A NEW APPROACH TO DYNAMIC INTEGRITY CONTROL

A dissertation submitted
to Kent State University in partial
fulfillment of the requirements for the
Degree of Doctor of Philosophy

by

Talal S Albalawi

May 2016
© Copyright
All rights reserved
Expect for previously published materials
Dissertation written by

Talal S Albalawi

B.S., Al-Imam Muhammad Ibn Saud University, 2004

M.S., Old Dominion University, 2009

Ph.D., Kent State University, 2016

Approved by

__Austin Melton_______, Chair, Doctoral Dissertation Committee
Name

__Johnnie Baker_______, Members, Doctoral Dissertation Committee
Name

__Michael Rothstein____
Name

__Kambiz Ghazinour Naini
Name

__Stephen Gagola_______
Name

__Alan Brandyberry_______
Name

Accepted by

__________________________, Chair, Department of Computer Science
Javed I. Khan

__________________________, Dean, College of Arts and Sciences
James L. Blank
Abstract

Proper access control is one of the most important issues in computer security. It consists of securing a system in the form of availability, confidentiality, and integrity. Integrity is about making sure that only proper modifications take place. Some integrity models are static in nature, which may limit their capabilities for better protection of a system. In some cases like in the collaborative authoring systems (e.g. Wikipedia), such static models are not desired because there is a need for continuous evaluations of posted work. This motivated us to present a dynamic integrity model based on a metric we call the modification factor to evaluate whether the integrity level should be changed up or down. Furthermore, our dynamic model allows us to establish a level of trustworthiness that an entity has as a source or destination of information.
TABLE OF CONTENTS

ABSTRACT .................................................................................................................. iii
LIST OF FIGURES ........................................................................................................ vii
LIST OF TABLES .......................................................................................................... ix
ACKNOWLEDGMENTS ................................................................................................. x

CHAPTERS

I. INTRODUCTION

1.1 INTRODUCTION .................................................................................................... 1
1.2 ORGANIZATION .................................................................................................. 4
1.3 ACCESS CONTROL ............................................................................................. 5
1.4 ACCESS CONTROL MATRIX .............................................................................. 7
1.5 MANDATORY ACCESS CONTROL ..................................................................... 8
1.6 DISCRETIONARY ACCESS CONTROL .............................................................. 11
1.7 ROLE BASED ACCESS CONTROL ..................................................................... 11
1.8 INTEGRITY .......................................................................................................... 16
1.9 INTEGRITY THREATS ......................................................................................... 18
1.10 THE BIBA MODEL ............................................................................................. 20
1.11 ACCESS POLICIES BASED ON BIBA ............................................................ 21
1.12 INFORMATION FLOW ....................................................................................... 24

II. PROBLEM DEFINITION AND LITERATURE REVIEW

2.1 PROBLEM DEFINITION ....................................................................................... 26
LITERATURE REVIEW

2.2.1 CHINESE WALL MODEL

2.2.2 CLARK-WILSON MODEL

2.2.3 A BIBA MODEL WITH DYNAMIC CHARACTERISTICS

2.2.4 INTEGRITY-ORBAC MODEL

2.2.5 TOTEL MODEL

2.2.6 MULTILEVEL-ORBAC

2.2.7 SELINUX

2.2.8 TACSIF MODEL

2.2.9 TAAC

2.2.10 A DYNAMIC TRUST MODEL FOR MOBILE AD HOC NETWORKS

III. THE MODEL

3.1 INTRODUCTION

3.2 MAC VS DAC

3.3 ACCESS MODES

3.4 THE PROTECTION POLICY

3.5 A DYNAMIC INTEGRITY MODEL

3.5.1 ASSUMPTIONS

3.5.2 READ ACCESS

3.5.3 WRITE ACCESS

IV. IMPLEMENTATION

4.1 THE IMPLEMENTATION
LIST OF FIGURES

Figure 1. The Relation between a User and Permissions via Roles................................. 14
Figure 2. An Overview of the Model Structure................................................................. 52
Figure 3. The Direction, of Flow of Information for Read and Write............................ 53
Figure 4. The List of Project Tom is Involved with......................................................... 65
Figure 5. Tom Invites and Sets the IL and MF for a Member via Email.......................... 66
Figure 6. Mike IL and MF Levels along with the Documents in Project Accounting
Application.......................................................................................................................... 66
Figure 7. Tom Assigns a Task to John................................................................................ 67
Figure 8. The List of Tasks Assigned to John.................................................................... 67
Figure 9. Only Tom the Manager can only Create and Delete the Tasks......................... 68
Figure 10. John is Warned not to Proceed Reading an LQR Document Created by Mike.... 69
Figure 11. John’s Current IL 5 and the Access Modes Next to Each Document.............. 70
Figure 12. John’s IL Changed to One Level Down After Multiple LQR............................ 70
Figure 13. Alice’s MF Reduced by One due to HQR......................................................... 71
Figure 14. Alice’s IL Changed One Level Up...................................................................... 72
Figure 15. Document Created by Bob is Modified by John and the MF is Reduced by One. 73
Figure 16. Sign Up Form to Create an Account................................................................. 92
Figure 17. Login Page......................................................................................................... 93
Figure 18. Creating a New Project...................................................................................... 94
Figure 19. Account Information Update............................................................................. 95
Figure 20. Requesting Account Password via Email......................................................... 96
Figure 21. The manager creating a task for a team member.............................................. 97
Figure 22. List of tasks assigned to John by Tom ................................................................. 98

Figure 23. List of projects that Tom is involved with ......................................................... 99
LIST OF TABLES

Table 1. Example of Access Control Matrix................................................................. 8
ACKNOWLEDGEMENTS

During my journey as a PhD student at the Computer Science department, Kent State University I have met people both faculty members and students who I am honored to know. It was a journey that took me couple of years of hard work to reach this day. The day where I successfully defended my dissertation. Along this journey I had the privilege to work with professors that made it possible.

First, my advisor Dr. Austin Melton. Thank you for your time, mentoring, patience and believing in me during this journey. I also would like to thank our team members my co-advisor, Dr. Michael Rothstein and Dr. Kambiz Ghazinour Naini, for the time they spent and the work they did to make this possible.

I also would like to thank my committee members, Dr. Johnnie Baker, Stephen Gagola and Dr. Alan Brandyberry, for taking the time and feedback to guide me during this process.

I also would like to express my gratitude to Dr. Jennifer Kulics the Dean of Students for supporting and guiding me when I needed during this journey.

To my family: my dad, Saad, mom, Fatmah, my sisters, Manal, Ohoud and Sarah my brothers, Mushari, Rakan and Salman. My Father-In-Law, Fahad and my Mother-in-Law, Maryam. This would have not been possible without your support and prayers.

Last but not least, my beloved wife, Tahani and our little princess Sulaf. This work is not possible without your support, patience and love. You gave me the push that I needed to achieve this goal. Thank you for both of you.
CHAPTER 1

Introduction

1.1 Introduction

Security issues in computing affect daily users. These issues can have a huge impact on a person’s finances, identity, and privacy. If a user has their password and username stolen, then the thief can use this information to access the victim’s account compromising their confidentiality. However, what happens if someone changes content without permission or changes content by accident. This will compromise the integrity of the data, which could lead to the spread of false information damaging anything that uses that falsified data. For example, if a patient is given a certain treatment using 10 milligrams of a certain drug but someone who has access to the patient electronic medical records changes that intentionally or accidentally to 100 milligrams, then that patient might lose their life due to using a stronger dose of that drug. So, in this case, the integrity of the electronic medical record was compromised leading to a chain of reactions with a fatal ending. Another real world example is the Stuxnet worm that targets control systems using Windows operating system. It reports data that is fabricated to control system causing it to react based on the provided data in a destructive way affecting production and damaging equipment. As presented in the previous examples the protection of integrity is as important as the protection of confidentiality.

For some time the research focus was on confidentiality; however, integrity related issues especially in commercial use made it a focus again. So the question is what do we mean by data
According to [24] integrity has multiple definitions based on the profession, and they are:

- A security officer: “Data integrity means data cannot be modified undetectably. From the perspective of data and network security, data integrity is the assurance that information can be accessed or modified only by those authorized to do so”.

- A database administrator: “Data integrity means that data entered into a database are accurate, valid and consistent”.

- A data architect or modeler—“Data integrity means that primary entities should be unique and not null. The uniqueness of the entities within a data set means that there are no duplicates within the data set and that there is a key that can be used to uniquely access each entity within the data set.”

- A data owner (i.e., the subject matter expert)—“Data integrity may be a measure of quality as it ensures that there are appropriate business rules that define the relationships between entities and that these provide validation mechanisms such as testing for orphaned records.”

- A vendor—“Data integrity is accuracy and consistency of stored data, indicated by an absence of any alteration in data between two updates of a data record. Data integrity is imposed within a database at its design stage through the use of standard rules and procedures and is maintained through the use of error checking and validation routines.”

- An online dictionary—“Data integrity is the Quality of correctness, completeness, wholeness, soundness and compliance with the intention of the creators of the data. It is achieved by preventing accidental or deliberate, but unauthorized, insertion, modification or destruction of data in a database. Data integrity is one of the six fundamental
components of information security.”

From the previous definitions, we can see terms like quality, completeness, correctness and modification. In addition, in some literature the term trust is also mentioned as integrity which reflects how much the data can be trusted as a source of information. This research is focusing on the definition from a security officer point of view. We are going to present more integrity definitions later in this section based on an information security approach.

Having decentralized information systems and also the availability of strong programming environment for end users have created potential integrity vulnerabilities because compromised data are used in decision making without the consideration of the data quality. Is integrity an information security issue alone? Integrity can be investigated as an information security issue or a software quality problem or a business intelligence issue. In our research, we view the integrity problem as an information security issue.

Many of the integrity models are static in nature, meaning that no change in integrity levels is allowed during the operation of the system. A few models are dynamic in nature, and that allows for such changes to take place. Static integrity models have the benefit of reducing the overhead involved in data integrity protection. Dynamic integrity models even when complicated do offer better security protection especially when running dynamic applications where the freshly generated code is evaluated. The dynamic integrity protection is becoming more important with the growing use of web browsers as application platforms because the client side running language is JavaScript, which is a highly dynamic language.

