2013

Developing focused auditing tools: A practical framework for creating formalized multi-level security policy specifications

Barbara Ann Brough

DEVELOPING FOCUSED AUDITING TOOLS: A PRACTICAL FRAMEWORK FOR CREATING FORMALIZED MULTI-LEVEL SECURITY POLICY SPECIFICATIONS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Business Administration

---

by

Barbara Ann Brough

June 2013

DEVELOPING FOCUSED AUDITING TOOLS: A PRACTICAL FRAMEWORK FOR CREATING FORMALIZED MULTI-LEVEL SECURITY POLICY SPECIFICATIONS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Barbara Ann Brough

June 2013

Approved by:

[redacted]

Dr. Joon Son, Chair, Information and Decision Sciences — 6-06-2013 Date

[redacted]

Dr. Tony Coulson

[redacted]

Dr. Jake Zhu, Department Chair

ABSTRACT

Designing and maintaining the security of system information is the primary duty of the cyber security professional. In today's world, nearly all government agencies manage some form of financial, defense, national security, and/or privacy information security policies. It is also necessary in this environment that government agencies are accountable for auditing the security systems that protect this information. However, the great number of security auditing tools and methodologies available still do not solve one key issue: How can auditors create a standard for verifying their security policies are being enforced correctly using these methodologies and tools? It is the premise of this paper that formalized policy specifications and focused penetration testing are needed to effectively audit any information system. This paper offers a framework for creating the semi-formal to formal policy specifications needed to produce a focused auditing tool capable of verifying such policies are being enforced in a Multi-Level Security environment.

ACKNOWLEDGMENTS

I would like to acknowledge the following people without whom this thesis would not have been possible. I would like to thank Dr. Joon Son without whose patience, assistance, and absolute dedication to the work presented in this paper, my thesis would not have been completed. I would also like to thank Dr. Walt Stewart and Dr. Tony Coulson for their help, pestering, and sense of humor in the face of my countless questions. Lastly, I would like to thank Dr. Jake Zhu for his enthusiasm, faith, and support of the work completed in this paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Policy Based Management

Designing and maintaining the security of system information is the primary duty of the cyber security professional. In today's world, nearly all government agencies manage some form of financial, defense, national security, and/or privacy information security policies. Whether they are NIST[1], FISMA[2], or FIPS[3] policies, the accountability and enforcement of these policies remain in the domain of security management and these policies outline the computer security management requirements of agencies operating within this environment. Such policy-based management is considered key in managing large-scale distributed systems such as government agencies operating in a Multi-Level Security (MLS) environment using the Bell-LaPadula model of security classification and categories.

However, due to the complexity of this type of management there is an issue in establishing a benchmark for policy enforcement. While auditing is generally thought of as the standard method for assessing and verifying security policy, the fundamental problem

remains: How to create a standard auditing methodology to achieve policy verification? For most government agencies, auditing is an incomprehensible web of law, best practices, training, and policy standards that make up the requirements of auditing, whether it is financial, operational, or technology based. In addition, government organizations now have the added responsibility of implementing security audits that are used to assess the security and the verification of policy within the network information system.

After completing the audit of the security system, additional testing is normally done to verify if the security policies are, in fact, in place and enforced. Black Box Penetration testing[4] is used to "blindly" look for vulnerabilities that can be exploited, allowing unauthorized access to the system. Verifying the security of systems using black box "blind" testing does establish whether a hacker, without previous knowledge of the system, can find a vulnerability. However, it is still unclear whether relevant security policies are being enforced when it is the normal practice in Black Box testing to only look for a single way in. This is particularly important when the skill level of penetration testers may vary significantly. Without building a

benchmark or framework for the correct enforcement of security policies, a security standard cannot be maintained.

However complex, this type of distributed policy-based management does have one advantage. Its very structure allows for a more standardized approach to configuring the systems according to security policy and allows for the universal implementation of security system-wide. This is particularly important when dealing with large, multi-segmented, distributed systems, covering different geographic areas. By creating semi-formal specifications, security policies can be visualized in diagrams demonstrating how users, objects, permissions, and the various trust boundaries interact from the security policies being employed. By formalizing these policies through the use of specification languages such as XACML, they can be translated into formalized, or programmed, machine-readable "specifications" that can be used to automate the configuration process. In this manner, the security policies are put into "formalized" scripts that are then directly configured to devices on the network. This eliminates much of the human error resulting from manual configuration, as well as

maintaining configuration consistencies while reducing the cost of implementing security policies system-wide.

The main goal of this paper is to create a practical framework for translating these policies into a "semi-formalized" form. These semi-formal specifications, or diagrams, defining MLS access control information flows will allow the reader to translate them in more formal specifications using the programming language of their choice, whether that is eXtensible Access Control Markup Language[5] (Moses, 2005), Ponder[6] (Damianou, 2001), or something else. The motivation behind this framework is to assist agencies in standardizing their approach to implementing policies and thereby increase the level of security, enforce policies correctly throughout the network, and automate policy configuration. This is especially useful when creating and implementing policy on new systems early in the system design phase of the System Development Life Cycle[7] (SDLC). By implementing this framework on new systems, the need for the verification of policy is mitigated. In addition, when applying this framework to existing systems, it adds an extra step in the verification of existing policy specifications through the use of an auditing tool, the penetration-testing database. By formalizing specifications, through the use

of high-level programming languages, agencies will not only be able to automate their configuration process but also verify their system's policies using this more focused auditing and penetration testing method. Without creating a semi-formal or, more specifically, a more formalized policy specification, auditing and testing cannot be accurately performed on a system. It is through the process of auditing and testing a system that make it possible to determine whether the security policies in place are actually being enforced correctly.

Specifically, this paper proposes a practical framework for combining the LDAP information directory tree (DIT) and MLS information flow policy to create semi-formalized MLS policy specifications. To gain the granularity necessary, this paper also proposes using the Lightweight Directory Access Protocol model (LDAP). The LDAP, through its own physical model, mimics the organizational structure of the agency. Combining the LDAP DIT with the MLS information flow policy permits the "allowed" information flows between agency organizational units to be defined. By mapping the information flows to the organization, security personnel can see distinctly the trust boundaries where access control becomes paramount.

To account for the additional security requirements within the MLS environment, we also propose adding additional attributes to the LDAP model. By creating additional LDAP attributes to filter these latent security requirements, shown by overlapping the LDAP DIT with the MLS information flow policy, we create an MLS "aware" LDAP. Through this approach it is possible to design greater granularity in security policy enforcement, such as security sensitivity and classification, category and IP address range. By carrying these additional security attributes, the information flows show the direction and level of the flow of information as it pertains to each user request.

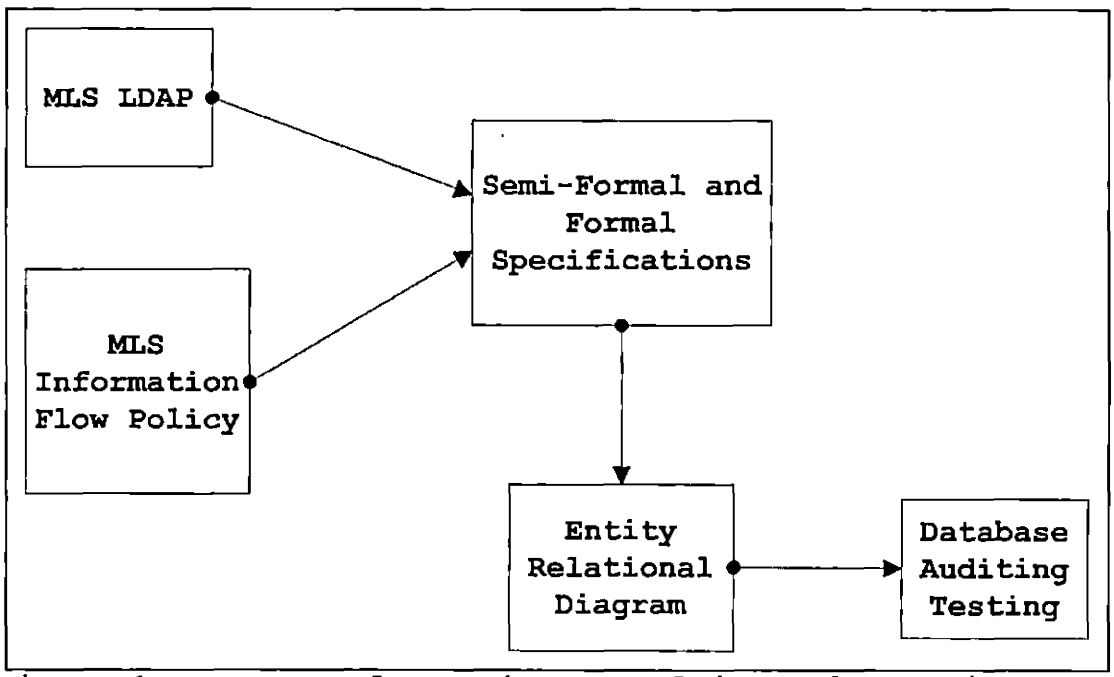

Figure 1. Process of Creating an Multi-Level Security Aware Lightweight Directory Access Protocol and Designing the Focused Testing Tool

The semi-formalized, or formalized, specification represents a security topology of the users' permissions similar to a map and also shows the location of trust boundaries[8] within the system. By analyzing semi-formal specifications, agencies can identify how policies translate to devices on the network and determine where the vulnerabilities resulting from such policies may impact the system as a whole.

By basing our method on this type of semi-formalized policy specifications, agencies can implement policies governing the authorized activities of legitimate users and more accurately mediate user access to resources on the system. It is also possible to specify the permissible location of the user and assets, verify whether authentication and authorization have been implemented correctly, and determine through audit and testing whether information flows contradict security requirements in any way.

In addition, organizing more focused penetration testing could be accomplished by designing a database to reflect our practical framework and information flow policy diagram to generate queries regarding potential vulnerabilities. Applying the LDAP's customized MLS attributes to query the target and user's permissions, the auditor could be given the corresponding vulnerabilities to be tested determining whether the required mitigation was actually in place. It is through using this tool that auditors would know, whether policies were being enforced across the board, rather than using black

box testing to look blindly for a single hole in the system.

CHAPTER TWO

BACKGROUND AND RELATED WORK

In order to fully understand the requirements of the MLS environment and the corresponding need for this framework, it is necessary to completely understand the origins and basis of the security environment, Multi-Level Security[9] (MLS), and the access control therein. This chapter details general background information about the Multi-Level Security (MLS), Bell-LaPadula (BLP) access model, and Lightweight Access Directory Protocol (LDAP). Related work focusing on verification of policy, access control mechanisms, specification languages and the results of the work already done in these areas are also discussed below.

2.1 Multi-Level Security (MLS)

"Many organizations, such as the military services, intelligence organizations, related government agencies, and their supporting defense industries require Multi-Level Security (MLS) model systems. These systems enable concurrent processing of data that is classified with respect to different levels of security. MLS has a capability that allows information with different sensitivity levels to be simultaneously stored and

processed in a system accessed by users that have different levels of security clearance, as seen in NISTIR 7316[10]" (Son, 2008). In this paper the sensitivity or security levels are unclassified, classified, secret, and top secret.

"The rationale behind the MLS model is to secure information at a higher security level from access by users at a lower level of security clearance. One mechanism for implementing this policy is to assign security labels to all assets in a system" (Son, 2008). The labels represent the level of sensitivity of the information or the level of sensitivity to which a subject is allowed access. This mechanism prevents subjects from accessing information with a security label for which they are not cleared. To decide whether a specific access mode is allowed, the clearance of a subject is compared to the classification of an object. This is the approach taken in the well-known Bell-LaPadula (BLP)[11] access model (Bell, 2005). The BLP model is explained in the next section.

2.2 Multi-Level Security (MLS) Policy Model

The main goal of MLS policies is to regulate how information may flow between designated sensitivities. MLS polices are designed to control the confidentiality of

information flows and prevent the information leakage from an entity of a higher sensitivity level (High) to an entity of a lower sensitivity level (Low). Protecting against such a leakage of information from High to Low is very important to nearly any organization whether governmental, military or private commercial enterprise.

