitor. At this time, the developed scripts only determine whether the user or process can access the data based on the security labels and category contexts on the subject and objects, and it would not be hard to scale up its application towards services, applications, ports and files. The following *ksh* script shows how to extract user labels from user-id and information labels from *metadb* of Table II. Comparing user labels with information labels to allow or disallow monthly corresponding budget files are shown in the following.

```
# Retrieve Sensitivity and Category labels from user ID
  id -Z > idZ
  cut -f4 -d":" idZ > idRange
  cut -f5 -d":" idZ > idCategory
  rag1=$(cut -f1 -d"." idRange| tr -dc '0-9')
  rag2=$(cut -f2 -d"." idRange| tr -dc '0-9')
  cat1=$(cut -f1 -d"." idCategory| tr -dc '0-9')
  cat2=$(cut -f2 -d"." idCategory| tr -dc '0-9')
# Retrieve information labels from metadb
      grep $i metadb > temp
      dbRag=$(cut -f6 -d":" temp | tr -dc '0-9')
      dbCat=$(cut -f7 -d":" temp | tr -dc '0-9')
      echo $dbCat $dbRag
# Compare category labels and sensitivity labels of subject (user)
  and object (data to access)
    if [[ ($dbCat -ge $cat1 && $dbCat -le $cat2) && \
        ($dbRag -ge $rag1 && $dbRag -le $rag2) ]]     #
```

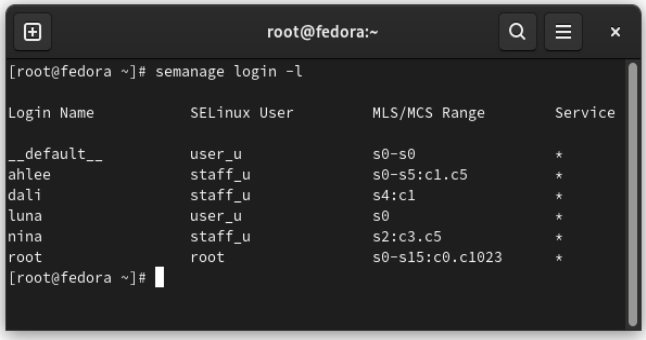

Fig. 2. SELinux users with sensitivity and category labels

B. SELIR Implementation on SELinux

SELinux system enforces a *Write Up/Read Down* security policy in which unprivileged subjects can write only to objects at a higher or equal level and read only from objects at a lower or equal level. A subject can read from an object only if the sensitivity label of the subject dominates the sensitivity label of the object (Read Down). A subject can write to an object only if the sensitivity label of the object (Write Up) dominates the sensitivity label of the subject.

The SELIR tool works fine and well if the user with its sensitivity label is higher or equal to that of the sensitivity labels of objects to retrieve. If not, SELIR tool will not be able to retrieve the objects that the user requests. The information retrieval tool only retrieves the objects with a lower-or-equal sensitivity label than that of the subject. When the shell script runs, it prompts the user to select one of the following three options: *numerical information, textual information, and exit*. As seen in Fig. 2, *the user ahlee is staff_u with s0-s5:c1.c5; luna is user_u with s0, and dali is staff_u with s4:c1, nina is assigned to staff-u with s2:c3.c5*. All the security labels and category levels of each file to access are displayed on Table II.

As seen in Fig. 3, the menu started with selecting numerical or textual information. When selecting *numerical information*, it displays the sensitivity and category levels of the current user. In this example, the user *ahlee* has clearance or sensitivity/security levels from *s0* to *s5*, and category

contexts of *c1* through *c5*. Now she can access all the files: *jan, mar, sep,…, dec* in Table II that are being saved with sensitivity levels *s0−s4* and category contexts of *c1-c5*. However, she accesses *jan, mar, sep, oct, nov, dec* at this time without any restrictions since these files are stored with a sensitivity level from s0 to s4 and categories from c1 to c5. The sensitivity level and category context of each file are displayed and compared with *ahlee*'s labels as seen in Fig. 3.

```
1. Look for numerical information
2. Look for textual information
3. Exit

Please select one of the above (1-3): 1
   user sensitivity: 0 5 user category: 1 5
Enter attributes to look up: jan mar sep oct nov dec
   data category: 1 data sensitivity:2
   /tmp/stc/2021/budget/january
   data category: 1 data sensitivity:2
   /tmp/stc/2021/budget/march
   data category: 4 data sensitivity:4
   /tmp/stc/2021/budget/september
   data category: 4 data sensitivity:4
   /tmp/stc/2021/budget/october
   data category: 5 data sensitivity:2
   /tmp/stc/2021/budget/november
   data category: 4 data sensitivity:4
   /tmp/stc/2021/budget/december

   1. Look for data processing
   2. Look for data tracking
   3. Exit

   Please select one of the above (1-3): 1
      1. List data
      2. Statistics of data
      3. Exit
   Please select one of the above (1-3): 1
   total budget is 5500000
```