We think with the increase of distributed systems and web services also the commercial requirements need something in the form of dynamic change to different integrity levels, and this
is the focus of this dissertation. The dynamic change will allow us to measure the trust level a given entity has in the system as reflected by its integrity level. Integrity is part of the problem related to access control. In the first parts of this chapter, we will present access control and an access control matrix. We then present the MAC, DAC and RBAC models followed by a detailed description of integrity and threats to integrity. In addition we introduce the BIBA integrity model, which we use in our work.

1.2 Organization

In addition to chapter 1, there are five chapter that are organized in the following structure:

- Chapter 2 presents the problem we address in this dissertation. We also present the literature review of some of the known integrity models.
- Chapter 3 introduces our dynamic integrity model in detail description.
- Chapter 4 presents the implementation of our model.
- Chapter 5 introduces an extension to our generic model using a Role Based Access Control approach.
- Chapter 6 we present our findings and conclusion also we present the future work.
- Appendices provides a list of access control related definitions. We also provide figures of our implemented model. A list of used abbreviations used throughout this dissertation is also provided in the appendix section.
1.3 Access Control

Access control is one of the oldest problems in computer security. Its primary focus is to control or restrict access to a system or resources by allowing or rejecting the access request either to observe or modify an entity inside the system. Computer security is concerned with three aspects; these are Confidentiality, Integrity, and Availability. Availability focuses on making sure that the system is usable and available for use. An example of availability would be a denial of service attacks. Under such attacks, the attacker wants to prevent users from using the system, making it unavailable or difficult to access. Confidentiality prevents unauthorized disclosure of information, and integrity prevents unauthorized modification to protected information. An example of such attributes is a banking system. Confidentiality is concerned with blocking an unauthorized user from seeing customers’ account balances. Integrity is to protect against a change to an account balance by such users, and availability is the assurance that the system will function on time as needed.

In access control, there has to be a system for verifying authorized and unauthorized access. In confidentiality, for instance, we make sure that only the appropriate persons can have access to the appropriate level of information via enforcing security level checks. This was a need in the military establishment. Integrity was driven by needs in the business world. There is a need to protect the content from unauthorized modification, and to make sure that any allowed modification does not compromise the system’s secure status or affect the quality of the data. The way to enforce such mechanisms is to use of a security policy. A security policy has to protect one of the following [1]:
• Confidentiality: protect the system from unauthorized disclosure of information.

• Integrity: protect the system from unauthorized modification to the information.

The reason for having the security policy protecting only one of the two controls is the difficulty of providing a single policy to protect both at the same time. This is they differ in nature, which makes it difficult to bring them both under one policy. Under confidentiality, for instance, a user would have to have a given security level to allow them to read information on their level and anything lower than that level. For writing, they would have to write to higher levels because the goal is to protect information from improper disclosure. This was a need in the military, and it fits their work requirements. However, integrity is the dual of confidentiality, and it is the opposite when it comes to the flow of information. A user with a given integrity level can read from higher integrity level objects, and write to lower integrity level objects. It is clear that the flow of information is opposite in the two models. In addition, integrity is driven from the commercial world to make sure that improper modification to information is not authorized and the data quality is preserved.

In [3] the integrity includes the integrity of the source of the information and the integrity of the data itself. The integrity of the source of the information has an impact on how people can trust that source with its information. If the source is highly reliable and trustworthy, then the information is expected to be of higher quality and more reliable. If the source is not trustworthy, then its information is not of good quality. So the integrity of the source of information impacts how people view that information as reliable or not.

The protection of the data itself from corruption and unauthorized modification is a
requirement of integrity. Also since data at higher levels are assumed to be of higher quality, this leads to a subject read from higher integrity level objects and modify lower integrity level objects. It is the opposite of confidentiality because integrity includes both correctness and trustworthiness and it is assumed that the higher the integrity level of the subject the higher the data quality it has.

In access control, we have two main access control models: [1] MAC and DAC. MAC is Mandatory Access Control and DAC is Discretionary Access Control. Also a generalized form of both MAC and DAC models is a model called Role-based Access Control RBAC. We provide an overview of all models in a later section of this chapter. We now present the Access Control Matrix.

1.4 Access Control Matrix

Each system has a state that is according to [3] “The collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system.” The protection state is the subset of this collection that deals with its protection.

“The Access Control Matrix is a tool that is used to describe the current protection state of a certain system [3].” It is the most precise and reliable model that can accurately describe a system protection state. Each system can be in a set of authorized protected states. Whenever a system is in any state other than those that are listed, then the system is not in a protected state. The Access Control Matrix shows those rights that a subject has over every other entity in the system.
Butler Lampson proposed this model back in 1971 to show the protection state by describing the rights a user has over a file or set of files. Entities that can compromise the protection state of a system are called a set of objects. The set of entities that are active and perform tasks on objects are called subjects. An example is [3]:

<table>
<thead>
<tr>
<th>Process 1</th>
<th>File 1</th>
<th>File 2</th>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Read</td>
<td>Write, execute</td>
<td>Read, write, own</td>
<td>Read</td>
</tr>
<tr>
<td>Process 2</td>
<td>Write, own</td>
<td>Read</td>
<td>Write</td>
<td>Read, write, own</td>
</tr>
</tbody>
</table>

Table 1 Example of Access Control Matrix

This shows the protection state of a system. However, what it exactly means differs from one system to another. It depends on the internal definitions of various tasks done. For instance, reading a file or writing to a file is clear. However, what does it mean to be reading from one process to another? It might mean accepting messages, and it might mean that whatever process 2 reads can be read by process 1, even when process 1 has no direct access. So what can be done has to be defined in terms of actions done from process to file and from process to process. Next, we present the models of MAC, DAC and RBAC that are the types of access control.

1.5 Mandatory Access Control (MAC)

The access control model MAC controls access by assigning different security levels to subjects and objects, and by protecting the disclosure of information by limiting access between different levels. It enforces the security policy when granting a set of actions to a subject over an object or set of objects. The two known examples associated with MAC are the Bell-LaPadula confidentiality model and the Biba integrity model. A MAC security model handles one of the
two individually because they cannot be combined into a single model. We next provide a brief
description of both models but first we use the following terminologies to describe each model:

- $s$: A subject, which is an active entity.
- $o$: An object, which is a passive entity.
- $il$: An integrity level.

First is the Bell-LaPadula confidentiality model (BLP). Introduced back in 1973, it has
security divided into levels. In this model, subjects and objects are assigned different labels
reflecting their levels. It uses the following controls (properties) to grant or deny access requests:

- The simple security property: A subject $s_i$ can read from an object $o_i$ only if the security
  level of $s_i \geq o_i$.
- The Star (*) property: A subject $s_i$ can write to an object $o_i$ only if it has a security level
  $s_i \leq o_i$.

So the flow of information is from the lower to higher. The labels allow the model to enforce
the properties. BLP features what is known as the tranquility feature. The tranquility feature
states that the subject or object label does not change while it is referenced. This feature can be
in the form of strong or weak. In strong tranquility, it is in restrict mode meaning no change to
labels can happen during the system operation to prevent any compromise from happening. The
other is weak tranquility, which allows for the changing of the levels of subjects to allow them to
have higher access, but it has to be done in a manner that maintains the system’s overall security.
Second is the Biba integrity model, which focuses on the integrity requirements. BLP was driven by military requirements, but Biba was based on the requirements from the commercial world. Biba uses the following properties:

- The simple security property: A subject $S_i$ can read from an object $O_i$ only if it has a integrity level $S_{il} \leq O_{il}$.
- The Star (*) property: A subject $S_i$ can write to an object $O_i$ only if it has a integrity level $S_{il} \geq O_{il}$.

Biba is the opposite of BLP. The flow of information is from higher to lower. Having Biba based on MAC made it fall short from achieving its goals. This is because MAC had the problem of confidentiality as its primary focus.

The MAC model [1] is considered an easy and straightforward model. It best suits implementations in the commercial world. Systems using MAC are protected against attacks like TROJAN horse attacks because users cannot, for instance, declassify information. However, MAC is not without limitations. It places restrictions on the users of the system through the used security policy. It makes it difficult to dynamically modify the policy; thus, this can limit the productivity. It may also require the operating system to be rewritten just to place trusted components of the system outside the access control framework. It is considered expensive and difficult to implement.
1.6 Discretionary Access Control (DAC)

The access control model DAC [1] uses matrices to determine if a subject can access a given object in a given system. It is a matrix having its rows as subjects and columns as objects. The mapping of subjects to objects shows the rights a subject has over an object. In addition, DAC is not like MAC that uses a security policy to restrict access. Instead, it allows the subject that owns an object to decide the access rights that can take place on that object. As an example, in UNIX files system the owner decides who can access and who cannot access the file. Also, they can decide if they have a read or write access.

DAC is considered easy to use and implement, compared to MAC. It provides better control over system objects. However, it is not without limitations. When we allow the users to have control over the objects, we make the entire system vulnerable to TROJAN horse attacks. Also, the maintenance and verification of the used security principle is considered extremely hard, due to having the control handed to users over objects that the user own.

1.7 Role Based Access Control (RBAC)

Another model, which is a generalized form of MAC and DAC, is the Role Based Access Control (RBAC) [1]. It addresses the security issue from integrity first, then addresses confidentiality as a second approach. It uses both MAC and DAC, but it defines the rights inside the system based on role or position, not the individual using the system. For example, a secretary would have defined roles inside the organization. Those roles are translated to a set of rights of access inside the system.
RBAC is based on commercial world requirements. Many of the early access control models as mentioned earlier did focus on military requirements. RBAC is one of the early models to address the requirements of commercial environments. The way RBAC handles access according to [20]: “Access to Computer System objects is based on a user’s role in an organization”. In a business environment, we can see that employees have roles, and with such roles they have certain privileges associated with them. This is to make it manageable to identify what an employee can do to achieve the assigned task or tasks based on the role and what access rights they have.