Information flow policy[12] in MLS systems can be defined by a lattice model[13], which was introduced to describe policies and channels of information flow: an information flow policy is defined by a lattice (SC, ≤), where SC is a finite set of security classes (or entities), and ≤ is a binary relation partially ordering the classes of SC. For example, for security classes High and Low, the relation Low ≤ High means class Low information is lower or equal to class High information (it is said that class High *dominates* class Low). Information is permitted to flow within a class or upward, but not downward or to unrelated classes. Thus, class Low information is permitted to flow into class High if and only if Low ≤ High. The most famous and influential security policy model which deals with information flow in the MLS system was first introduced by David Bell and Elliot LaPadula in 1973 for the Department of Defense (Son, 2008).

Bell-LaPadula (BLP) access model was proposed to enforce access control in government and military applications and supports Mandatory Access Control[14] (MAC) by determining the access right from the sensitivity levels associated with subjects and objects. These subjects are given a sensitivity level or security clearance, and objects are also given a similar security classification. To properly enforce access control in a MLS environment, the Bell-LaPadula access model defines two security properties:

- Simple Security (SS) property: subject S can read to object o only if the security level l_s of s dominates the security level l_o of o, i.e., $l_o \leq l_s$. This property is also known as "no-read-up" (Son, 2008).
- *-property: subject s can write to object o only if the security level l_o of o dominates the security level of l_s of s, i.e., $l_s \leq l_o$. This property is also known as "no-write-down" (Son, 2008).

To allow for greater granularity of information control, the BLP model was expanded by adding categories that group information into a need-to-know-basis. These

categories serve to restrict access to certain types of information, while keeping users within the confines of their security clearance. A subject must have a superset of the object's categories to dominate the object. The categories and security sensitivity or clearance allow for a more granular restriction of information assets as well as tighter information control throughout the system as a whole.

Figure 2 illustrates an MLS information flow policy enforced by BLP model. The arrows in the figure show the direction of permissible information flow. For example, as shown in the figure, information can flow from entity A with a sensitivity of unclassified and {C1} as the category to entity B with a sensitivity of classified and {c1, c3} as the categories; This is because the sensitivity of classified dominates the sensitivity of unclassified (classified $\leq$ unclassified), and {c1, C3} is a superset of {c1} (i.e., {c1} $\subset$ {c1, c3}). However, the security policy is violated if information flows from an entity with sensitivity classified and {c1, c3} as the categories to an entity with a sensitivity of secret and {c1, c2} as the categories. This is because classified $\leq$ secret, and {c1, c2} is not a superset of {c1, c3} (i.e., {c1, c3} $\not\subset$ {c1, c2}) - this is why there is no arrow

drawn from classified with {c1, c3} to secret with {c1, c2}.

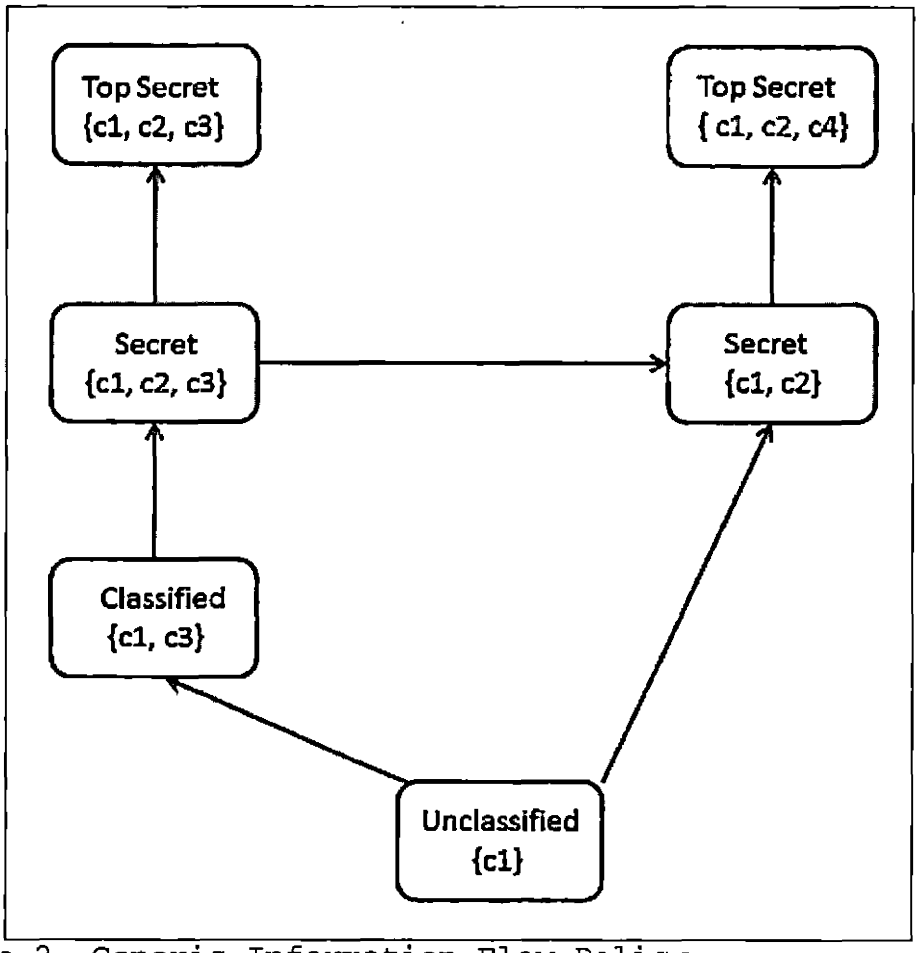

Figure 2. Generic Information Flow Policy

2.3 Lightweight Directory Access Protocol (LDAP)

As a protocol for accessing directory services, development of the LDAP began in the 1980s by the International Telecommunications Union and Telecommunications standardization sector (ITU-T) and the

International Standards Organization (ISO) for Unix and Linux to improve network communications. LDAP was initiated in conjunction with the development of Transmission Control Protocol and Internet Protocol (TCP/IP) and is the standard for internet communication today. The LDAP model is outlined in Request for Information (Wahl, 1997). The LDAP was intended to provide access to directories supporting the x.500 model.[15] The LDAP is referred to as "lightweight" because it does not require the same intensive amount of resources as x.500 and it is not as complex to implement. As a directory service, X.500 was a weighty and complex platform that required the use of the OSI stack and was not compatible with TCP/IP. The x.500 protocol lacked the flexibility of the LDAP model and could not be used with TCP/IP, which was quickly becoming the standard for internet packet transmission. In addition, unlike the x.500 model, the LDAP protocol was designed specifically for management applications and browser applications that offer read/write interactive access to directories (Wahl, 1997).

LDAP directories use a hierarchical model to store data about the system. The LDAP protocol assumes that there are one or more servers, with each providing access to the Directory Information Tree (DIT). The DIT is a

collection of entries, where each has several attributes that declare a Distinguished Name (DN), that is made up of one or more of these attribute values. The Distinguished name (DN) must be unique within the DIT and can be created from a concatenation of several attributes. For example: DN: CN = Betty Brown, O = Federal Energy Regulatory Commission, C = US.[16]

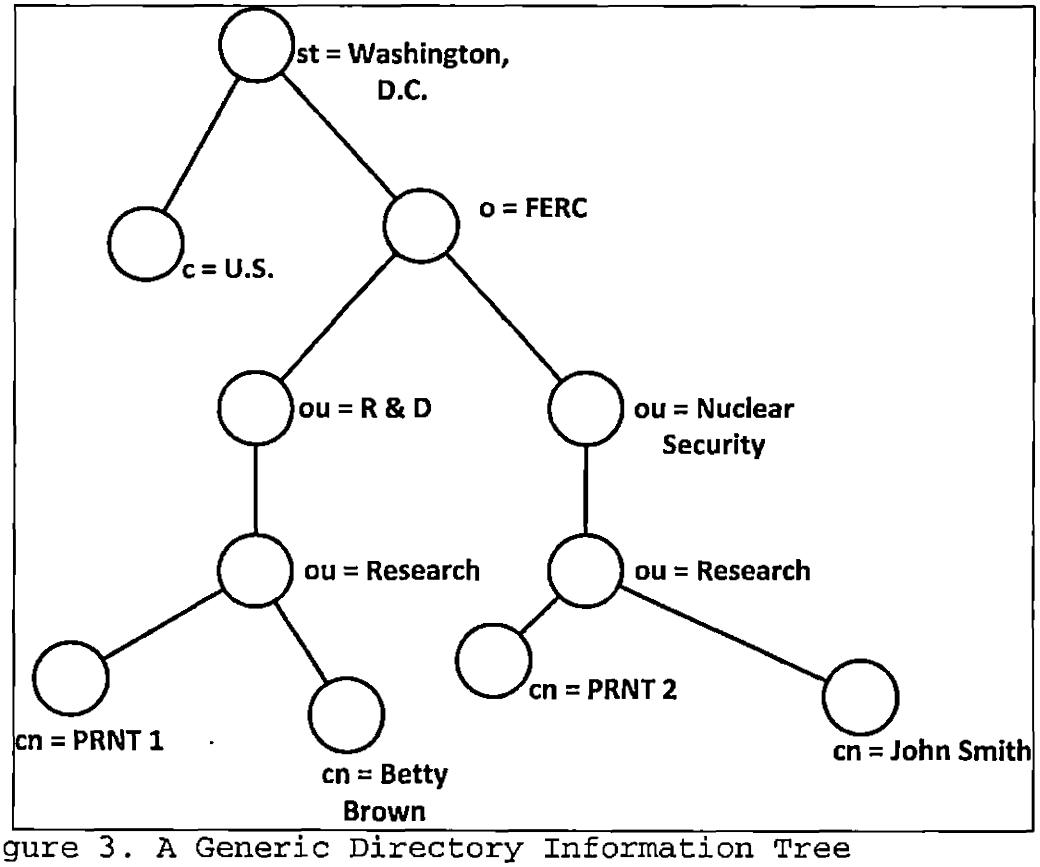

Figure 3. A Generic Directory Information Tree

Each entry must also have an objectClass. The "objectClass" refers to the class that is assigned to the

entry. ObjectClasses define what attributes are required, optionally required and their attribute type. It also refers to how many values can be stored, as well as the kind of attribute stored. Attribute type refers to what kind of information is being stored, outlines the syntax that is to be used when creating the attribute, the kinds of matching that can be applied to the values of each attribute, and any other functions allowed.

As added flexibility in the LDAP model, clients can modify the values for each objectClass; however, servers can restrict these modifications to prevent the basic structure of the class from becoming too altered. Servers must also prevent clients from adding additional classes not already introduced in the schemas. There are many different types of attributes ranging from operational- which cannot be modified because they are used for administering the directory itself- to optional and naming attributes, which can be altered to suit the needs of the client.

In addition to the (DN), the LDAP also calls for attributes such as organizational unit (OU), which refers to the name of the container the entry belongs to, e.g., department, type, or location of an entry. The organizational unit allows the LDAP to break down the

organizational structure into smaller and smaller units, or containers, within the directory. These mandatory attributes may also be single or multi-valued as well (Arkills 2003). In addition, there are naming attributes that make up the LDAP directory, like Common Name (CN), and Domain Component (DC) which give the entry's name and the type of network it is on, such as .org, .com, or .edu. Naming attributes are used for access control, such as User ID (UID), Group ID (GID), User ID Number (uidNumber), and Group ID Number (gidNumber), all of which control user access authentication on the system (Wahl, 1997).