Fig. 3. SELIR Implementation based on user ahlee

This example is to demonstrate that users can be isolated based upon resources they may access; and that even though they may maintain high levels of security, they still should meet the category context of the objects. If not, the users with higher-or-equal security labels cannot access all the data not matching the category contexts of the users and data even below sensitivity levels of the users. It demonstrates that even though two users may have access to the same security level, they may not have access to the same contextual/categorical clearance, providing a means of separating higher critical information from the lower within the same level of clearance. As seen in Fig. 4, the user *nina* is assigned to *staff-u with s2:c3.c5*. Hence, *nina* is not allowed to access the file *jun* with s3:c4. Nina's category matches with *jun* but *nina*'s sensitivity level (s2) is lower than the file *jun's* (s3). In the same way, *nina* cannot access the files, *oct* and *dec*, which have the same sensitivity level (s4) but category context (c4). However, the *root* user who may access all the resources without restrictions since the *root* user's sensitivity level is from s0 to s5 and category label is from c0 to c1023. If a user has the proper authorization, then the authorized information of *jan ... dec* are displayed for the user, otherwise the user is alerted for that particular piece of information the user was restricted either based upon the mismatch of their category context or sensitivity level to data trying to access. This effectively demonstrates how resources in a cloud system could properly label MCS/MLS and restrict role or user specification to ensure that data integrity is preserved.

```
Please select one of the above (1-3): 1
   user sensitivity: 2 2 user category: 3 5
1
Enter attributes to look up: apr may jun aug oct nov dec

   data category: 3 data sensitivity:0
   /tmp/stc/2021/budget/april
   data category: 3 data sensitivity:0
   /tmp/stc/2021/budget/may

   data category: 3 data sensitivity:4  #user sensitivity level is 2

   data category: 4 data sensitivity:2
   /tmp/stc/2021/budget/august

   data category: 4 data sensitivity:4  #user sensitivity level is 2

   data category: 5 data sensitivity:2
   /tmp/stc/2021/budget/november

   data category: 4 data sensitivity:4  #user sensitivity level is 2

   1100000     #april
   800000      #may
   500000      #august
   400000      #november
   total budget is 2800000
```

Fig. 4. SELIR Implementation based on user nina

An example of the main menu and submenus of the SELIR tool is shown above briefly, executing the SELIR tool by selecting options from each menu. This SELIR tool is a menu driven system where users are only required to select the options from the menu on their screen. This menu driven system is easier to use than a command line system since users do not need to know the commands as well as their sensitivity levels and category contexts.

V. Conclusion

In this paper, we have represented a MLS framework for information retrieval and data tracking in a Security Enhanced Linux (SELinux) system using Fedora 35. Information labels are used to track the flow of information in SELinux systems as shown in section 3. As we mentioned in the previous sections, information labels are applied to retrieve the previous information label of the current data by using the information labels of the current subject and object. The previous immediate data or file can be retrieved by accessing the audit file of a SELinux system by using the information label just found. As shown in our previous examples, the information label will be a good medium to track or monitor activities of objects or files, even in heterogeneous distributed multilevel secure environments. We note that modification and upgrade of the proposed Korn shell program are recommended to satisfy the secure use of SELIR tool for more practical heterogeneous MLS cloud systems. The SELIR on SELinux system with MLS worked well using Korn Shell in this project. However, we could not fully implement our proposed SELIR tool yet on cloud environments. We expect to work more implementing SELIR

tool on real cloud systems in near future.

REFERENCES

[1] Google App Engine. Accessed: March 30, 2021. [Online]. Available: http://appengine.google.com/

[2] Google Colaboratory, Accessed: March 5, 2020, [Online]. Available: colab.research.google.com

[3] Amazon. AWS Security, "Amazon web services: Overview of security process," AWS SecurityWhite Paper, 2013.

[4] Amazon. Amazon S3. Accessed: March 30, 2021. [Online]. Available: http://aws.amazon.com/s3/

[5] Windows Azure. Accessed: April 5, 2021. [Online]. Available: http://www.microsoft.com/windowsazure/

[6] Apple iCloud. Accessed: April 5, 2021. [Online]. Available: http://www.icloud.com/

[7] F. Liu, *et al.*, "The NIST cloud computing reference architecture," National Institute of Standard and Technology, NIST SP 500-292, US Department of Commerce, Gaithersburg, Md, USA, 2011.