Controlling roles is much easier than ACL (Access Control List). An ACL would require setting the rights of access for each individual user, and this could be a significant overhead if the organization has a large number of employees. For example, suppose we have a shared printer and we have multiple departments accessing that printer. If we then decide to block a certain department from accessing that given printer, under ACL we have to do it for each individual employee in that department. If we have an RBAC system, used then we only have to disable the roles of that department from accessing that printer this is an easy and more manageable approach than the ACL.

A role is viewed as a job or position [20] within an organization. This role is a separate structure regardless of the user. In early research into roles, it was recognized that roles do support the principle of least privilege. Under least privilege, an entity is given access to least amount of rights allowing it to do its tasks. So the role is created with minimum permissions allowing that role to accomplish the tasks associated with it.

Having a DAC model does not provide sufficient level of security since it gives the user (owner) the options to decide who should access their objects. This is a problem since it does not
fully protect against users who have the right to access the object and then give information from that object to another user who does not have the right to access such object. A MAC model tends to focus its protection on confidentiality first. So integrity is not a focus, which is problematic in a commercial environment. RBAC is viewed as a general format of both MAC and DAC to address the commercial requirements.

In the research presented by Ferraiolo and Kuhn [20], they present three basic rules that are required for a role:

1. Role assignment: A subject is allowed to execute a given transaction only if the subject is assigned to a role. Authentication is not a transaction, and all users who carry out a transaction must have a role assigned to them.

2. Role authorization: An active role must be authorized for the user to carry out transactions.

3. Transaction authorization: A subject can execute a transaction only if it has an active role that is authorized to perform such transaction.

So we can see that a role is a collection of permissions, and the only way for a subject to perform a transaction is via roles. In figure 1 we see how the relation between permissions and users connect via a role.
Now the question becomes what is the difference between groups like in UNIX operating system and roles? In group based systems, a user can access an object by being part of a group or the user ID. However, if we remove the permission for the group, the user might still have access based on its user ID that is viewed as a weakness in a group based approach. Under roles the same situation is not possible because the user needs an authorized role to carry out a transaction. In the same paper [20] presented by Ferraiolo and Kuhn, roles are hierarchal which means that roles can inherent permissions from other roles.

Many researchers including the famous Ravi Sandhu did point out that an RBAC system is more flexible and customizable leading to a better security protection when compared with other models like the Clark-Wilson model.
These are some of the access control models used. The focus of this dissertation is going to be to introduce a model of dynamic integrity. This model will allow changing the level of integrity dynamically, using a factor called the modification factor.

Many researchers view integrity as a level of trust. A subject that has a high level of integrity also is highly trustworthy as a source and destination of information inside the system. And just as humans view trust, for instance, if you know someone who is trustworthy and that person gave you a piece of information, then you are expected to believe that. But if you happen to be given any piece of information from a person who is known not to be trustworthy, then he/she will be disbelieved most of the time until they start to build a certain level of trust and it would take time for them to do this.

We think the same thing should be done inside a system. A subject must start to build their level of trust by reading information from highly trusted objects. Then can they start to increase their integrity level by increasing their level of trust as they progress in the system.

A subject is an entity in a system requesting an access to an object that could be a file or part of the system. Having a dynamic model has been proposed in different policies, as we will see in a later chapter.

An example of dynamic integrity is a policy called the low water mark policy [3]. The idea of this policy is to change the subject’s integrity model each time it accesses an object with a low integrity level. The only problem is that this policy may downgrade the integrity level with no possibility for it to go back up again. Some may argue that there are benefits of such policy; this could be true in some cases however the vast majority of researchers do point out that this is
a problem for this policy to never recover from lower levels preventing subjects from ever reaching higher good quality data at higher integrity levels.

This, among other reasons, motivated us to develop a dynamic integrity model based on a metric we call the modification factor. We use this factor as a metric to decide when a change in integrity level can take place either by going up or down. A detailed explanation is provided in Chapter 3 of this dissertation.

Next we present a detailed description of integrity, what we mean by integrity, and issues related to integrity in addition to some known policies.

1.8 Integrity

Integrity was driven from the requirements in the commercial world. Confidentiality was introduced to handle military requirements, but it did not take into consideration the need to protect the changes happening to data, especially the changes without authorization. According to Biba [4], “a subsystem is said to possess the property of integrity if it can be trusted to adhere to a well-defined code of behavior.” So the developer of the system expects their design to function as it was intended. If anything unintended happens, then it is said that the system integrity is compromised. This can include extended subjects and objects. If a subject is not supposed to access a certain object, then it should not. Otherwise, it will be considered as an attack on that object.

The definition of integrity may be different from one research study to another and as we presented at the beginning of this chapter, even the type of profession may play a role in what we mean by integrity. We did say that we are following a security officer’s approach when defining
integrity, and we have multiple definitions based on this approach. According to [5] integrity may be defined in five ways. These are based on:

1. Quality: The data has a quality that meets or exceeds the quality requirements set by the user or owner.

2. Modify: Integrity is the ability to protect against unauthorized modification of data by an entity that is not allowed to access an object.

3. Modify: Integrity is the ability to protect against improper modification of data by an entity that is allowed to access an object, but conducts the modification in a wrong way.

4. Information Flow: The ability to protect against unauthorized modifications of data by restricting the flow of information.

5. Network: The ability to protect and detect unauthorized modification of data in a network.

The definition we adopt is as presented in [3]. Integrity is defined as “The trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change to data”.

This definition covers all the previous definitions, and we view it as a generalized definition of all five. It does cover, improper modification, quality, trustworthiness and information flow. So based on this, we come up with the following definition: Integrity is the protection against unauthorized or improper modification by an entity wanting to modify an object. This is due to the entity not having the required integrity level to allow it to perform the modification or because it is trying to maliciously modify an object that it should not modify.
We use this definition as the basis for our model in Chapter three of this dissertation.

Many of the proposed models in integrity are driven or based on the Biba model. When a subject wants to access a certain object, then the intent is to request a read or write access or both. We will look inside the Biba model to see how it handles both requests in this chapter.

1.9 Integrity Threats

Integrity threats could cause a change in any part of the system [4] (e.g. subject or object) without the approval of such change by the policy enforcer. The change can take the form of modifying an object by writing to it from a subject with the intent to corrupt data and then infect the system with false information. This form of change is direct since the subject interacts directly with the object. Here the subject, in this case, the attacker, wants to infect the system intentionally; however, there are cases when integrity can be affected accidently like in the case of storage. Storing data could be corrupted by having a hard disk crash, for instance, affecting the integrity of the data stored. Our focus is going to be on the first part, that is, the threats that are related to modifying an object without the approval of the policy enforcer.

Just as with known security threats, it can take the form of external or internal threats. Also, it can be direct or indirect modifications. For example, in the external threats a subject that does not belong to a given part of a system accesses an object and modifies it’s content with the goal of corrupting that subsystem that it does not belong to. An example of an internal threat would be a subject in the same subsystem modifying an object to corrupt its data to affect other entities in the same subset.
Direct and indirect represent the means of the attack. If the subject modifies and corrupts an object directly, then it is a direct attack; otherwise, if it modifies an object with the goal of corrupting another subject or object, then it is considered an indirect attack on integrity.

Again, our goal is to identify these threats and prevent them from taking place. To do that, we have to enforce a security policy that handles these attacks and fully supports the integrity requirements of our system. There are two forms of integrity policies [4]:

1. Mandatory Integrity Policy: This policy, when defined for a given entity, means that the integrity level of that entity may not change during its lifespan. It is viewed as a static policy.

2. Discretionary Integrity Policy: This allows for change to the policy protecting the entity by expanding or limiting the access to owned system resources by the user and is viewed as dynamic.

Choosing which of the previous two options to use is based on the requirements of the system. For instance, it is expected to choose the Mandatory Integrity Policy if the system we are trying to protect is highly related to secure systems. Changes to the policy are not welcomed here since we want to prevent these changes even if the surrounding environment shifts.

The policy would have to list all possible relationships for each access type a subject would have with an object with the goal of securing the object from unauthorized access. It must take into consideration that the subject has to have the object within its domain. That is with respect to the access guidelines presented by Biba.
1.10 The Biba Model

We briefly described this model early in this chapter. Here we provide more information. In 1977, Biba [3] studied the requirements of integrity for a given system. He came up with a number of policies to handle these requirements. In a given system, an integrity policy consists of a set $S$ of subjects, a set $O$ of objects, and a set $I$ of integrity levels. The basic policy of Biba is that when we have a subject with low-level integrity, then information from such a subject should not be allowed to flow to higher-level integrity objects. However, when we have a high-level integrity subject, then we can allow for its information to flow to lower-level objects. Biba has the following controls:

- The simple security property: A subject $s_i$ can read from an object $o_i$ only if it has an integrity level $s_{il} \leq o_{il}$.
- The Star (*) property: A subject $s_i$ can write to an object $o_i$ only if it has an integrity level $s_{il} \geq o_{il}$.

The two controls prevent unauthorized reads and writes. The first simple integrity property prevents a subject from reading from lower integrity objects. The star property prevents a subject from writing to an object unless the subject has an equal or higher integrity level. Under such control-flow, a subject reads up and writes down.

This is the Biba model that most other proposed integrity models are based on. We use this model as the basis for our solution to have a dynamic integrity model.
1.11 Access Polices Based on Biba Model

Here we present some of the policies from [3] to handle integrity requirements that are based on the Biba integrity model:

- Low Water Mark Policy: In such a policy, if a subject accesses a given object, then the integrity level for the subject will be changed to be the lower of the subject and object. The writing process under this policy is protected in that a subject cannot write to a higher-level integrity object to prevent the corruption of such object. However, reading is allowed, allowing a subject to read from any object with no condition on whether the object has a higher or lower integrity level. When a subject reads from a higher integrity level object, nothing is expected to happen to that subject because the policy will pick the minimum integrity level, i.e., the subject’s, and reassign it to the subject. On the other hand, if the subject reads from a lower integrity level object, then its integrity level will be lowered to be that of the object. The policy protects against any form of unauthorized direct modification in the case of writes and also any unauthorized indirect modifications in the case of reads. However, the problem with such a policy is that there is no way for the subject’s integrity level to recover after accessing a lower level integrity object. Because even if the subject reads from higher integrity level objects, their level will still continue to be the same, until another lower integrity level read takes place, which will result in the subject level dropping again until it reaches a minimal level.
- **Ring Policy**: This policy focuses only on the direct modification with respect to writes. It assumes that any subject can read any object without decreasing its level of integrity, even after a low-level integrity read.