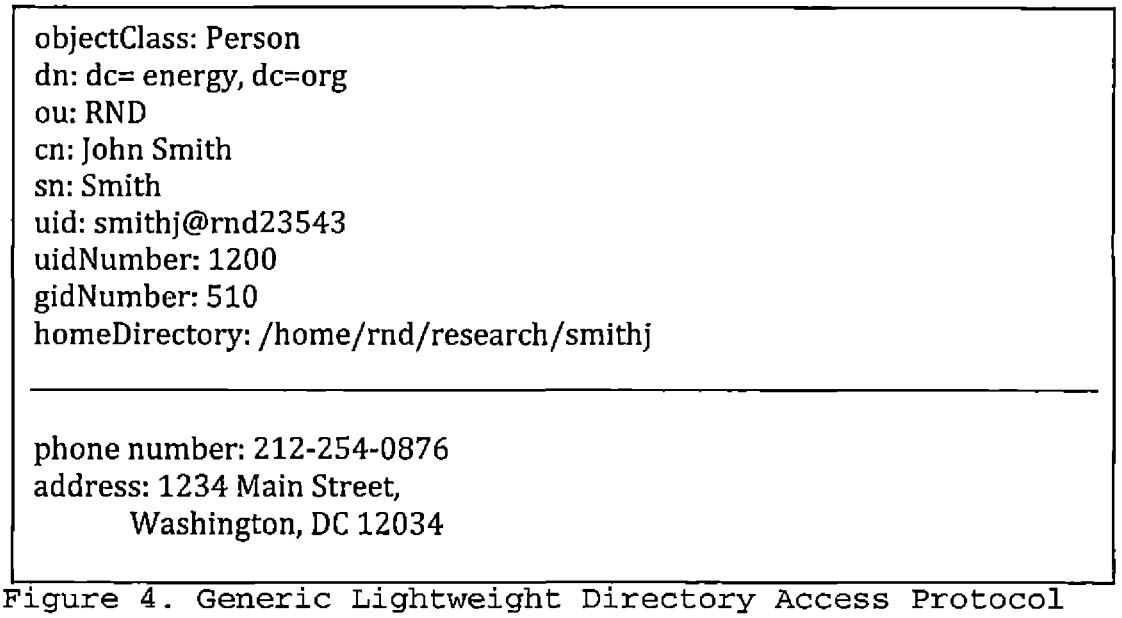

Figure 4. Generic Lightweight Directory Access Protocol Entry

To maintain control over the various attribute types, the LDAP model uses the schema as a way to sort and order the attribute type definitions, object class definitions, and other information. The server uses this information to find how to match a filter, or attribute value insertion, when comparing an operation request from a client against the attributes of an entry (Wahl, 1997). There are several default schemas used when creating an LDAP server, as well as option of creating a customized LDAP schema. These default schemas are available for download on the various LDAP websites for all the different LDAP software platforms. OpenLDAP, for instance, offers several default distributed schemas on their website: core.schema, cosine.schema, inetorgperson.schema, misc.schema, nis.schema, and openldap.schema (OpenLDAP Software 2.4 Administrator's Guide, 2013).

To create a customized schema for an LDAP directory, a local.schema must be employed. The local.schema allows customized attributes to be created and defined for implementation into the directory. The schemas are located in Linux as: /etc/openLDAP/slapd.conf. The local.schema would be added to this file with the rest of the distributed schemas in order to include the customized object class and attribute definitions. It is through the

DIT organizational structure and use of the additional attributes in the custom schema that allows more transparency in the security filtering process and allows auditors to see how security policies impact the system. The following is some of the content of a slapd.conf file showing how to implement a local.schema using CentOS 5.8 (Linux).

```
#See slapd.conf (5) for details on configuration options.
#This file should not be world readable.
#
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/local.schema
```

Figure 5. Slapd.conf File Schema List

The local.schema, or custom schema, would include the object classes, matching rules, and data type rules for value insertion, as well as definitions for the attributes themselves (See Appendix A for an example of implementing a custom schema based on this framework). Based on the above DIT, the attribute definitions would include the following information: the heritance (if any), the equality (comparison rules), Substring rules (if any), and

the syntax rules (OpenLDAP Software 2.4 Administrator's Guide, 2013).

To begin creating a new schema for an LDAP design, the local.schema will need to be renamed to reflect the custom object class and attributes being added to the directory. For example, if making a custom schema for mapping an IP range and location to a local person type, make a local.schema and use the following path: /etc/openLDAP/schema/local.schema.

The attributes are included in the schema first because they must be read first by the LDAP before being included in the object class definitions. As is shown in the following example:

```
#New attribute definitions:
attributeType ( 1.1.2.2.1 NAME 'ipRange'
        DESC 'IP Range'
        EQUALIITY caseExactMatch
        SUBSTR caseExactSubstringsMatch
        SYNTAX 1.1.2.2.1.1466.115.121.1.15(1024))

attributetype ( 1.1.2.2.2 NAME 'location'
        DESC 'location'
        EQUALIITY caseExactMatch
        SUBSTR caseExactSubstringsMatch
        SYNTAX 1.1.2.2.1.1466.115.121.1.15(1024))
## objectClass definitions for 'mlsPerson' depends on the core.schema.

objectClass (1.1.2.2. Name 'localPerson'
        DESC "local person type'
SUP top Structural
        MUST ( ipRange $ location )
        MAY ( userPassword )
)
```

Figure 6. Generic Custom Schema

Following the creation of the custom schema, the LDIF file would need to be configured to accept the new data attributes, create the directory structure, and would allow new entries to be created. The slapd.conf file is important because it outlines the structure of the LDAP directory service on the server. It also specifies all the objectClass and attribute's rules (schemas), global and otherwise, which govern the LDAP directory. The order of the schema files listed in the slapd.conf file is also important (OpenLDAP Software 2.4 Administrator's Guide, 2013). Like the custom schema, the attributes need to be

read before they can be included in the new objectClass definition.[17]

Once the schemas are complete, the slapd.conf file is amended to reflect the new schemas (see *Figure 6*). The LDIF file is built to outline the directory's structure, and the LDAP can be used and new entries added. By including LDAP model to the design of the MLS access control, there is flexibility to use it as an auditing tool. Because of the methodology in designing the DIT, mapping the Information Flow policy to its structure is a natural process and creates a visual representation of the access control mechanism itself. Designing an auditing tool necessary to utilize this information is the next logical step following creating the methodology for generating the semi-formal specifications of the MLS information flow policy.

```
dn: dc=energy, dc=.org, o=ferc
Description: Custom objectClass for MLS sensitivity and category objects in
 LDAP

cn: energy
objectClass: top
objectClass: person
objectClass: mlsPerson
name: Betty Brown
sn: Brown
ou: R&D Dept
location: Washington, DC
ip range: 192.168.1.10/254
phone: 212-234-0987
email: brownb@energy.org
```

Figure 7. Generic LDAP Data Interchange Format File

2.4 Related Work

Access control is one of the most important and widely used authorization policies available. It controls which subjects such as users or processes have access to which resources, or objects, in a system. Over the past several decades, many access control policies, or models, have been proposed. Some of models introducing the early concepts of access control mechanisms were Discretionary Access Control (Lampson, 1971), Mandatory Access Control (Bell & LaPadula, 1976), Task Based Authorization Controls (TBAC) (Thomas, 1997), Role Based Access Control (RBAC) (Ferraiolo, Sandhu, & Gavrila, 2001), and Organization Role Based Access Control (ORBAC) (Kalam et al., 2003).

While these studies lay the foundation for access control as we know it today, none of these models were able to prescribe the methodology for verifying security policies. However, the study by Kalam et al (2003), does add some insight in its focus on the concept of organization; it proposes the Organizational Role Based Access Control policy model (ORBAC). Using the concept of organization, a security policy can be applied to a target organization and is defined as a collection of permissions, prohibitions, obligations, and recommendations. Even so, it does not offer the added granularity of the MLS configuration or the flexibility and organization of the LDAP directory configuration. The ORBAC model does not offer the same stringent access control that is offered in the combination of the MLS aware LDAP model and can be complex in its administration. The use of an LDAP is proposed to simplify the task of managing security in a large distributed system (DMTF: DEN Initiative, n.d., Jamhour, 2001).

As today's information systems are rapidly growing in scale and complexity due to an emergence of new technologies and security requirements, policy-driven management is gaining popularity. Policy driven management systems have been researched to specify their targets,

constraints, and access control mechanisms in the form of policies. Policy languages are used to write specifications for the policy-driven management systems and categorized into network management policy languages and security management policy languages (Han & Lei, 2011). Network management policy languages aim to allocate resources within a network according to the system requirements (e.g. bandwidth, device configuration, access control, etc.) whereas Security management policy languages focus on the protection of system resources and the administrator's method of security management. For the purpose of reference, lists of the most widely used in this paper are as follows.

Table 1. Network Specification Languages

Network Policy Language	Reference Paper
Knowledge Acquisition in automated Specification (KAOS)	Dardennen, A. V. (1993). Goal Directed Requirements Acquisition. *Science of Computer Programming 20, 3-50.*
Policy Description Language (PDL)	Lobo, J. B. (1999). A Policy Description Language. *16th National Conference on Artificial Intelligence* (pp. 291-298). Orlando, FL: Association for the Advancement of Artificial Intelligence.
Ponder	Damianou, N. D. (2001). The Ponder Policy Specification Language. *Policy 2001: Workshop on Policies for Distributed Systems and Networks* (pp. 18-39). Bristol, UK: Springer.
CIM Simplified Policy Language	DMTF Policy Working Group: Lobo, J. B. (2009). *CIM Simplified Policy Language DSP0231.* Portland, OR: Distributed Management Task Force, Inc.

Table 2. Security Specification Languages

Security Policy Language	Reference Paper
A P3P Preference Exchange Lange (APPEL)	Cranor, L. L. (2002, April). *A P3P Preference Exchange Language 1.0 (APPEL1.0)*. Retrieved April 2013, from World Wild Web Consortium: http://www.w3.org/TR/P3P-preferences/
Rei: A Policy Specification Language	Kagal, L. F. (2003). A policy Language for a Pervasive Computing Environment. *IEEE 4th International Workshop on Policies for Distributed Systems and Networks* (pp. 6374). Lake Como, Italy:IEEE
eXtensible Access Control Markup Language (XAMCL)	Mosses, T. (2005, February). *OASIS eXtensible Access Control Markup Language (XAMCL)* Version 2.0 Retrieved April 2013 from OASIS Open: http://docs.oasis-open.org/xacml/2.o/access_controll-xacml-2.0-core-spec-os.pdf
Platform for Privacy Preferences Project (P3P)	Cranor, L. D. M. (2006, November). *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. Working Group Note*. Retrieved April 2013, from World Wide Web Consortium: http://www.w3.org/TR/P3Pp11/

Currently, out of all the known security policy languages listed in table 2, XACML is the most widely accepted, both in industry and academia, as a de facto standard. XAMCL is a declarative, XML based policy language, mainly employed for access control management in distributed systems. The framework proposed in this paper is offered as a method of preparing policy implementation for translation into formalized form through the use of policy languages such as XACML. By creating this

methodology, the process of simplifying policy into machine readable or formalized specifications becomes standardized and lends itself towards the correct implementation of security policy. While many of the papers mentioned previously discuss the access control mechanism or language model, they do not offer a similar framework for its application, nor a method of verifying that the policies are being applied effectively.

As access control policies become more complex and are implemented to manage large distributed networks with many different organization units, policy makers and auditors will find it difficult to assure that policy specifications are correct, thereby allowing that these policies are incorrectly enforced or implemented. Much research has been developed to deal with the conformance checking of access control policies for different security levels (Hu et al., 2007, 2011, Bryans, Fitzgerald, & Periorellis, 2006, Bryans, 2005, Hughes & Bultan, 2008). These approaches are based upon formal methods such as CSP (Hinchey & Jarvis, 1995), Alloy model checker (Jackson, 2000), Vienna Development Method Specification Language (Hansen & Bruun, 1996), among others. Besides being focused on Role Based Access Control, these models are heavily math based in nature, making them problematic for

use by people not well versed in this area. In addition, these formal verification models do not easily map onto implementation mechanisms (e.g., organizational structure and units), making it harder for auditors to test whether a target network has implemented or enforced the access control policy correctly.

CHAPTER THREE

MULTI-LEVEL SECURITY ACCESS CONTROL POLICY

The main objective of this chapter is to demonstrate a practical way to create an MLS access control policy from an information flow policy diagram and an MLS LDAP model of an organization.