[8] C. Lee, R. Bohn, and M. Michel "The NIST cloud federation reference architecture," National Institute of Standard and Technology, NIST SP 500-XXXX, US Department of Commerce, Gaithersburg, Md, USA, 2011.

[9] A. Gholami and E. Laure, "Security and privacy of sensitive data in cloud computing: A survey of recent developments," NETCOM, NCS, WiMoNe, CSEIT, SPM-2015, pp. 131–150, 2015.

[10] S. Sharma, "Evolution of as-a-Server era in Cloud," 2015, arXiv:1507.00939. [Online]. Available: https://arxiv.org/abs/1507.00939

[11] S. Subashini and V. Kavitha, "A survey on security issues in service deliverymodels of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[12] C. Coles and J. Yeoh, "Cloud adoption practices & priorities survey report," *Tech. Rep*., *Cloud Security Alliance*, 2015.

[13] S. Dorairaj and R. Kaliannan, "An adaptive multilevel security framework for the data stored in cloud environment," *The Scientific World Journal*, vol. 2015, Article ID 601017.

[14] W. Zhang, "Analyzing the overhead of FileSystem protection using Linux security modules," in *Proc. Conference'17*, July 2017, Washington, DC, USA.

[15] S. Sultan, *et al*., "Container security: Issues, challenges and the road ahead,"DOI 10.1109/ IEEE ACCESS.2019.2911732, May 1, 2019.

[16] B. Im, *et al.*, "An historical analysis of the SEAndroid policy evolution," arXiv:1812.00920. [Online]: https://arxiv.org/abs/1812.00920

[17] Y. Zhao, *et al.*, "Secure and efficient product information retrieval in cloud computing," DOI 10.1109/IEEE ACCESS.2018.2816919, April 4, 2018.

[18] S. Hussain, *et al*., "Multilevel classification of security concerns in cloud computing," Applied Compuing and Informatics (2017) 13, 57-65.

[19] S. Achleitner, *et al*., "MLSNet: A policy complying multilevel security framework for software defined networking," arXiv:2009.10021. [Online]: https://arxiv.org/abs/2009.10021

[20] E. Zaghloul, *et al.*, "P-MOD: Secure privilege-based multilevel organizatioal data-sharing in cloud computing," 2018, arXiv:1801.02685. [Online]: https://arxiv.org/abs/1801.02685

[21] J. Beacon, *et al.*, "Information flow control for secure cloud computing," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, March 2014.

[22] Oracle Technology Networks, "Comparmented Mode Workstation Labeling: Encodings Format," 2022, [Online]: Available: https://docs.oracle.com/cd/E23824_01/html/821-1480/sec1-4.html .

[23] D. Leinenbach and T. Santen, "Verifying the Microsoft Hyper-V hypervisor with VCC," *FM 2009: Formal Methods*, Springer LNCS 5850, 2009, pp. 806–809.

[24] F. Mayer, K. MacMillan, and D. Caplan, "SELinux by example: Using security enhanced Linux," 2006, Pearson Education, Informit.

[25] B. McCarty, "SELinux: NSA's open source security enhanced Linux," O'Reilly Media, Inc. 2004.

[26] B. Hicks and S. Rueda, "A logical specication and analysis for SELinux MLS policy," *ACM Transactions on Information and System Security*, vol. 13, no. 3, Article 26, Publication date: July 2010.

[27] McAfee Cloud Security, "What is cloud security architecture?" [Online]. Available: https://www.mcafee.com/enterprise/en-us/security-awareness/cloud/what-is-cloud-security-architecture.html

[28] D. E. Bell and L. J. LaPadula, "Secure computer systems: mathematical foundations and model," M74-244, MITRE Corp. , May, 1973.

[29] Department of Defense Trusted Computer System Evaluation Criteria, Department of Defense Computer Security Center, Fort George G. Meade, MD 20755, August, 1983.

[30] M. D. McIlroy and J. A. Reeds, "Design of IX, a multilevel secure UNIX system," CSTR #163, AT&T Bell Laboratories, December 1991.

[31] M. McIlroy and J. Reeds, "Multilevel security in the Uuix tradition," ACM DL: Software-Practice & Experience, August 1992.

[32] H. Kimm and J. Ortiz, "Multilevel security embedded information retrieval and tracking on cloud environments," in *Proc. 2021 IEEE Cloud Summit (Cloud Summit)*, 2021, pp. 25-28, doi: 10.1109/IEEE Cloud Summit 52029.2021.00012.