- **Biba Model (Strict Integrity Policy)**: This policy is the full Biba Model; it has the flow controls mentioned early in this chapter: the simple property and the star property with the addition of execute.
  - The simple security property: A subject $s_i$ can read from an object $o_i$ only if it has a integrity level $s_{il} \leq o_{il}$.
  - The Star (*) property: A subject $s_i$ can write to an object $o_i$ only if it has a integrity level $s_{il} \geq O_{il}$.
  - A subject $s_1$ can execute a subject $S2$ only if the integrity level $s_1 \geq s_2$

What this policy does is protect against the direct and the indirect accesses. We saw that in the Low Water Mark policy the same is done by lowering the level of integrity of a subject to a level where it cannot recover. Under the strict integrity policy, the access of a subject requesting to read from a lower level object is not authorized. This is why it is called a strict Integrity Policy, and the low Water Mark policy cannot be applied.

In order to protect the data integrity, we have to know what the integrity requirements are. In [11] the authors list four main integrity requirements that any integrity management system must address. These are:

- **Controlling the flow of information**: That is to protect of high integrity data from being contaminated by lower integrity data.
- **Data verification**: Makes sure that verified data is only what is provided to a given transaction.
• Fraud and error prevention: Only correct and legitimate data is provided to information systems.

• Automatically validating data: Maintains the integrity of the data independently from data access.

The protection of data integrity and the detection of data integrity violations fall into two main classes as presented in [3]:

• Prevention: Maintain the integrity by blocking any attempt to modify the data improperly or by an unauthorized entity.

• Detection: Does not prevent unauthorized modification but reports it by analyzing the system events to determine if it may be the case that the data integrity is no longer trustworthy.

Under the prevention mechanism, the goals of an integrity protection policy as in [3] is:

• Preventing unauthorized users from modifying data and programs.

• Preventing authorized users from modifying data or programs improperly.

• Ensuring that the internal and external consistency of data to be maintained all time.
1.12 Information flow

At the center of the access control problem is the information flow. In [3] an information flow policy is one that defines how information moves through a system. Such a policy is designed to protect confidentiality or integrity during the information flow. For a policy protecting confidentiality, it wants to prevent information from flowing to a receiver that is not authorized to receive it. On the other hand for a policy protecting integrity, it wants to make sure that information flows only to processes that have a trust level equal or less than entity itself. So in general the information flow policy describes the authorized paths along that information can flow.

An access control program verifies access rights at the beginning before a program is allowed to run. After that theirs is no way to make sure that the running program accesses information as intended, which may cause a leak in information.

So how do we make sure that the flow of information is secure? In [28] program variables are classified into different security levels. The low security levels variables, public information as $L$; and other variables as high security level, private information as $H$. The goal of the security policy is the prevention of the leakage of information in $H$ variables. We want information in $H$ variables from flowing to $L$ variables.

In a lattice of security levels, the information would flow upwards. For example, if $L \leq H$ then the allowed flows are from $L$ to $L$, from $H$ to $H$, and from $L$ to $H$ but not allowed from $H$ to $L$. This represents the flow of information under confidentiality. Under integrity it is assumed that it is the possibility that variables at lower security levels are tainted and we don't want to
have such variables affecting the data at higher levels that are not tainted or untainted. So we would allow flows from $L$ to $L$, from $H$ to $H$, and from $H$ to $L$.

The information flow policy aims at preventing the leak from taking place. This usually happens after an access is granted to information. Since the confidentiality flow of information is to the opposite in direction it is difficult to have a policy that can protect both at the same time.
CHAPTER 2

Problem Definition and Literature Review

2.1 Problem definition

Integrity has been discussed since the late 1960’s, and most currently proposed models are static in nature, meaning that a subject or object starts with a pre-set integrity level and continues without any integrity-level change for its entire life cycle. Static integrity models do provide good security and reduce the overhead involved in data integrity protection. Dynamic integrity models may provide better data integrity protection especially when dynamic applications are involved. Currently web applications and languages like JavaScript are highly dynamic and that could demand a dynamic integrity protection. Also, some collaborative work is done online; for instance, Wikipedia is a dynamic and collaborative knowledge sharing system. Anyone can contribute to Wikipedia articles. However how much trust can be placed in them and how can we evaluate the integrity of the users who contributed to the articles? Under collaborative environments, it can be a challenge to evaluate the trust of others work, and it could be very helpful to do this evaluation dynamically.

A dynamic integrity model would allow us to establishing the level of trust an entity should have based on its behavior and history and this trust or integrity level could continue to change during its lifecycle. We are developing a model under which subjects can change their level dynamically according to a security policy without the need to involve the system administrator for each change. We think subjects can build their level of trust slowly, allowing them to change their integrity level dynamically based on their sources of information.
Some dynamic models have issues such as not being able to recover if the level of integrity goes down just like in the Low-Water-Mark policy. We think that it is possible to implement a dynamic integrity model in a simple and effective way.

Our goal is to allow for the information flow of trustworthy subjects to travel from source to destination up or down inside the system however we want to restrict that flow in a way that recognizes the trustful ones and allows them to take place. We do this by enforcing a security policy that rewards trustworthy entities by expanding their access. We also penalize untrustworthy ones by further limiting their access. The expansion and limitation of the access is done by increasing or decreasing the integrity level of an entity.

This research introduces and verifies the value of dynamic approach to changing the integrity levels of a subject or object dynamically by using a factor called the modification factor. This policy is based on the concept that if an entity modification factor is high, then its trust level is questionable, and if its modification factor is low, then its trust level is showing signs of improving.
2.2 Literature review

In this section, we review representative integrity models.

2.2.1 Chinese Wall Model

The Chinese wall model was introduced by Brewer and Nash in 1989. It is a security model that addresses both integrity and confidentiality and is thus viewed as a hybrid model. It describes policies that addresses possible conflicts of interest in business. An example of a conflict of interest situation is the stock exchange. Suppose that a trader represents two clients, and these clients’ best interests conflict thus, the trader might help one client gain at the expense of the other.

The security policy has three levels of abstraction:

- Objects: Objects contain information about only one company.
- Company dataset (CD): Collection of objects related to a certain company.
- Conflict of interest (COI): Class containing the datasets of competing companies.

Under the Chinese wall policy we have:

- Simple access control rule: Subject $S$ can access object $O$ only if:
  - $S$ is in the same CD as the $O$ (within the Wall) or
  - $S$ is in a different COI than $O$.

An example is if we have a COI containing GMC, Ford and Nissan then if a subject access GMC information they are blocked from accessing Ford and Nissan information to prevent conflict of interest.
The access permissions change dynamically based on the history of the past access in this case as in the example the access rights has GMC only related information to prevent conflict of interest.

- The *_property Rule: Subject S can write to Object O only if:
  - Access is allowed by the simple security rule allowing S to read O, and
  - O contains unsanitized information.

Example: Trader 1 can read information about bank A and another company A. Trader 2 can read information about bank B and company A. However since bank A has unsanitized information, the trader is not allowed to write onto company A CD to prevent a conflict of interest if trader 2 access information of company A.

This policy is designed with integrity and confidentiality in mind to solve a specific problem that is the conflict of interest. Since the rights of access change dynamically based on the previous access this makes this policy a dynamic one.
2.2.2 Clark-Wilson Model

David Clark and David Wilson first introduced their Clark-Wilson model [6] in 1987. The motivation behind this model is that previous models based on the BIBA model focused on the military requirements and were viewed as lacking the requirements to perfectly model the commercial needs. Some even think that the BIBA model fails to handle the commercial requirements as needed. In a commercial environment, the integrity of the data and the operations conducted on it are very important. Also the integrity of the conducted transactions is important. The policy has to make sure that the conducted transaction is performed correctly in accordance to what is required in the policy itself. In such policies, according to Matt Bishop, the separation of duty requires the implementer and the certifier to be to different people so that corruption of the data cannot take effect on a mistake made by one person but both have to make the same mistake to cause the corruption of data.

The Clark-Wilson model is based on the idea of transactions for the operations. According to Matt Bishop [3] under such a model “The data is said to be in a consistent state if it satisfies given properties.” This condition must be true and hold before and after each transaction is made. The system must transition from one consistent state to another in order to maintain its consistency.

The Clark-Wilson model defines two sets of data. The first set is the Constrained Data Items (CDI). These are data that are important, and their integrity will have an effect on the system when modified. For example, the account balance needs to maintain its integrity after and before each transaction so that the account balance data is a CDI. The other set is called the Unconstrained Data Items (UDI). These are any data items which will not have any effect on the
integrity of the system. For example, a bank account owner is given a set of gifts when opening his account. These data items are not related to the integrity of the system [3].

The model also has the set of Integrity Verification Procedures (IVPs) that are responsible for testing the CDIs to see if they maintain their integrity each time the IVPs are run. If the case is yes (i.e., maintain their integrity), then the system is said to be in a valid state. After that, the TPs, which are the Transformation Procedures, transform the system from one valid state to another in the manner of a well-formed transaction. According to Bishop [3], a well-formed transaction is “A series of operations that transition the system from one consistent state to another consistent state.”

According to the original paper published by Clark and Wilson describing their model [6], their model has a two-part process. The first one is related to certification, which is handled by the security officer or the system owner. They are responsible for the security policy. The second process is the enforcement, which is conducted by the system. For this, they introduce a number of certifications and enforcements based on the two processes as described in their paper [6]:

“C1: (Certification): All IVPs must properly ensure that all CDIs are in a valid state at the time the IVP is run.”