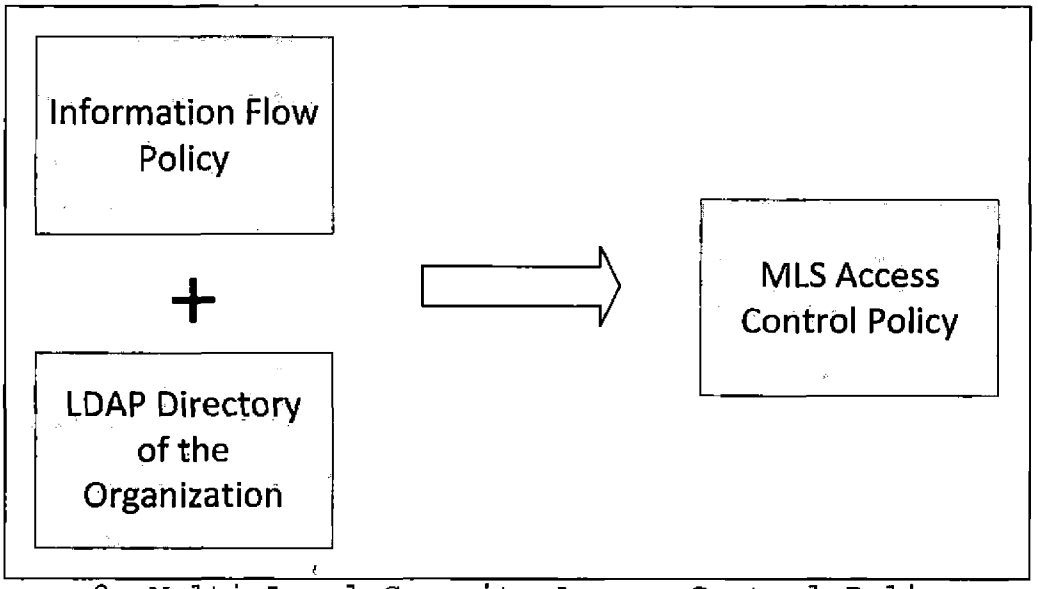

Figure 8. Multi-Level Security Access Control Policy Diagram Creation

3.1 Multi-Level Security Access Control within a Department

The MLS Information Flow Policy relates the security policies (clearances, classifications and categories) and how they interact with the information flows between entities. In the example shown above, we have created an

information flow policy based on our fictitious example of the Department of Energy. Within this information flow policy there are four clearances/classifications and five categories for the agency depicted.

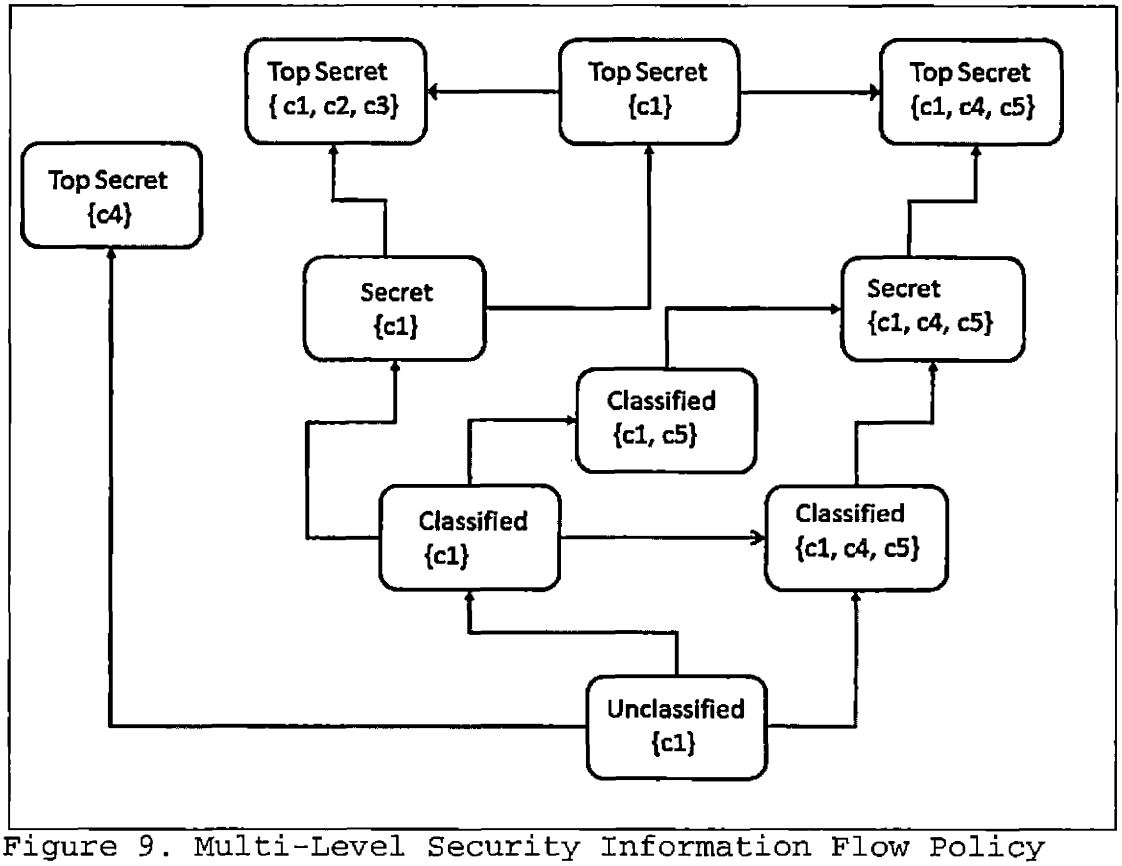

Figure 9. Multi-Level Security Information Flow Policy Diagram

The clearances/classifications range from unclassified, classified, secret and top secret. The categories or need to know can be tailored to any agency's specific information security requirements and are

therefore not described within this thesis. However, when combined, the access control mechanism is a combination of clearance, classification, and categories.

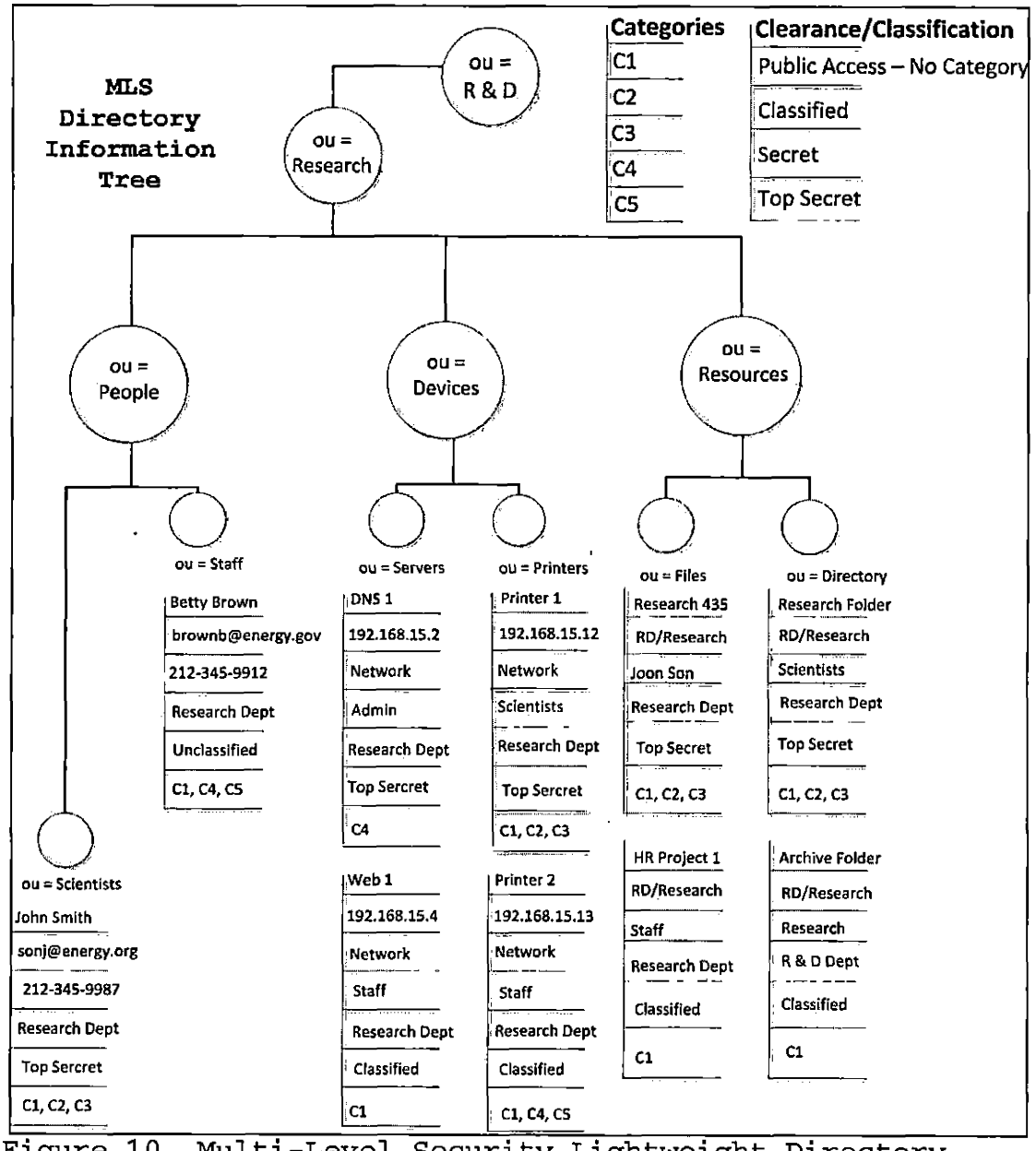

Figure 10. Multi-Level Security Lightweight Directory Access Protocol Directory Information Tree for the

Research Sub Department of the Research and Design Department

Figure 10 outlines a simple example of an MLS Directory Information Tree (DIT) and contains the additional attributes for security sensitivity and categories discussed in the previous chapters. The MLS DIT shows many different attributes for the Research Department. These attributes assist in the defining of permissions by filtering location or IP address, role, sub-department, and department information. The additional attributes created in the custom schema demonstrate the MLS attributes of sensitivity and categories.

```
#base OID 1. 2. 3. 4. 5. 1021. x. y
#x = 4 for objectClass
#y = 3 for attributetype
objectIdentifier MLSschema 1.2.3.4.5.1021

attributeType ( MLSschema: 3. 1 NAME 'sensitivity'
DESC 'MLS sensitivity level'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1. 3. 6. 1. 4. 1. 1466. 115. 121. 1. 15{40}
)
attributeType ( MLSschema: 3. 2 NAME 'categories'
        DESC 'MLS categories'
        EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1. 3. 6. 1. 4. 1. 1466. 115. 121. 1. 15{40}
)

ObjectClass ( MLSschema: 4. 1 NAME 'mlsperson'
        DESC 'MLS person type'
        SUP inetOrgPerson STRUCTURAL
        MUST ( sensitivity $ categories )
        MAY ( userPassword )
)
```

Figure 11. Customized Schema for Multi-Level Security Access Control

Demonstrating the application of these new attributes, the two subjects, John Smith, Scientist and Betty Brown, staff also illustrates the flexibility of the LDAP model. John Smith has a clearance of top secret, with a category of {C1, C2, C3}. This clearance and category give John Smith access to information at top secret or lower with the necessary categories, or need-to-know, to access workgroup, sub Department, and Department information. As the objects listed in the DIT apply the

MLS information flow policy showing classification level, as well as category restrictions, the subject wishing to commit an action on a particular object must hold a clearance equal to or greater than the classification of the object, as well as a superset of the object's categories, to commit any action on any particular object.

According to the MLS information flow policy, John Smith can access certain resources, assets, or devices on the network, such as a printer or server. However, there are also objects that he cannot access because he does not hold the correct category superset to do so. To demonstrate the filtering that takes place when the sensitivity and category attributes are implemented in the LDAP, please refer to *Figure 12* showing the MLS Access Control Diagram for John Smith (scientist) and Betty Brown (staff) in which the relationship between the two subjects and the available objects within the research sub-department interact under the influence of the information flow policy. The MLS access control policy diagram illustrates the filtering process between objects and the permission restrictions of John Smith and the permissions restrictions of Betty Brown.

It is through combining the attributes of sensitivity, category, location, and object that the

focused auditing tool can determine the potential vulnerabilities and allows the auditor to focus their penetration tests to verifying that the policies protecting the security of these assets have been implemented correctly. In comparison, black box testing blindly looks for a single hole throughout the entire system rather than focusing on where the holes are most likely to be. By employing both methods of penetration testing, completing a thorough and productive audit of the system's security is much more likely.

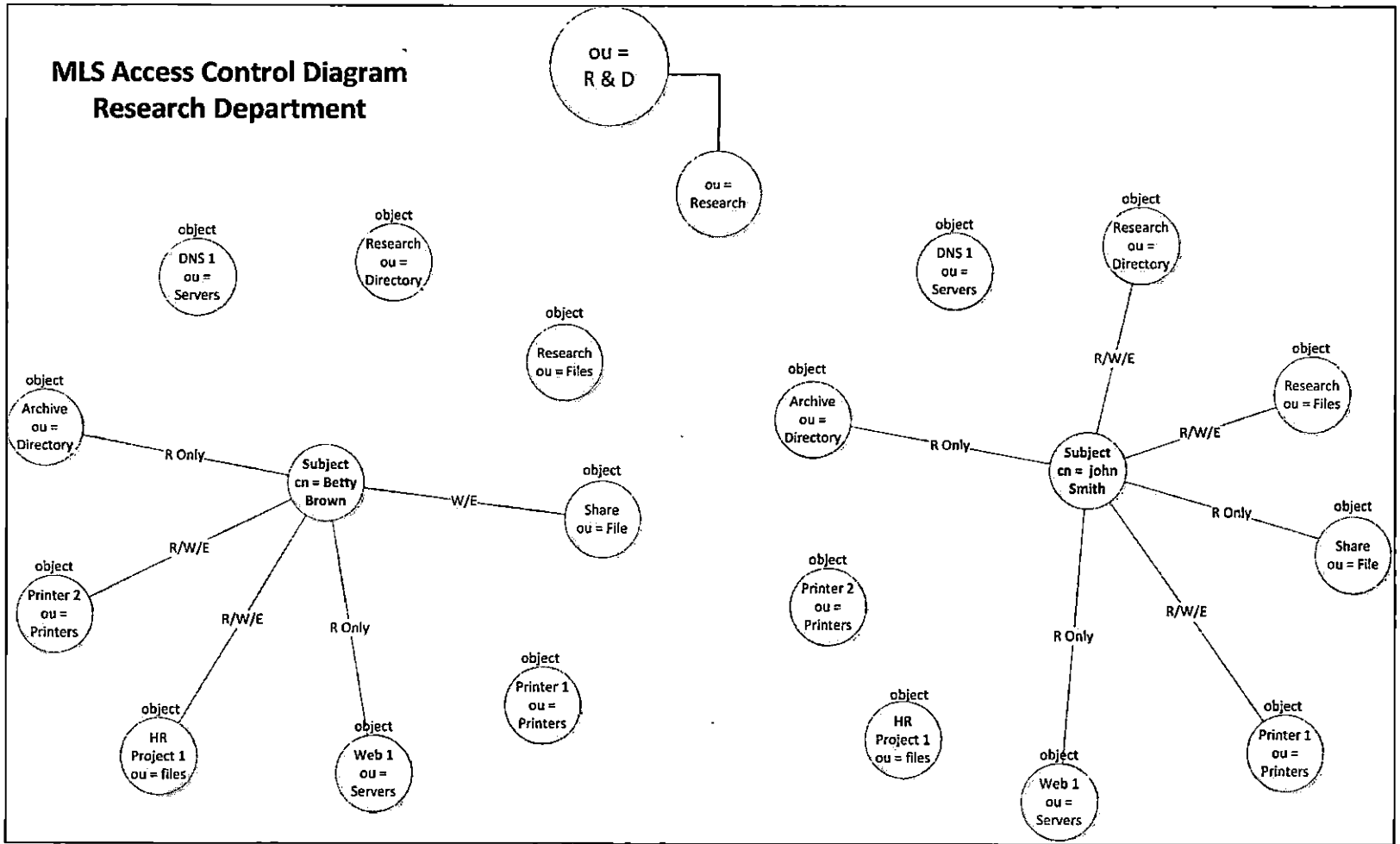

Figure 12. Multi-Level Security Access Control Diagram

The MLS access control policy diagram applies the MLS information flow policy to the DIT and demonstrates how the subject may access, or is prevented from accessing, resources and devices on the research network, and this depends on the sensitivity and category restrictions. The subject John Smith holds a top secret clearance and categories of {C1, C2, C3} and has access to classification levels of top secret or lower as long as he holds a superset of the objects' categories. For classification levels lower than top secret, it is no-read-up and no-write-down. This indicates that John Smith can access or read objects at a lower level, but he cannot change or alter these objects in any way. This also assumes that he holds a superset of the objects' categories as well. The same holds true for Betty Brown, who can access or read objects at a classified or lower level but cannot change or alter those objects in any way; This maintains the objects' information integrity at all times.

3.2 Information Flow between Sub Departments

The below DIT includes added people, resources, and devices and shows the sensitivity restrictions that filters and controls the subject's access of objects and

the corresponding object classifications and protections. As the complexity of the DIT increases the object classes and attributes built in the schemas of the LDAP, particularly those built into the custom schema, acts as the filters for the access control mechanism defined by the MLS Information Flow Policy. By building these additional security "filters" in the LDAP, access control become significantly more transparent, effectively applying the permissions/restrictions of the MLS environment. Within the LDAP model directory, it is possible to see the organization as a whole, defined by the access control attributes which give the DIT its filtering power.

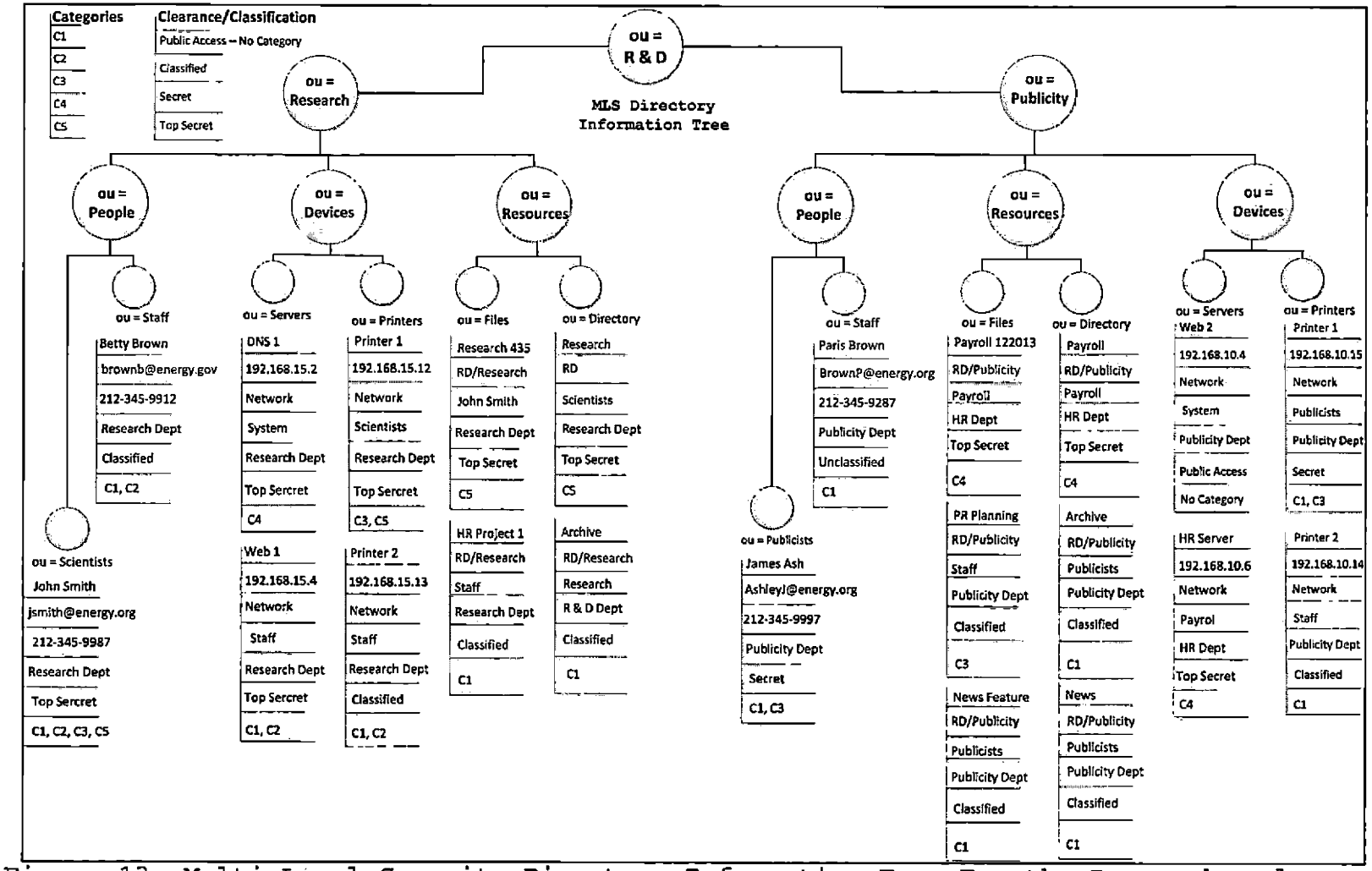

Figure 13. Multi-Level Security Directory Information Tree For the Research and Design Department

Figure 14 details the effect of sub-department flows on the MLS Access Control Diagram between Sub-Department Research and Sub-Department Publicity. Here the relationship between the Research personnel and objects from both the Research and Publicity Sub-Departments are illustrated. By creating this MLS Access Control Diagram trust boundaries can be identified and potential threats and vulnerabilities can be identified for the auditor to investigate.

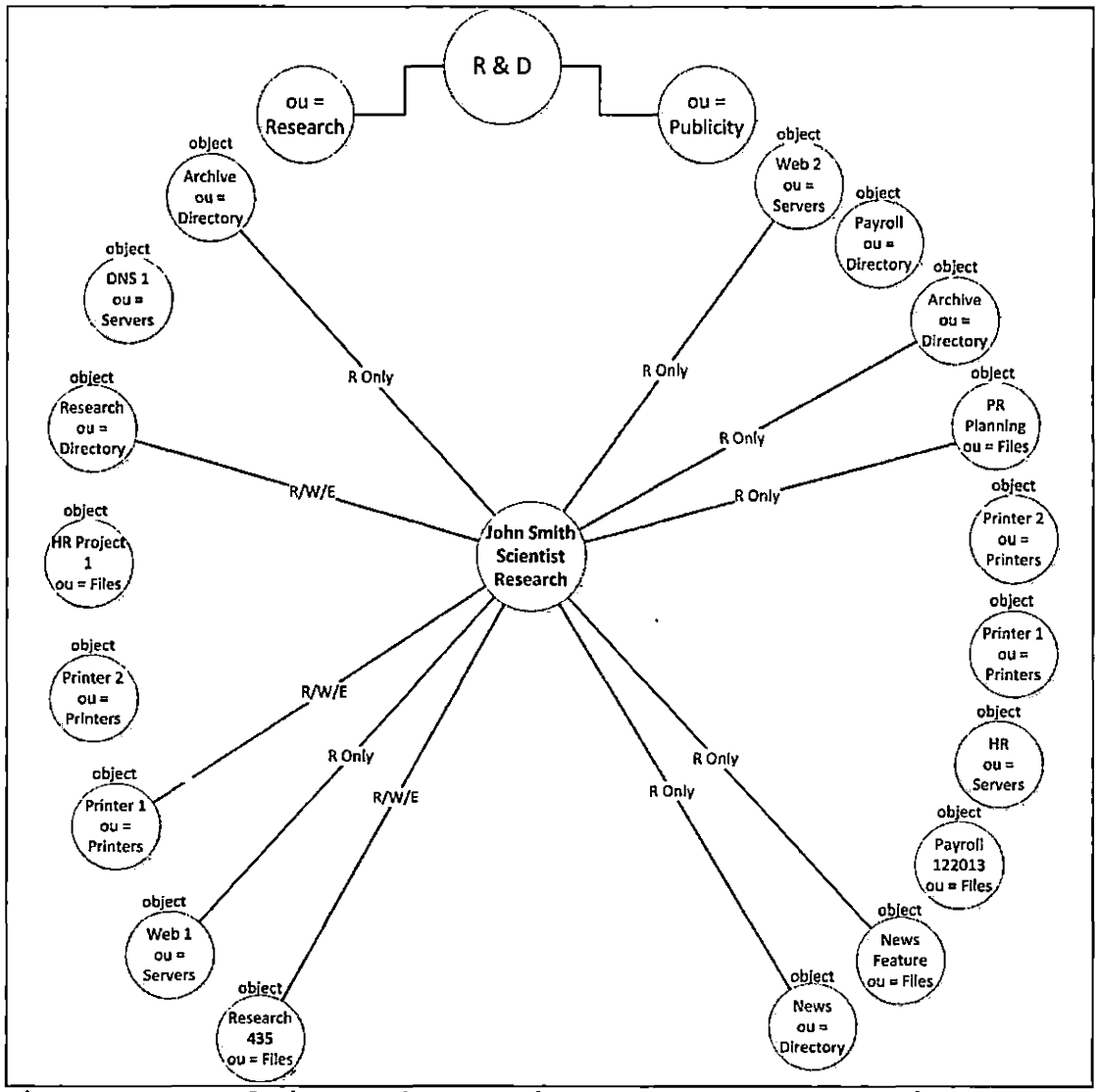

Figure 14. Multi-Level Security Access Control Diagram between Sub-Departments Research and Publicity