“C2: All TPs must be certified to be valid. That is, they must take a CDI to a valid final state, given that it is in a valid state to begin with. “

“E1: (Enforcement): The system must maintain the list of relations specified in C2 and must ensure that the only manipulation of any CDI is by a TP.”
These rules are used to ensure the internal consistency of the CDIs. The authors argue that there has to be another set of rules to ensure the external consistency of the CDIs, since the separation of duty principle is used. They need to make sure who can and cannot execute a program on a given CDI in the system. So they introduce the following rules to handle such requirements [6].

E2: The system must associate each user with each TP and a set of CDIs. It also must make sure that the user’s TP can access only those CDIs associated with it.

C3: Relations made in E2 must satisfy the requirements of the principle of separation of duty. This rule points out that the system is handling two forms of consistencies: internal and external. While E1 handles the internal consistency, E2 handles the external consistency.

E3: The system must be able to perform the authentication process for a user attempting to execute a TP.

E3 is used to enforce a policy on a given user, since it might enforce more restrictions added to the user.

C4: All TPs must append-only information to a CDI in order to reconstruct the operation.

The need for this certification rule is to log everything done to a CDI, so that we can trace the audit track that has been done.

C5: Any TP using a UDI as an input and then using it to modify a CDI must be certified to conduct only valid transformations or no transformations for all possible values of the UDI. The input is taken from a UDI and transferred to a CDI.
E4: In the system only the certifier can change the list of entities associated with a given TP to prevent unauthorized modification by the user for the TP associated sets.

So how does this model compare to previously proposed integrity models? The authors did compare their proposed model to other models like the Biba model, which is an integrity model.

The Biba model, according to the authors, assumes that subjects and objects do exist at different levels of integrity. Also that a lower level subject cannot write to a higher level object, since it is prohibited by the policy implemented in the system. According to the authors, their model has two levels of integrity: lower level UDI\text{\textsc{is}} and the higher-level CDI\text{\textsc{is}}. The authors think that since in the Biba model only the security officer can convert the UDI to a CDI that this is not functional, and would add more overhead. Thus, they think since the CDIs can be verified by the IVP, then it would be more flexible and realistic when simulating daily work commercial environment.

The Clark-Wilson is a realistic model to control integrity of a system; however, it has some drawbacks as mentioned in [18]. The model classifies data integrity as only valid and invalid. Having only these two representations of data might fail to deal with various types of data like the case where we can extract useful data from a group of untrustworthy data. The other issue is the assumption that data preserve their valid states as long that the data is verified by IVPs and only accessed by TPs. These assumptions work very well with data that is static, and this makes a problem for data that have dynamic factors that cause them to change over time. Another issue with this model is that it does not address the non-verifiable CDIs and how it should be handled. The current assumption is to delete it, which is viewed as too simplistic by some researches.
2.2.3 A Biba Model with dynamic characteristics

The authors of [17] present the strict integrity policy (SIP) with dynamic characteristics. They argue that their model secures a system by extending the integrity level of a subject into separate reading and writing integrity levels. The model tracks the minimum integrity level the subject has read from and the maximum integrity level the subject has written to. Previous behavior of the subject affects the access requests allowing this proposed policy to be more flexible for the following reasons according to the authors:

- The policy enhances the reading and writing ability of the subject while maintaining the system’s integrity just like as in the original SIP.
- The policy can represent different behaviors of the subject because of the allowable range of different integrity levels for both reads and writes.
- The policy increases software compatibility which according to the authors prevents an authorized user from accessing the system with malicious intent.
- The policy allows for one-time arbitration systems without repeating the arbitration.

The policy has dynamic aspects by extending the range the subject can read down or write up. Although that data at higher levels of integrity is assumed to have high quality data, this policy allows the read down with no impact on the integrity of the subject. In addition writing up is problematic because extending the range of the subject to write up raises the question: are high integrity level objects going to be impacted with such write?
2.2.4 Integrity-OrBAC Model

Another model is provided in [7]. The authors present an Integrity-OrBAC model. It is based on RBAC (Role Based Access Control). It defines roles that subjects can take to perform tasks. For instance, in an organization, a subject can be in the role of a department secretary, who needs to be given a set of privileges associated with that role. OrBAC is an extension of the RBAC, thus making it based on the roles concept. OrBAC provides easy to manage policies by dividing them into levels. The first level is the abstract level, which represents the policy through abstract entities, regardless of how they are implemented in a given organization. The second level is the concrete level in which it responds to user access requests based on the defined rules of the organization.

OrBAC is usually associated with critical infrastructures (CIs) in an organization. Such CIs may be in government or may be organization all important structures that rely on the heavy use of technology in their environments. However, the authors think that even when OrBAC is attending the needs and requirements of the CIs, it fails to have proper access control mechanisms. In addition, it fails to address the integrity issue because subjects can inherit the privileges without establishing any creditability. Also, OrBAC does not distinguish between highly important and less important entities in the system. This motivated the authors to provide an enhancement to the OrBAC model, with their new proposed model taking into consideration the integrity problem.

The authors think that when designing the security policy, the designer must take into consideration what is important to the organization from three different perspectives. First, the objective of the security requirements, what needs to be protected, and from whom or what. The second point is related to the actions or requests inside the system. The designer must be able to
differentiate between what is a routine access request and what is not. The third one is related to
subjects. The designer must be able to assign the correct privileges for subjects and know all
possible scenarios under which the policy is going to operate.

In this paper, the authors argue that assigning integrity levels to roles is hard and not
realistic. Therefore, the integrity level is assigned to subjects instead. The integrity level reflects
the creditability and reliability that a subject has inside the organization. The higher the integrity
level of the subject, the more confidence an organization can place in that subject to successfully
perform a certain task. This integrity level is based on the history of the subject, and its
reputation in its given domain.

The authors provide the reasons why they think that this is more realistic. First, the
integrity given to a role is not logical, since people having the same role may have different skill
levels. Second, it is easy to track the person who caused harm to the system integrity. They are
easily identified and possibly penalized for their action. This will allow for different integrity
levels for a subject having roles in different domains in the organization. This we think is a form
of dynamicity since the subject can have multiple integrity levels in different domains in the
organization.

When it comes to objects, all objects having the same level of sensitivity are grouped into
the same view. The integrity level is assigned to the view, which is inherited by the object. The
integrity level is assigned to the view based on the criticality and confidence of the objects it
contains. The idea here is to ease the cost of management by having all objects of the same
importance grouped in the same view.
All actions are grouped into activities based on their sensitivity. The integrity level is assigned on the basis of criticality of the activity, meaning what sort of an impact such activity could cause inside the organization.

This way, the authors think that it provides more flexibility for any organization by defining different integrity levels with different activities based on the organization requirements. An access control decision is based on three important parameters in order to ensure that the integrity of the source is not compromised. These are context integrity level, the view integrity level, and the activity integrity level in the role. These parameters will impose constraints on the subject integrity level. In the case of abuse, the subject who caused it will have its integrity level reduced, causing its privileges to be restricted and possibly removed.

We think that attaching the integrity level to the subject instead of the role will add more overhead since the subject will have multiple integrity levels across different domains. Another issue is collecting all possible activates inside the system to attach an integrity level to it. We believe this is might not be possible in large systems.

2.2.5 Totel Model

Another integrity model is called the Totel model [8], which is based on an object-oriented approach. In this model, the authors define objects as an entity providing services requested by a client. The integrity level of that object is based on the level of trust it has. The level of trust is reflected by its correctness and the severity of a failure. Since this model is based on an object oriented approach, then messages are used to invoke a service. A message has the
same level of integrity as the object that created it. Also, each message has a label indicating how much trust can be placed in it.

The authors provide an integrity policy on the basis that low integrity data should not be allowed to corrupt a high-level integrity object. The goal of the integrity policy, according to the authors, is to provide as presented in[8]:

- “A strict information flow integrity policy.”
- “A flexible approach with fewer constraints than the Biba policy.”
- “Mechanisms to handle information integrity upgrade.”

These goals lead the authors to define three object types. These types are:

- Single Level Objects (SLOs): these objects will have a single constant integrity level. An SLO can contain information and can create flows using the internal data they have. This type of object is restrictive, preventing it from being used at different levels of integrity.
- Multi-Level Objects (MLOs): these objects have more flexibility allowing them to provide services at different levels of integrity. It is validated to a given integrity level, and can accept data from any entity of a lower or equal level integrity. It can dynamically modify its integrity level by going back to its starting integrity level at the beginning of each invocation.
- Validation Objects: it takes low integrity inputs and runs fault tolerance procedures to produce high integrity outputs.
An invocation takes the forms of a read, write, and read-write. The authors give the following flow control policy:

- Single Level Object: an SLO is given a constant integrity level based on the trust that can be placed in it. It must only access data with higher integrity than the one it has. Also, accepting data from lower integrity objects is not allowed. A lower or equal integrity object can read an SLO. Also, an SLO can write to a lower level object.

- Multi Level Object: each MLO is given an integrity level based on its trust ability. This is called the intrinsic level, which is the highest that it can reach. Upon invocation, the MLO can run between lower levels and the intrinsic level. It also can downgrade its integrity level during the execution to reflect the level of the information it is using. However, it cannot upgrade until the executed method is complete.

- Validation Object: the goal here is to validate the flow of information from lower integrity to higher integrity objects. It must be an exception to the policy to allow for such flow to take place. Implementing the fault tolerance mechanism will upgrade the information to the requested level and prevent the corruption of higher integrity level objects.

The integrity rules under which this model operates are:

- SLO to SLO must only access data with higher integrity level than the one it has. Also, accepting data from lower integrity objects is not allowed. A lower or equal integrity object can read an SLO. Also an SLO can write to a lower level object.

- An MLO accessed in the read mode must have its maximum integrity level higher than that of the calling object. If this condition holds, then its minimum integrity level
becomes the same as the one of the calling object. This way, if the integrity level of the MLO falls below that level, it can be detected.

- The result of an invocation will be checked, and the MLO will have its integrity level changed to reflect the integrity level of the results it received.

- An MLO accessed in the write mode will have its integrity level calculated from the minimum of its intrinsic integrity level and the level of the calling object. If the object has a higher integrity than the MLO, the access is not granted.

The authors proposed this model based on the object-oriented approach to simulate multi-level requests. It defines an access control model for software components of multiple integrity levels, which we view as a dynamic. The goal of the model is to make sure that data of low integrity level objects does not flow to objects of higher integrity level objects. The assumption in this model is that the integrity level of a given entity changes during the execution of the software.

2.2.6 MultiLevel-OrBAC

The work done in [8] motivated the authors in [9] to use it as the basis for a model called the MultiLevel-OrBAC. The authors propose using an OrBAC access control model in addition to the Total [8] integrity model in an environment with several subject and objects with different criticality levels. The authors present the following two rules:

- Read Integrity rule: using validation objects just like in [8]. A subject is allowed to read an object only if it has an integrity level that is equal to or lower than the one the object
has. Or it can read it when using the validation objects to make the integrity level of the object higher than the one that the subject has.

- Write Integrity rule: A subject can write to an object if it has an integrity level higher than or equal to the one the object has. If the subject has an integrity level lower than an object, then the validation object is used to change the subject’s level to a higher integrity level, allowing for such a write to take place.

The authors argue that their proposed model, by combining OrBAC and Totel, results in a better access control that takes into consideration integrity rules along with integrity levels. They also think that it provides an improved management to the security policy and also reduces complexity. It also, according to the authors, allows for access (read, write) between different levels of subject and objects via the use of validation objects that allow for such action to take place using the fault tolerance mechanism.

2.2.7 SELinux

An implemented access control model that is part of the Linux operating system is SELinux [10]. It is based on a mandatory access control (MAC) model first developed by the National Security Agency and which became part of the Linux operating system kernel. SELinux is designed to protect both confidentiality and integrity at the same time by preventing unauthorized reads or writes.

For this research, our focus is on the integrity part of SELinux. SELinux uses Type Enforcement (TE) to enforce its security model. A typical TE places each subject into a domain,
and each object into a type. Subjects in a given domain are treated identically. Objects in a given type are also treated identically. The system maintains two access matrices. The first is the Domain Definition Table (DDT) that is responsible for determining authorized access between domains and types. The second is the Domain Interaction Table (DIT), which is to determine authorized access between domains. The TE access matrix specifies what each domain can invoke in terms of types.

SELinux uses permissions to grant access from subject to object. TE in SELinux is different from the original TE, since it does not distinguish between a domain and type. They are both the same. TE uses a security class and type to make an access decision or deny it. This way, objects having the same type but belonging to different security classes can have different access decisions. This makes SELinux more flexible, and it is expected to provide a better level of security.

SELinux also uses roles to define what actions users can do inside the system. However, the way roles are defined is different from the RBAC traditional policy. SELinux defines allowable actions by users for a given role using the TE policy described previously. The TE policy will specify what domains can be entered by a given role, which is different from the RBAC policy that would define what permissions are granted to a given role.

Even with the access protection power SELinux provides it has issues rising from being complicated. The Admin skill has to be high to manage policies under SELinux it also requires detailed configuration, which makes it not easy to use especially since the targeted users are an enterprise users. The policy language under SELinux is complex and long, which makes it hard to adjust as the requirements change.
2.2.8 TACSIF

Another access control model based on trust [11] is the TACSIF (Trust-Oriented Access Control Based on Sources of Information Flow). It uses the information flow to describe the integrity of the system, and also adjusts them dynamically.

The authors define information flow as “Transferring information from one entity to another.” The source of this flow is the entity that information came from, and is called Source of Information Flow (SIF). The destination of the flow is the entity receiving the information and is called the Destination of Information Flow (DIF). The model has two forms of information flow:

- Direct Information Flow: This is when there is no entity between the SIF and DIF.
- Indirect Information Flow: This is when the flow of information has to go through multiple entities between the SIF and DIF.

TACSIF has two types of operations: observe and modify. Observe would be like a read where a subject reads from an object. Modify is when a subject writes to an object.

Each entity has a set of SIFs and they are used to describe the integrity level of that entity. The authors point out that the more SIFs a given set has, the lower the integrity level of that entity. On the other hand, if that SIFs set has few SIFs, then that entity is expected to have a higher integrity level.
TACSIF has the following definitions:

- **Lower Bound of an Entity**: The set of all SIFs of that entity at a given time when there is minimal information flow transferring to that entity. It represents the highest integrity level of that entity at that time.

- **Upper Bound of an Entity**: The set of all SIFs of that entity at a given time when there is maximal information flow transferring to that entity. It represents the lowest integrity level of that entity at that time.

- **Dominate Relation**: \( E_1 \) is said to dominate \( E_2 \) when \( E_1 \) has a higher integrity level than \( E_2 \).

- **Threshold Integrity Level (TIL)**: It has the same integrity level as the entity.

- **Current Integrity Level (CIL)**: The set of SIFs which are elements in TIL and also have transferred information to the entity.

- **Current Access Set**: A set that has all performed accesses in the system.

Each time information is transferred to an entity, the system updates the integrity level of the DIF receiving the information flow.

TACSIF has the following access rules:

- **A subject observing an object (read)**: The system has to check and see if the CIL of that object dominates the TIL of the subject. If that is correct, then the subject can read from the object.

- **Updating after an observe (read)**: The system compares the CIL of both the subject and object. If the object CIL does not dominate the CIL in the subject, then the system
updates the integrity level of the subject and adds the SIFs in the object’s CIL to the subject’s CIL.

- A subject modifying an object (write): The system has to check that the CIL of the subject dominates the TIL of the object. If yes, then the subject can modify the object.
- Updating after a modify (write): The system uses the subject’s CIL SIFs by adding them to the object’s CIL to indicate that a flow of information has been recorded.

TACSIF uses these rules to make a decision to grant or deny request to observe or modify an object in an environment where there is a flow of information like a networking environment. In addition these rules are used to update the integrity level of subjects and objects to reflect the flow of information taking place, which we view as a dynamic approach to changing the integrity levels. Our model conducts the dynamic change to integrity level in a simpler and manageable way by observing the behavior of entities operating inside it.

2.2.9 TAAC

Another proposed is called the Trust-Aware Access Control Model (TAAC) [12]. It is designed with the goal of handling the integrity problem in distributed systems. TAAC has an integrity level for each entity (subject or object). The authors argue that using only the integrity level to decide to grant or deny access is not enough to make a good decision. The reason is they think that it is not reflecting the true integrity of that entity, which in turn could lead to a denial of access. Adding trust to their proposed access model, the authors think will provide for a better
access control model. Using the trust concept derived from the work of the Trust Computing Group (TCG) [13], the authors present three trust definitions:

1. “Trust: An entity can be trusted if it always behaves in the expected manner for the intended purpose.”
2. “Trust Level: The degree of trust that an entity believes a special entity behaves in their expected manner.”
3. “Trust Threshold: The lowest trust level needed by an entity that interacts with other entities.”

The TAAC system has the following security properties:

1. Threshold property: A subject can read another object only if it has a trust level higher than the one associated with the object. This is to establish the threshold between a subject and object.
2. Access list property: A subject can read any object that is higher than a certain threshold. The collection of threshold list of an entity is called the access list of an entity.
3. Capability property: Ignoring the integrity level, a subject can access an object if it has the capability to do so. For example, communicating via a certain communication port like in SSH.

TAAC uses the previous security properties to establish the access rules for reads and writes:

- Read: A subject can read an object if it falls into one of the following:
The trust level and trust threshold maintain security properties 1 and 2 in addition to known integrity policies like Biba.

- Security properties 1, 2 and 3 hold and the integrity levels of the subject and object may be ignored.

- Write: just like read. However, if we use Biba then the subject integrity level has to be higher than or equal to the one of an object. The rest is the same.

We think that in this model more restrictions are added and it could lead to possibly denying entities access to higher quality data. Because for instance in property 1, the subject has to have a trust level higher than the one of the object and under integrity policies like Biba, the subject can read from an object if it has an integrity level equal to or lower than of the object. In addition many researches associate trust with integrity level.

2.2.10 A Dynamic Trust Model for Mobile Ad Hoc Networks

Integrity is also viewed as the level of trust that can be placed in a subject. In [19] the authors aim at changing the levels of trust of a mobile node based on its behavior and what its neighbors know about it. The work is presented to solve the trust and integrity problem in mobile ad-hoc networks. We believe that the work in this paper is close to our model. The authors define trust “as the reliability, timeliness, and integrity of message delivery to their intended next-hop”. They make the assumption that each node is able to detect the malicious behavior of its neighboring node. The authors believe that having a distributed Intruder detection system (IDS) will help the model achieve its goal.
The authors propose that each node is authenticated using a mechanism that is based on its history and behavior to assign it a trust level. In the case that there is no history or knowledge about the node then it is given an unknown trust level until an observed behavior takes place that will change its trust level. In the event that a node is attacked then its trust level must drop down to allow other nodes to be aware when using it to send information. That level is adjusted dynamically based on techniques they use like the use of IDS (Intruder Detection Systems) implemented in each node to monitor if it is compromised. Also the neighboring nodes would be able to detect what is happening with each neighboring node using IDS tools. When a node wants to send a message to another node the message has a level and the routing algorithm to allow it to travel only on paths to the destination having the same level or higher.

The behavior of each node has an impact on its trust level. If a node behaves in an appropriate manner then its trust level should be raised according to the authors; otherwise if it behaves in a bad manner then its trust level should be decreased. Other nodes will be able to trace that change in trust level and will make the routing algorithm aware of such changes when sending information via neighboring nodes.

According to the authors each node should broadcast its trust reports reflecting its trust level to its neighboring nodes so other nodes can evaluate how trustful that node is for future communications. Although the work done by the authors is close to what we are trying to do, our work does not allow the entities to change their integrity level independently but we have a central policy enforcer that does the changes according to a policy.

In the next chapter, chapter 3, we present our integrity model which includes an easily understood method for dynamically changing integrity levels of entities without direct interaction of a system security officer.
CHAPTER 3

The Model

3.1 Introduction

In this chapter we present our dynamic integrity model in a generic form. In chapter 4 we present the implementation to test the generic model. In chapter 5 we add extensions to this model in order to make it applicable in other situations.