The relationship between subjects of differing sub-departments demonstrates the filtering and compartmentalization process of applying the MLS Information Flow Policy of the organization. Enforcing the "no-read-up, no-write-down" can be clearly seen in the

"flows" between sub departments and the subject John Smith. Using the LDAP to create additional filtering, or application of the MLS policies, can create synergies in controlling the flow of information from one subject to another. This assists auditors and security professionals in creating new methodologies and tools for verifying that these MLS policies are being enforced properly. By standardizing this process of building semi-formal specifications, it allows security professionals and auditors to become familiar with their policies and systems in such a way that the potential vulnerabilities become very apparent to them. In addition to applying black box testing to find the "unknown" vulnerability in the system using this auditing tool allows the IT staff to continually look for holes in the implementation of their security policies. Without the white box testing suggested earlier in this paper, organizations are only as secure as the skill level of the ad hoc talent of their testing personnel.

CHAPTER FOUR

DEVELOPING AUDITING TOOLS

4.1 Verifying Multi-Level Security Policies

Auditing and the process of verifying security policies can overwhelm the best security professionals. Depending on the security requirements that an organization adopts, security audits can require hundreds of man-hours and often interrupt business processes if not managed correctly. Auditing often becomes a messy business and can impose high costs on both agencies and private organizations if not properly prepared. It is imperative that through the process of the security assessment auditors are able to clearly define the security goals of the organization. A security audit may be different for every organization, but they typically they include the following.

Auditors must first determine the scope of their security audit. By doing so, they can limit the depth and focus of the audit, controlling the size and organization of the audit process. The *Management Planning Guide for Information Systems Security Auditing* (December 2001) outlines the auditing planning process written as a joint initiative by the National State Association (NSAA) and

the U.S. General Accounting Office (GAO). In this document, the GAO describes the security auditing planning process and creates a guide for organizations looking to set up auditing within their agency. However, the remaining problem still exists in what is the most effective way to verify that security policies are being implemented and enforced correctly, and that the risk of lost, stolen or destroyed information is minimized. Typically the following steps are included in the audit process.

1. Define the physical scope of the audit. This usually involves determining the area of the organization being audited and tabulating the physical assets that have been determined to be critical to the organization.
2. Define the process scope of the audit. This means determining the critical processes to the organization and their security.
3. Develop a historical reconstruction of any known security breaches, known vulnerabilities, and issues that relate to the defined scope of the audit.
4. Create the actual plan for conducting the audit itself. Where will you start and how will you

assess the system? This usually involves creating a description of the audit, any significant dates, participants to the organization and dependencies.

5. Complete a thorough security assessment of the assets and processes within the organization. This includes:
   a. Identify the exact location of assets within the organization.
   b. Identify the potential threats to these assets.
   c. Document the perceived vulnerabilities to these assets and processes.
   d. Outline the current security controls in place protecting these assets and processes.
   e. Determine the quantitative likelihood of threats being realized and the monetary impact of these threats would have if successfully exploited.
6. Document the results of the audit.
7. Specify and update assets with regards to any new mitigation scenarios and newly established controls.

8. Usually accompanying a thorough audit is the use of black box penetration testing to verify that there are no unknown vulnerabilities and that the policies are being effectively enforced.

While this might sound clear in its direction, the problem remains of how black box testing will verify that all of the policies are being effectively enforced. This is particularly troubling when it becomes clear that it is the usual practice to have penetration testers blindly look for only one way into the system. How does "blind" black box testing verify the complete and correct implementation and enforcement of policy? How can organizations be sure that the testing is thorough given that the skill levels between penetration testers may vary greatly?

By its nature, black box testing does not and cannot accomplish this. This lack of effectiveness forms one of the main problems in conducting a thorough audit of any security system. Without focused penetration testing, there cannot be a thorough audit of an information system's security.

By adding this practical framework to the audit planning, security auditors are now able to standardize the process in which policies are specified more formally

within their system. This gives them the design tools for creating a database for focused penetration testing and the ability verify security policies within their organization.

4.2 Designing a Tool for Focused Auditing

The following Entity Relationship Diagram (ERD) outlines the design of the penetration-testing database that supports this framework. Based on the MLS directory scheme of the LDAP model, the database matches vulnerabilities to the permissions of object classification, categories, and users. By imputing the information flow policy and detailing all of the subjects' permission sets, auditors can query the relationship between subject and objects allowing them to see where the information flow policies should be enforced. The database can correlates the vulnerabilities associated with these relationships and directs the auditor to the appropriate penetration test to verify the policy's enforcement.

Organizations would need to input their own users, objects, and permission sets. After populating these entries, the organization would input all known potential vulnerabilities associated with the trust boundaries outlined within their system. These vulnerabilities could

be matched with the appropriate penetration testing method to ascertain whether, or not, the organization's security policies are being enforced. After building this database, auditors would then develop queries to look at subjects and their permissions related to the objects within the system.

Given the information provided in the query, auditors would have a list of testing to be done based on each user or groups of users that they examine. *Figure 16* examines the Entity Relationship Diagram (ERD) for a penetration testing database modeled after the example used in the previous chapters. This model could be applied to suit any organization using this framework.

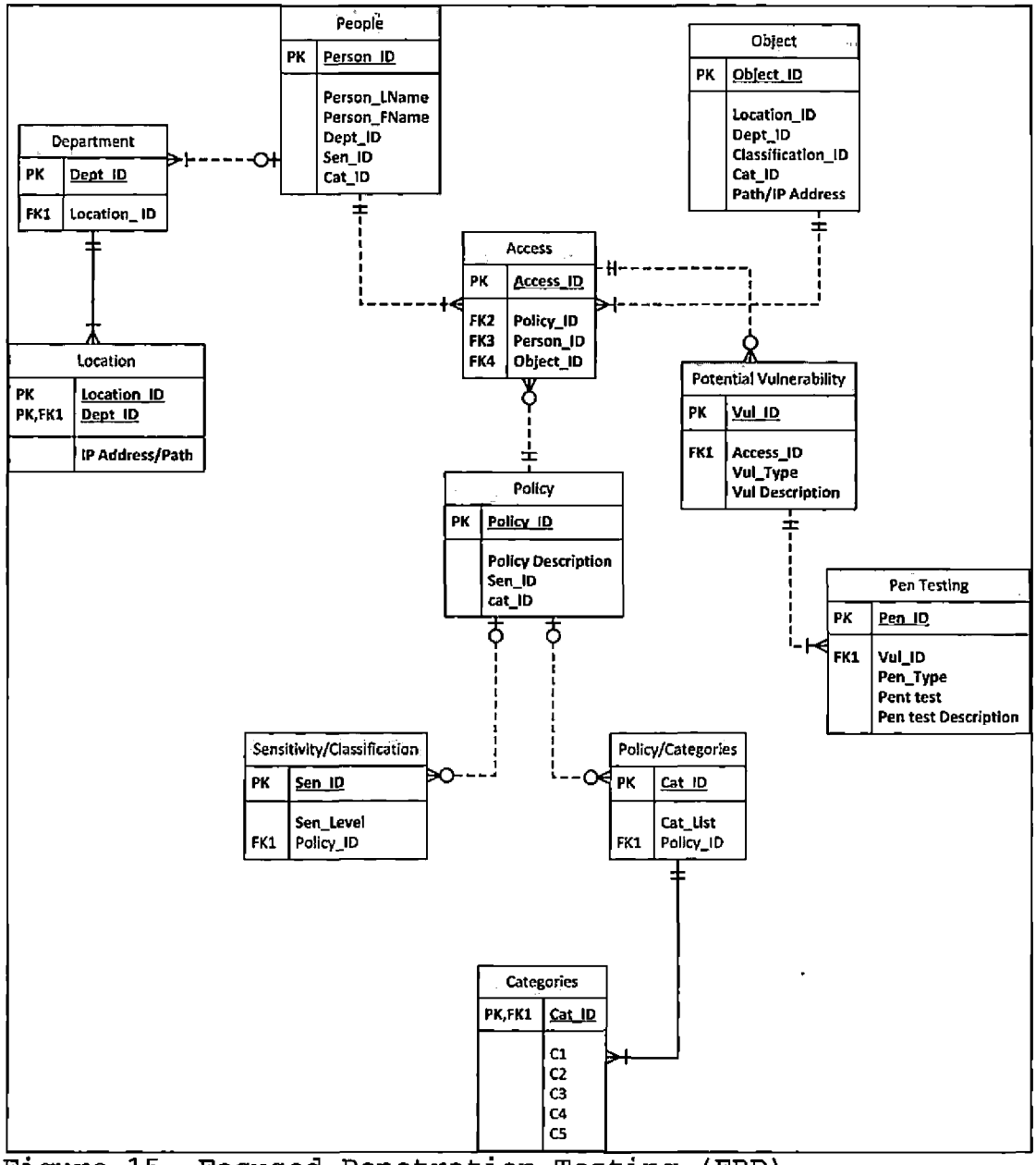

Figure 15. Focused Penetration Testing (ERD)

4.3 Database Entity Relationship Diagram

While the idea behind a penetration-testing database and its implementation can be complex, the design itself is simple. *Figure 15* illustrates that table People

connects to table Object when selecting an object, the query is sent to the bridge table Access in order to create a primary key using the Person_ID, Object_ID, and Policy_ID and then table Policy then pulls the sensitivity and categories of both the Person_ID and Object_ID, matching the sensitivity and category set of the Object_ID to the superset of the Person_ID. Once this is achieved, and confirmed, the Potential Vulnerability table creates a vulnerability ID from the Access_ID in the Access Table, listing the vulnerability type and description, showing the vulnerabilities associated with the subject, object, and permission sets allowed/not allowed. The Potential Vulnerability table then matches the penetration test types and lists the penetration tests for the potential vulnerabilities in the Pen Testing table. By using this tool, the auditor can query each user and their permissions on each object within the system to test for potential holes in the system. Organizations, if they prefer, can also apply role based permission sets instead of each individual persons in order to simplify the testing process.

CHAPTER FIVE

FUTURE WORK

5.1 Direction for Future Research

While this paper has helped to answer questions held by the author, it has also created additional ones, in particular, the application and proper use of categories. While a small question, it affects the use and control of attributes when designed into the information flow policy. This has become an important issue in access control and one that needs to be explored more fully.

The next step is to build a penetration testing Database based on something real. This is necessary to test the aptitude of the ERD design and application of the database in verifying security policies in a real world setting. In addition, automating the actual testing by creating scripts to run the tests, allowing the system to monitor itself constantly would be ideal. This would allow security policy people and system administrators to automate the policy configuration. In addition they would be able to standardize the security policy implementation process for devices on the network and also automate the process by which these policies are verified.

APPENDIX

INSTRUCTIONS FOR IMPLEMENTING THE CUSTOMIZED SCHEMA IN LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL

Instructions For Implementing The Customized Schema In Lightweight Directory Access Protocol

In this appendix, the following procedure is demonstrated for implementing custom shemas in the LDAP model(for complete LDAP installation and configuration instructions, please visit www.openlap.org):

1. Create a new custom schema called MLS.schema.

2. Based upon the MLS.schema, create a LDIF file (MLS1.lidf) and construct a new directory structure.

First, in order to create a MLS schema as shown in the following diagram, the MLS.schema contains two attribute defintions for sensitivity and categories and three objectClass definitions for MLSperson1, MLSperson2, and MLSperson3. Note that MLSperson1, MLSperson2 and MLSperson3 are sub-classes of Person, organizationalPerson, and inetOrgPerson object class, respectively.

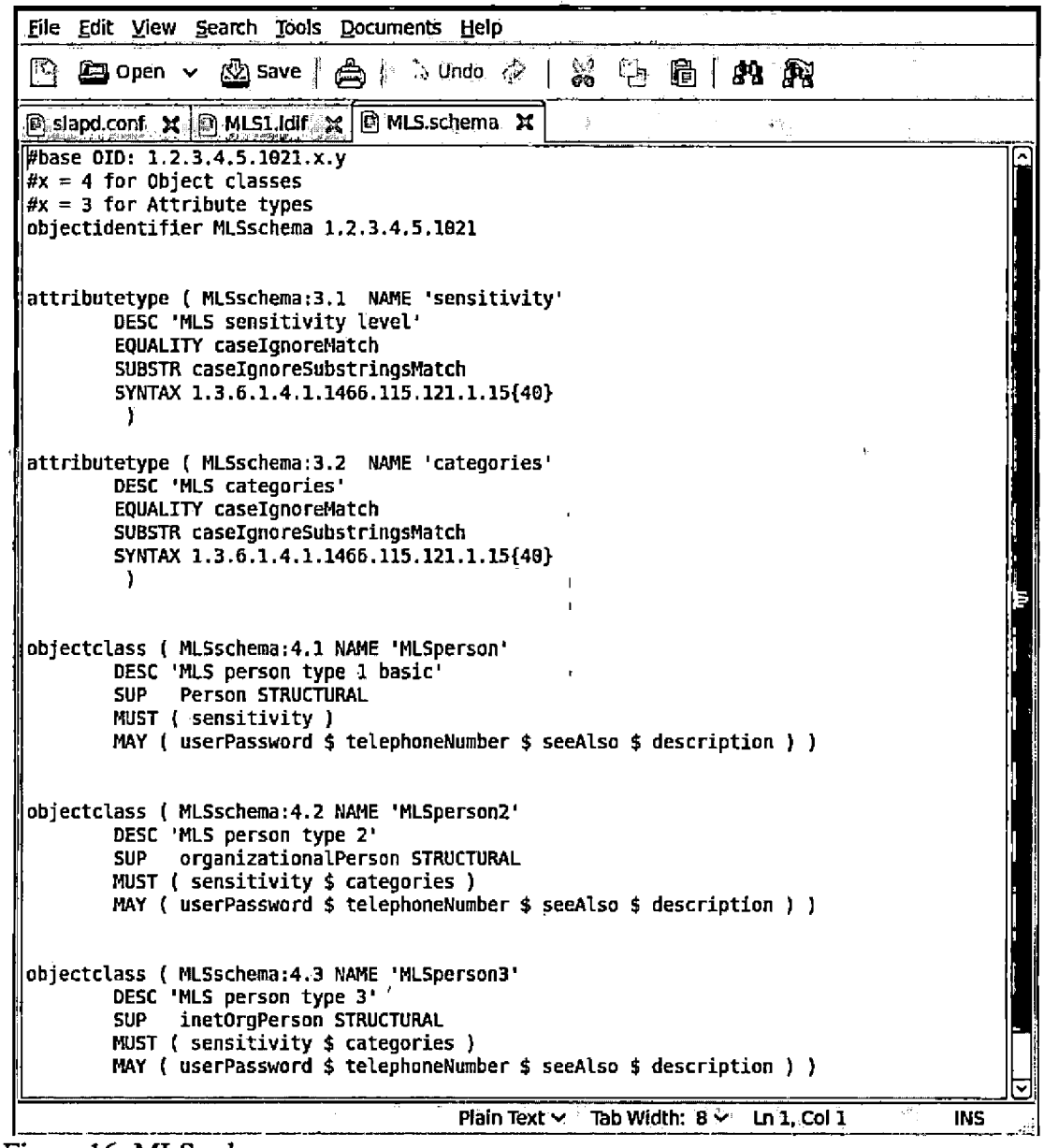

Figure 16: MLS schema.

Three objects (John, Betty and Jennica) of class MLSperson, MLSperson2, and MLSperson3 are created in the MLS1.ldif file.

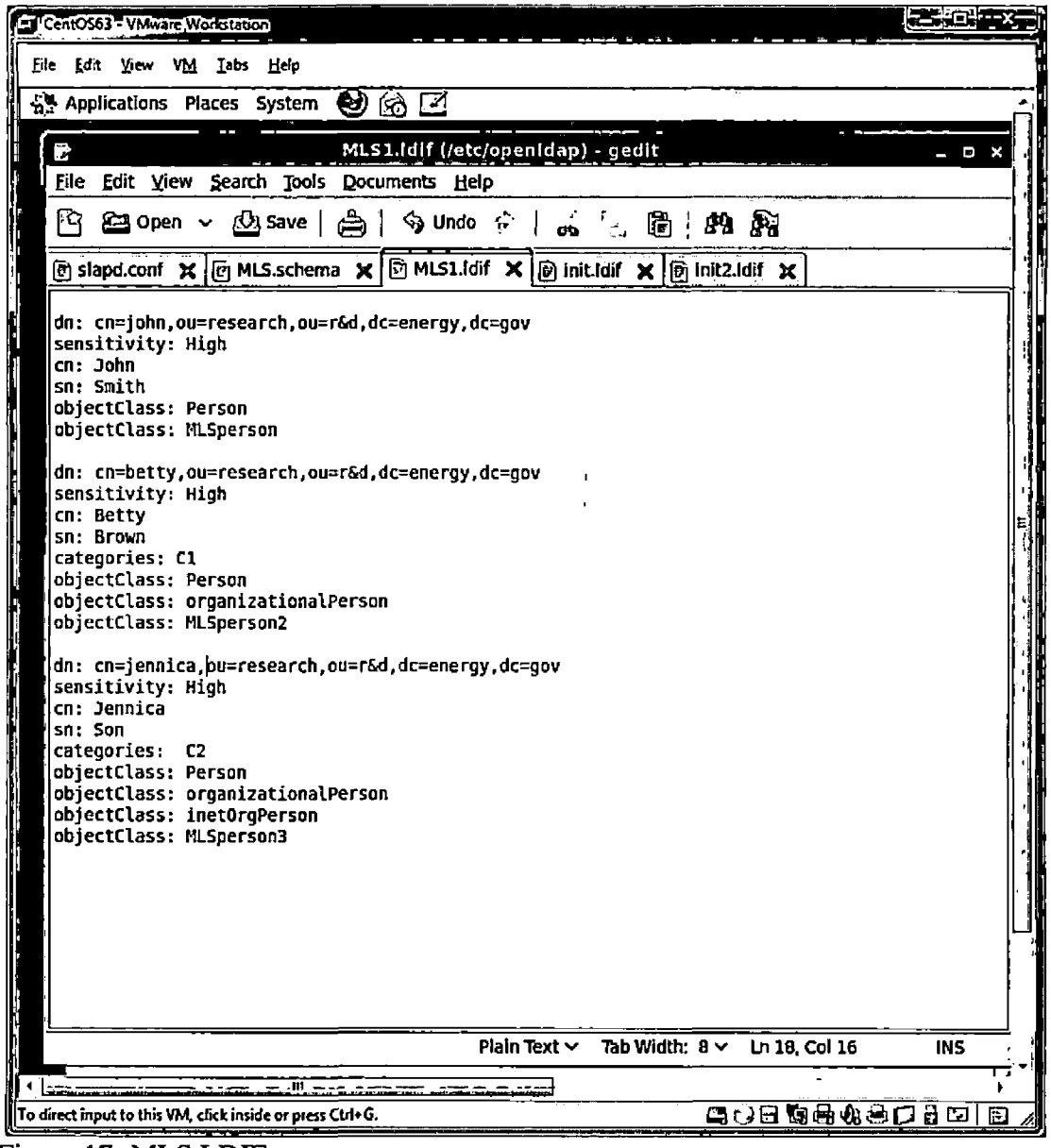

Figure 17: MLS LDIF.

The below diagram shows that three objects (John, Betty, and Jennica) are added to create a LDAP tree, using the ldapadd command with MLS1.ldif as an input.

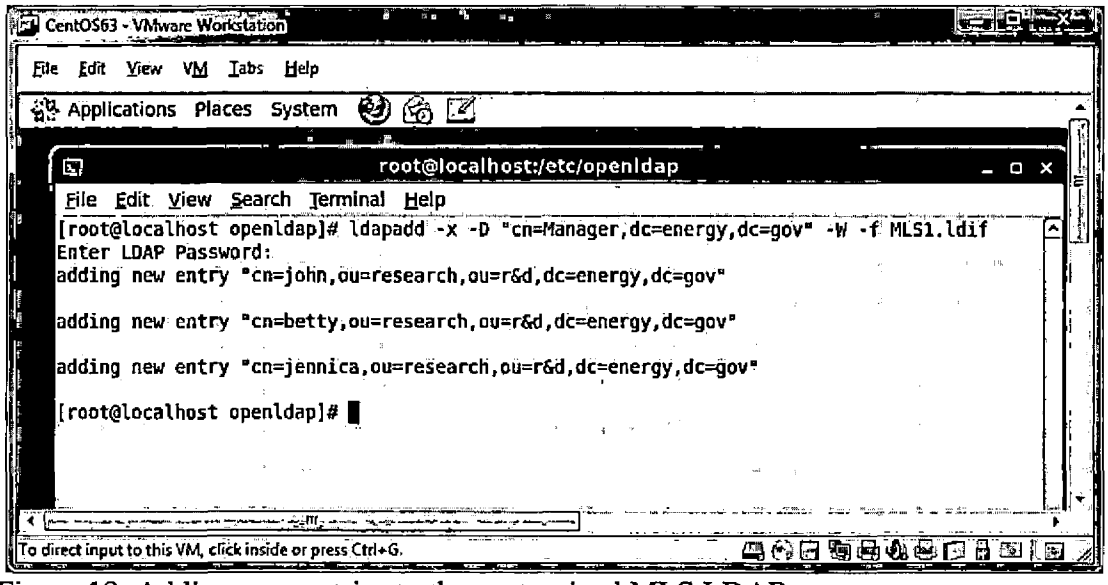

Figure 18: Adding new entries to the customized MLS LDAP.