In order to protect the data integrity we have to know what the integrity requirements are. In [18] the authors list four main integrity requirements that any integrity management systems must address. These are:

1. Controlling the flow of information: That is to protect of high integrity data from being contaminated by lower integrity data.

2. Data verification: Makes sure that verified data is only what is provided to a given transaction.

3. Fraud and error prevention: Only correct and legitimate data is provided to information systems.

4. Automatically validating data: Maintains the integrity of the data independently from data access.

The protection and detection of data integrity violations falls into two main classes as presented in [3]:

49
• Prevention: Maintain the integrity by blocking any attempt to modify the data improperly or by an unauthorized entity.

• Detection: Does not prevent the modification but detects and reports by analyzing the system events to determine if the data integrity is no longer trustworthy.

Under the prevention mechanism, the goals of an integrity protection policy as in [3] are:

• Preventing unauthorized users from modifying data and programs.
• Preventing authorized users from modifying data or programs improperly.
• Maintain internal and external consistency of data at all times.

Integrity is a way to guarantee the authenticity of data. We think it is of use to have a dynamic integrity level change model for some applications, for instance in the case of open author collaborative systems. The strength of such systems is that anyone can contribute to it. It is shared knowledge however that its weakness stems from the fact that it is open to anyone to add, delete or even falsify information. In some of the current implemented systems like Wikipedia, integrity is changed dynamically and statically to address these issues. We believe that our proposed model can benefit such systems.

We also believe that our model can be of use to business environments when they have shared documents. Shared documents among different business departments must maintain their integrity to prevent their corruption or improper modification. This means that we have to adjust the integrity level dynamically based on the behavior of the entity in order to preserve the high quality of such shared documents. Using the behavior history of the entity we can upgrade its integrity level or we can downgrade it preventing the corruption of higher quality documents.
3.2 MAC vs DAC

As seen in chapter 1, access control models can be a DAC (Discretionary Access Control) or MAC (Mandatory Access Control). If the model is a DAC model, then the integrity protection has drawbacks. Under DAC the user can grant or deny access to whom they choose; however, they have no control on accessed data because the user who is allowed to access the data might be accessed by another person who should not access such data. The problem becomes the indirect access problem in DAC.

Under MAC the security policy determines who can access the data based on the policy which attaches labels to subjects and objects. Comparing these labels will determine if the subject has the right to access the data. Even when MAC is complicated it is still desirable because it offers better protection than to DAC.

In our model we choose the MAC approach, and we will present in chapter 5 an extension which we believe makes our model more applicable in other situations.

Under our model, all access is interpreted by the policy enforcer which consults with the protection policy to make the decision (to grant or deny access request) based on level and type of access from a subject to an object. Figure 2 shows the structure of the model.
3.3 Access Modes

Before we present our model we would like to address the types of access to data in our model. We have two main access modes: Read and Write.

A read access where the subject observes an object with information so the direction of flow of information is into the subject as in figure 3 where S1 is reading from object O. We do treat read also in two different forms. The first one is a non-protected read which means that the subject has no way of filtering out the information which can have an impact on the observing entity.

The second is a protected read where we make the assumption that the observing entity can filter out bad data and preserve its quality all the time.
In a write access, the flow of information is from the subject towards the object. This type of flow has an impact; it modifies the information already in the subject. Figure 3 shows the flow of information into the object where S2 is modifying, i.e., writing to, object O.

Figure 3 The direction of flow of information for read and write

3.4 The Protection Policy

The protection policy has to identify which entities are subjects and objects so it can offer the needed protection. It is the case that some subjects can be objects in some situations. When it comes to assigning the integrity levels, it is usually based on the organizational requirements and the impact that the compromise of an entity could have and the amount of damage it could do. Therefore we think that protected read is an example for protecting higher integrity level entities from corruption of information if read from lower level entities.

Assigning the integrity level to users is based on the trustworthiness of the user. When compared to confidentiality, confidentiality bases its security levels on the fact of disclosure of
information and sabotage. Integrity on the other hand is based on the behavior of the user and how much trust can be placed in them to modify or read information. Modification is the critical part because it can cause large amounts of damage to information and possibly other users.

In our model we use the history and previous behavior of the entities to adjust their integrity levels dynamically. We implement this in a manner to change the levels to higher levels more slowly than changing the levels to lower levels.

Biba presented an audit policy where the policy measures the amount of contamination across the database based on the modifications taking place. However this policy does not change the integrity levels but rather reports the levels of contamination.

Next we present our model in detail description and in chapter 4 we present our implementation.

3.5 A dynamic integrity model

The goal of our model is to change the integrity level of an entity dynamically using a metric we call the modification factor. Our modification factor is used to evaluate how much an entity is influenced by another entity whether that influence is in the form of read or write.

A read from a higher integrity level object will have a positive impact on the observing entity in the form of a reduction in the modification factor of the subject, which can lead to an increase in its integrity level. On the other hand a read of a lower integrity level object will have a negative impact on the observing entity by increasing its modification factor, which could lead to a change in its integrity level down.
In the case of writes we use the integrity level differences to decrease the modification factor for the modified entity to allow it to change its integrity level up dynamically. We will present a detailed explanation of how we intend to implement these changes later in this chapter. At the center of our model is a policy enforcer that handles every access request and makes the decision to grant it or not based on the defined rules.

First we present some of the terminologies that we are going to use in our model:

- **S**: Set of subjects where \( s \in S \) is the active entity.
- **O**: Set of objects \( o \in O \) where \( o \in O \) is the passive entity.
- **I**: The set of integrity levels.
- **il**: Integrity level, a function mapping a subject or object to an integrity level such that \( s_{il} \) is the integrity level of the subject and \( o_{il} \) is the integrity level of the object.
- **HQR**: High Quality Read. When a subject reads from a higher integrity level object.
- **LQR**: Low Quality Read. When a subject reads from a lower integrity level object.
- **MF**: The modification factor associated with each entity has levels ranging from \(-M\), which is the minimum, and \(M\), which is the maximum and 0, which is the middle.

For integrity levels range compared to the MF range they are in opposite direction. For the integrity levels the higher the integrity level the higher the quality of the data is whereas in the case of the MF the lower the value is the less contaminated the data is.
3.5.1 Assumptions:

Next we present our assumptions:

- The environment where our model is applicable is one where continuous dynamic integrity changes could naturally take place like in a collaborative environment.
- We use the integrity levels to make access decisions and the MF to decide whether or not to change the integrity level of an entity up or down.
- We assume that having a higher integrity level does not mean a lower MF. These two values are independent until the MF reaches its maximum or minimum values.
- The security officer sets up the initial integrity levels of subjects based on the policy written; however, the MF always starts in the middle.

We use the Biba access policies to grant access from a subject to an object. Our model allows for the use of the ring policy, which does not restrict the reads a subject can perform, and also for the use of the strict integrity policy because we might need to protect some subjects by restricting that the reads take place only in the form of HQR. (This will be decided based on the importance of the entity by the security officer).

The Biba ring policy states that:

1. Any subject can read any object regardless of the integrity level of the object or subject.
2. \(s \in S\) can write to \(o \in O\) if and only if \(o_{il} \leq s_{il}\).

The Biba strict integrity policy states that:

1. Simple property: \(s \in S\) can read \(o \in O\) if and only if \(s_{il} \leq o_{il}\).
2. Star property (*): \(s \in S\) can write to \(o \in O\) if and only if \(o_{il} \leq s_{il}\).
In our model we have a Reach List for each subject based on its integrity level. The subject is provided with a Reach List $RL$ containing all objects it can reach for access. Objects in this list may have different integrity levels, and when the subject submits a request to the policy enforcer it makes the decision to grant the access or deny it based on the integrity level of the subject compared to the object and that the object is in its reach list also the type of access (read or write).

3.5.2 Read Access:

In our model we allow for a subject to read any object as long as the object matches the one in the subject’s $RL$. The possible effect of that read is on the modification factor of the subject. Also, we can place that subject under a HQR, which restricts that the subject reads only from equal or higher integrity level objects due the importance of that subject. So the ring policy in case of reads is modified to handle these requirements. When the security policy enforcer receives the read request the following takes place:

1. The policy enforcer checks object is in the $RL$ of the subject.
2. If the object is part of the $RL$, the subject is allowed to proceed and read the object.

Our model rewards for reading from higher level objects. We don't restrict that because the information at higher-level integrity objects has a better quality and can impact positively the quality of data at that reading subject.

Any access taking place at the same level of integrity, regardless if it is read or write, has no impact on the integrity level or the modification factor.

We use the modification factor in our model as a metric to provide the dynamic change in an integrity level by assessing the modification a subject has from previous accesses to other entities.
at different integrity levels. It is expected that an entity can have a high integrity level associated with a high modification factor. Our model studies the previous behavior of an entity and it is recorded in the modification factor that is associated with each entity to monitor for a maximum or minimum modification factor level that could lead to a change in integrity level for that entity.

After a read access takes place, the $MF$ of the subject has to be recalculated and the following rules are applied to do that:

- A subject having a higher integrity level than the object, i.e., after $LQR$ that subject gets its $MF$ changed by:

\[ S_{NEW\_MF} = (s_{ill} - o_{il}) + S_{Current\_MF} \]

- If the new $MF$ is equal or greater than $M$, which is the maximum, the integrity level is changed one level down and starting with a Middle $MF$ on the new integrity level. Otherwise, the same integrity level is kept with the new $MF$ until it reaches a maximum or minimum.

If the subject accesses an object with the same level of integrity then nothing happens in terms of changes to the $MF$.

If it is the case where the subject $s$ is under HQR only access then when the security policy enforcer receives the read request the following takes place:

1. The policy enforcer checks to ensure that the object is in the RL of the subject.
2. For $s \in S$ and $o \in O$, $s$ can read $o$ if and only if $o_{il} \geq s_{ill}$

After the $HQR$, the subject’s $MF$ changes by decreasing the current $MF$ value by one. We do this to make it slow for the subject to recover allowing it to build trust slowly. If the $MF$ is equal
or less than $-M$ which is the minimum $MF$, we change the integrity level of the subject to one level up and start the $MF$ for the new level at middle. Otherwise keep the same integrity level with the new $MF$.