The LDAP directory displays all the entries in the LDAP tree using the ldapsearch command as seen in *Figure 20.*

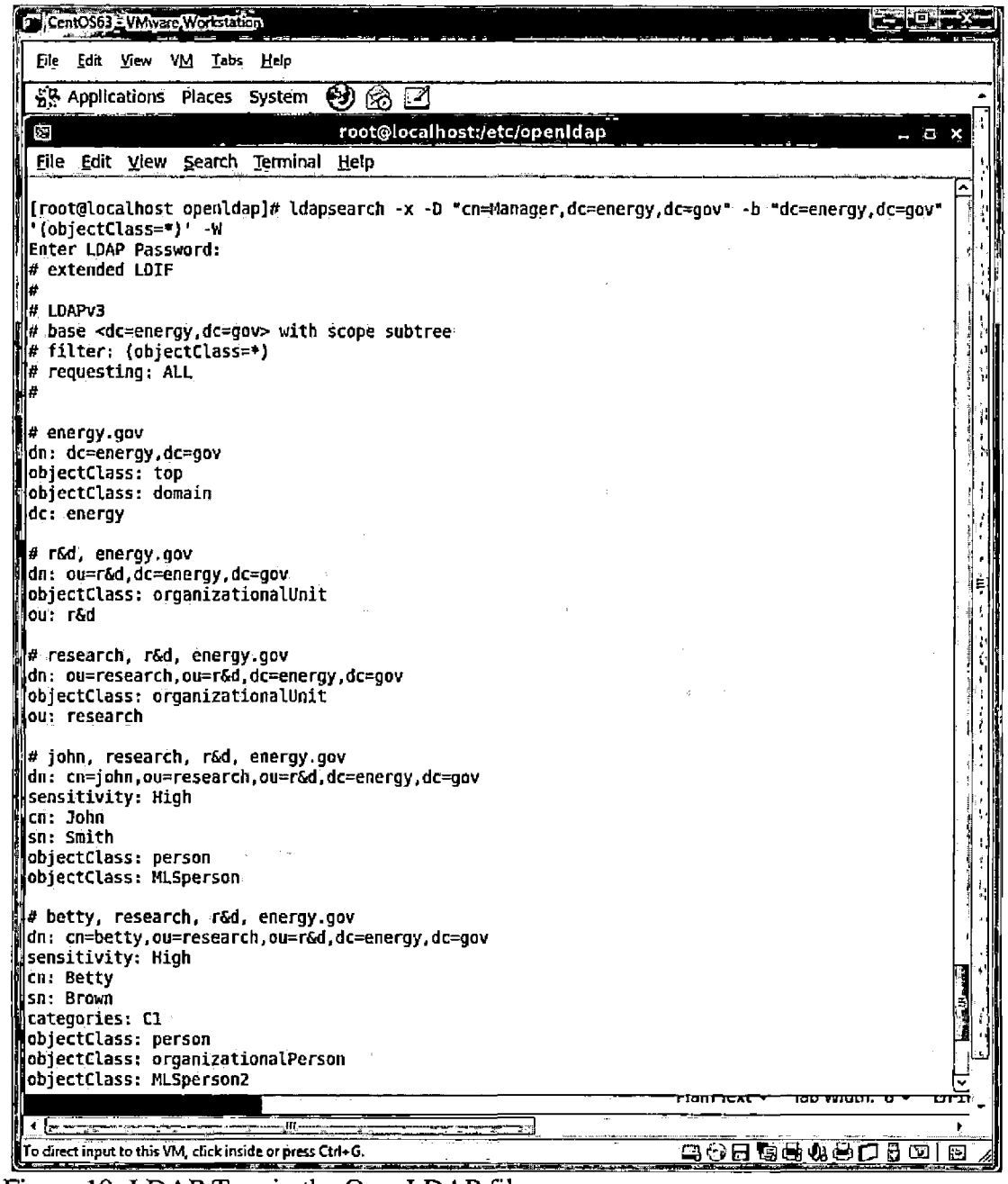

Figure 19: LDAP Tree in the OpenLDAP file.

The complete ldapsearch result is shown below in *Figure 21*.

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=energy,dc=gov> with scope subtree
# filter: (objectClass=*)
# requesting: ALL
#

# energy.gov
dn: dc=energy,dc=gov
objectClass: top
objectClass: domain
dc: energy

# r&d, energy.gov
dn: ou=r&d,dc=energy,dc=gov
objectClass: organizationalUnit
ou: r&d

# research, r&d, energy.gov
dn: ou=research,ou=r&d,dc=energy,dc=gov
objectClass: organizationalUnit
ou: research

# john, research, r&d, energy.gov
dn: cn=john,ou=research,ou=r&d,dc=energy,dc=gov
sensitivity: High
cn: John
sn: Smith
objectClass: person
objectClass: MLSperson
# betty, research, r&d, energy.gov
dn: cn=betty,ou=research,ou=r&d,dc=energy,dc=gov
sensitivity: High
cn: Betty
sn: Brown
categories: C1
objectClass: person
objectClass: organizationalPerson
objectClass: MLSperson2

# jennica, research, r&d, energy.gov
dn: cn=jennica,ou=research,ou=r&d,dc=energy,dc=gov
sensitivity: High
cn: Jennica
sn: Son
categories: C2
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: MLSperson3

# search result
search: 2
result: 0 Success

# numResponses: 7
# numEntries: 6
```

Figure 20: LDAP search results.

REFERENCES

Arkills, B. (2003). *LDAP directories explained. An introduction and analysis.* New York, NY: Addison-Wesley Press.

Bell, D. A. (1976). *Secure computer systems: Unified exposition and multics interpretation. Tech report. ESD-TR-75-306.* Bedford, MA: The MITRE Corporation.

Bell, D. E. (2005). *Looking back at the Bell-LaPadula model. Proceedings of the 21st annual computer security applications conference* (pp. 337-351). Reston, VA: ACSAC.org.

Bryans, J. (2005). *Reasoning about XACML policies using CSP. Workshop on secure web services* (pp. 28-35). Fairfax, VA: ACM.

Bryans, J. F. (2006). *Model based analysis and validation of access control policies. Technical-report series.* Newcastle Upon Tyne, England: University of Newcastle Upon Tyne Computing Science.

Cranor, L. D. M. (2006, November). *The platform for privacy preferences 1.1 (P3P1.1) specification. working group note.* Retrieved April 2013, from World Wide Web Consortium: http://www.w3.org/TR/P3Pp11/

Cranor, L. L. (2002, April). *A P3P preference exchange language 1.0 (APPEL1.0).* Retrieved April 2013, from World Wide Web Consortium: http://www.w3.org/TR/P3P-preferences/

Damianou, N. D. (2001). *The ponder policy specification language. Policy 2001: Workshop on policies for distributed systems and networks* (pp. 18-39). Bristol, UK: Springer.

Dardennen, A. V. (1993). Goal directed requirements acquisition. *Science of Computer Programming, 20,* 3-50.

Distributed Management Task Force. (1998). *Directory enable networks* (DEN) *initiative.* Retrieved from DMTF Distributed Management Task Force. http://www.dmtf.org/standars/standar_den.php

Ferraiolo, D. S. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security, 4*(3), 222-274.

Han, W. A. (n.d.). A survey on policy languages in network and security management. *Computer Networks, 56*(1), 477-489.

Hinchey, M. A. (1995). *Concurrent systems: Formal development in CSP.* New York, NY: McGraw-Hill, Inc.

Hu, V. H. (2007). *Conformance in XACML. 31st Annual IEEE International checking of access control policies specified Computer software and applications conference* (pp. 275-280). Beijing, China: IEEE.

Hu, V. K. (2011). Model checking for verification of mandatory access control models and properties. *International Journal of Software Engineering and Knowledge Engineering, 21*(1), 103-127.

Jackson, D. (2000). *Alloy: S lightweight object modeling notation. Technical report 797.* Cambridge, MA: MIT Laboratory for Computer Science.

Jamhour, E. (2001). *Distributed security management using LDAP directories. Proceedings of the XXI international conference of the chilean computer science society* (pp. 144-153). Antofagasta, Chile: SCCC.

Kagal, L. F. (2003). *A policy language for a pervasive computing environment. IEEE 4th international workshop on policies for distributed systems and networks* (pp. 63-74). Lake Como, Italy: IEEE.

Kalam, A. B. (2003). *Organizational based access control. Policies for distributed systems and networks, proceedings. IEEE 4th International Workshop* (pp. 121-131). Lake Como, Italy: IEEE.

Keliiaa, C. (2001). *Directory enabled policy based networking, SAND2001-2899.* Eubank, Albuquerque: Sandia National Laboratories.

Lampson, B. (1971). *Protection.* Princeton, NJ: 5th Symposium on Information Science and Systems.

Lobo, J. B. (1999). *A policy description language. 16th National Conference on Artificial Intelligence* (pp. 291-298). Orlando, FL: Association for the Advancement of Artificial Intelligence.

Lobo, J. E. (2009). *CIM simplified policy language DSP0231.* Portland, OR: Distributed Management Task Force, Inc.

Moses, T. (2005, February). *OASIS eXtensible access control markup language (XACML) version 2.0.* Retrieved April 2013, from OASIS Open: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

National Institute of Standards and Technology. (2006, September). *Publication NISTIR 7316.* Retrieved January 2013, from National Institute of Standards and Technology: http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf

National State Association and U.S.General Accounting Office. (2001). *The management planning guide for information systems security auditing.* Washington, D.C.: National State Association and U.S.General Accounting Office.

OpenLDAP. (2013, March). *OpenLDAP software 2.4 administrator's guide.* Retrieved March 2013, from OpenLDAP: http://www.openldap.org/doc/admin24/OpenLDAP-Admin-Guide.pdf

Son, J. (2008). *Covert timing channel analysis in MLS real-time systems.* University of Idaho.

Thomas, R. (1997). *Task-based authorization control (TBAC): A family of models fort active and enterprise-orientation management. 11th IFIP Working Conference on Database Security.* Lake Tahoe, CA: International Federation for Information Processing.

Wahl, M. H. (1997, December). *Lightweight directory access protocol (v3) (RFC 2251).* Retrieved February 2013, from The Internet Engineering Task Force (IETF): http://www.ietf.org/rfc/rfc2251.txt

Zeilenga, K. (2006, June). *Lightweight directory access contro (LDAP)1: Technical specifications road map (RFC 4510)*. Retrieved March 2013, from RFC Editor: http://www.rfc-editor.org/rfc/rfc4510.txt

NOTES

---

[1] National Institute of Standards and Technology is the federal technology agency that works with industry to develop and apply technology, measurements, and standards. For more information see http://www.nist.gov and http://csrc.nist.gov/\publications/PubsSPs.html

[2] Federal Information Security Management Act is a federal law enacted as title III of the E-Government Act of 2002. For additional information see http://csrc.nist.gov/groups/SMA/fisma/overview.html for additional information.

[3] Federal Information Processing Standard issued under the Information Technology Management Reform Act of 1996 began to issue standards and guidelines developed by the National Institute of Standards and Technology (NIST) for Federal computer systems. For additional information see http://www.itl.nist.gov/fipspubs/geninfo.htm

[4] Black Box Penetration Testing requires no knowledge of the system being tested and simulates the approach of an uninformed attacker, or hacker, attempting to breach the system.

[5] XACML stands for eXtensible Access Control Markup Language and is used in programming access control policies. The language is implemented in XML and used a process model to evaluate access control queries according to the information flow policy of an organization.

[6] The Ponder language provides a common means of specifying security policies that map onto various access control implementation mechanisms for firewalls, operating systems, databases and Java (Damianou, 2001).

[7] A conceptual model used in project management through which a new information system develops from the initial feasibility study and Design stage of the new system to maintenance of the completed system.

[8] Trust boundaries, according to Microsoft, are determined by identifying whether an asset's upstream data flows, or user input, are trusted or not and the method used to authenticate, or authorize, these data flows, or user inputs, if they are not.

[9] Multi-Level Security is the application of security controls in a computer system to restrict the access of information/resources/assets based on security clearance/sensitivity (i.e. top secret, secret, etc…) and the need to know level (categories) of people using the system.

[10] NISTIR 7316, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930 September 2006. http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf

[11] Bell-LaPadula Model is used to define access control in government and military organizations and was developed by David Elliott Bell and Leonard J. LaPadula, to establish U.S. Department of Defense (DoD) Multi-Level Security (MLS) policy.

[12] An information flow policy defines the different classes of information in an agency that can exist in a system and how information flows between them.

[13] The lattice based security model (access control model)is based on a complex interactions between any combination of objects (assets or resources) and subjects (people or groups of people).

[14] Mandatory Access Control (MAC) is a security mechanism that restricts the level of control that users (subjects) have over the objects that they create. Unlike in a Discretionary Access Control (DAC) implementation, where users have full control over their own files, directories, etc., MAC adds additional labels, or categories, to all file system objects. Users and processes must have the appropriate access to these categories before they can interact with these objects. Source: http://www.centos.org/docs/5/html/5.1/Deployment_Guide/sec-mac-intro1.html

[15] X.500 data model has been adopted by the LDAP protocol. For additional information please refer to RFC 2251.

[16] For more detailed information regarding attributes, please refer to RFC 2251.

[17] Newer versions of CentOS have changed the method of implementing OpenLDAP; please refer to the website www.openldap.com for updated instructions.