3.5.3 Write access:

For the write access we follow Biba star property. When the policy enforcer receives the write access request it performs the following steps:

1. For $s \in S$ and $o \in O$, $s$ can write to $o$ if and only if $s_{ll} \geq o_{ll}$

Under some cases we might want to monitor objects as well and evaluate if they also can change their integrity levels since they have been written to by a higher integrity level subject like in the open collaborative author systems. In such systems the subjects are the authors and the objects are the documents that are modified. Therefore, we propose an optional procedure where the $MF$ of the document (object) is reduced each time the document is written to by a higher integrity level subject. We use the following formula:

$$o_{NEW\_MF} = o_{Current\_MF} - (s_{ll} - o_{ll})$$

If the new $MF$ is equal or less than $-M$, which is the minimum, the integrity level is changed one level up and starting with a Middle $MF$ on the new integrity level. Otherwise, the same integrity level is kept with the new calculated $MF$.

Our model studies the behavior of the entities to change the integrity level of an entity dynamically. We reset the $MF$ at each new integrity level; regardless whether the last integrity level change was up or down, in the middle to avoid another change in integrity level quickly.
Going up in integrity level is done slowly, and this is why our $MF$ is decreased by one level at a time to prevent a subject from reaching higher integrity levels quickly. If an entity manages to change its integrity level quickly then it might be the case where it is a malicious one with the goal to infect other entities with malicious content. All the actions conducted are entered into a log to make it easy for tracing the changes in integrity levels. In the following chapter we provide an implementation of our model.
CHAPTER 4

Implementation

4.1 The implementation:

To demonstrate our model we developed a web application where people can work on documents collaboratively. The idea is to have a user create a project and then invite others to join him/her via email so they all can work together on the project. We used PHP, MySQL and JavaScript in building the website.

4.2 Assumptions:

Before we get into the detail, we mention a few assumptions in addition to previously mentioned:

- We assume that the environment we target is one where people can work in a collaborative way to accomplish tasks.
- We assume that users can read data up or down. However, the project manager is under a protected read so he/she can filter out information in the documents without being contaminated with it.
- Users are aware and warned that if they choose to access information at lower integrity levels then their integrity levels will be impacted accordingly.
- We assume that users in our system are the subjects and documents are the objects.
- Any user can have different integrity levels and different MFs with each different project.
• We assume that an impact of LQR happens only once per session per document to prevent the integrity level falling too quickly.

• We assume that an impact of HQR happens only once per session per document to prevent malicious users from accessing high level integrity quickly.

Next we describe how the website works. A user would have to register to start to work on the website; once they do that they can create a project by providing a name of the project. Then members are invited via email. We used six levels to reflect the integrity level of the user. Based on the expertise level the manager assigns, we make it into an integrity level. So level 1 is the lowest integrity level and six is the highest.

The manager is always given a level of integrity equal to six and a MF equal to zero. During this phase of testing we chose to protect the manager under the assumptions that he/she can filter out information based on the fact that they are the project managers.

Each invited member user starts their integrity level as indicated by the manager and the MF at 5, which is the middle. We ranged our MF to be from 0 to 10 where 0 is the lowest and 10 the maximum. We did mention in chapter 3 that the MF is from –M to M and 0 is the middle. In our implementation we adjusted that by adding 5 to view the changes in MF on a scale of 10.

After the invitation to users is completed, the manager then can specify tasks for each member indicating what they are supposed to do or what tasks they are assigned to complete. Each member user can create a document that will take their integrity level and a MF of 5 inside that same project.

A list of documents associated with each project are visible to the user members with access modes next to each document based on its integrity level. For instance documents with higher
integrity levels will have a view only link next to them. If the document integrity level is equal to or less than the user integrity level, then the user can view or modify the document.

Based on the integrity level of the member user, they can view or modify the document at the same integrity level with no impact on their integrity level or MF. However, if the document is at an integrity level lower than the user, then that user can modify the document or view it as a LQR. If the user chose to view the document, then their MF is recalculated based on this formula:

$$User_{NEW,MF} = (User_{il} - Document_{il}) + User_{Current,MF}$$

Then we test the new calculated MF to see if it greater than or equal to 10. If it does, we change the integrity level of the user one level down with a new MF equal to 5; otherwise, keep the same integrity level with the new calculated MF.

If it is the case the document integrity level is greater than the member user, then the user MF is recalculated by reducing it one level. We do this to prevent a quick integrity level change up, and this is also why we calculate such access only once per session per document. This way we can slow down a malicious user from accessing a higher integrity level with intentions to do damage.

In the case of writes, the user member has to be at a higher integrity level than the document. Then we recalculate the MF for the document as follows:

$$Document_{NEW,MF} = Document_{Current,MF} - (User_{il} - Document_{il})$$
If the document new MF less than or equal to 0, we change the integrity level for that document to one level up and change the MF to 5. Otherwise, we keep the same integrity level with the newly calculated MF.

We ran the test by creating a project and inviting 4 users. After testing with different integrity levels and creating different documents, we saw how the policy adjusts the reach list and expand or minimizes the access to these documents based on the integrity levels.

We also noted that the change to higher integrity levels is happening in a slow manner allowing only trusted users and documents reach such levels. If a user accesses lower level documents, then they can increase the MF fast but the recovery is slow. So malicious users will not have the chance to reach higher levels quickly.

4.3 Application:

We have a project called, “Accounting Application” where the manager is Tom. Figure 4 shows the project list that Tom is part of, and we can see that next to each project that Tom is the manager of is a link to invite project members.
Tom invited the following people and assigned each one of them an integrity level (IL). Mike with IL = 2, Bob with IL = 3, Alice with IL = 3 and John with IL = 5. Each one of them, after signing into the website, uploaded a file having their name and IL. In figure 5 we see the invite form and the assignment of the IL and the MF to each member. Figure 6 shows the project Tablet interface with all members of the project. The screen is from Mike under the skill level you can see that he has an Integrity Level (IL) = 2 and a starting MF = 5.
Figure 5 Tom invites and sets the IL and MF for a member via Email

Figure 6 Mike IL and MF levels along with the documents in project Accounting Application
After that as in figure 7, Tom assigned some tasks to the group members. He started with John by assigning two tasks for him to complete. John can view the tasks as in figure 8 but he cannot delete them. Only Tom the manager can do this as in figure 9.

**Figure 7** Tom assigns a task to John

**Figure 8** The list of tasks assigned to John
**Figure 9** Only Tom the manager can only create and delete the tasks

A user can view or modify the documents at the same level with no impact on their integrity level or MF. However, if the document is at an integrity level lower than the user, then that user can modify the document or view it in the form of a LQR. In figure 10, John has an IL = 5 and wants to view a document created by Mike that has an IL = 2. If the John chooses to view the document, he is warned as in figure 10. If he chooses to proceed his MF is recalculated based on this formula:

\[ User_{NEW, MF} = (User_{IL} - Document_{IL}) + User_{Current, MF} \]
Figure 10 John is warned not to proceed reading an LQR document created by Mike.

Then we test the newly calculated MF and see if it greater than or equal to 10. If it does, then we change the integrity level of the user one level down with a new MF equal to 5; otherwise, we keep the same integrity level with the newly calculated MF. John’s newly calculated MF is increased to 8. After multiple sessions of access to lower IL documents, John’s IL is changed to one lower IL level as in figure 11 and figure 12. You can see the difference taking place to access modes next to each document based on the behavior of user John.
Figure 11 John’s current IL 5 and the access modes next to each document.

Figure 12 John’s IL changed to one level down after multiple LQR
If it is the case that the document integrity level is greater than the user level then the user MF is recalculated by reducing it one level at a time. We do this to prevent a quick integrity level change up, and this is also why we calculate such access only one per session per document. This way we can prevent a malicious user from accessing higher integrity level with intentions to damage. Figure 13 shows the reduction of the MF by one for user Alice. In figure 14 we changed the level one level up after the MF reached the lowest possible.

**Figure 13** Alice’s MF reduced by one due to HQR.
For the documents in the project, each time the document is updated by a higher level integrity member the MF is reduced by one until we are ready to change the IL of the document to the following level up. Figure 15 shows a document MF reduced by one after it has been updated by member John.
We tested our model with different scenarios to see what happens, and we found that our model does capture users’ behaviors and history to enable us to change the integrity levels dynamically. It also makes it hard for a malicious user to reach higher levels quickly since the MF is reduced by at most one each time.
5.1 An RBAC extension

In this chapter we present an extension to our model that we base on the RBAC model approach. We provided an introduction to RBAC in section 1.7 from chapter 1. We do not tie our model to work only with RBAC, but we demonstrate how our model can be extended to be used in different access control mechanisms.

In chapter 3 we presented our model to change the integrity dynamically. Our generic model assumes that some entities are under protected read and sometimes lower quality data can be filtered out. The example we provided is a collaborative environment where the subjects (users) can identify lower quality data and are warned before they access such data.

There are situations where all subjects are in an environment where they do not have the protected read option. For example a work environment where workers view and distribute work documents that are published by other workers, which they know nothing about. So there has to be a way to protect the users and limit how far they can read down based on their role description. Under such environments, it can be harmful to access lower quality data.

So we introduce an extension to our model based on the RBAC model. We do not limit our general model with RBAC; however, in this section we use RBAC as an extension to implement
the original model with additions. Our generic model is applicable and can be extended with other access control models if the following requirements are meet:

- The environment to control integrity is enhanced if we monitor the integrity levels dynamically. It is the case in some environments that dynamic change is not desirable. Then we do not use our model in these cases.
- In the environment it is helpful to evaluate the trustworthiness of a given entity. This is needed to block or expand the access of such entity.

Next, we present terminology that we are going to use in our model in addition the terminology presented in chapter 3:

- MLL: The maximum lower integrity level difference for a read access that a subject is allowed to reach.

Our model is based on the RBAC model. In [21] the work the authors present is to protect privacy in relational databases. Even when the model is to protect privacy, the authors present an extension to the basic RBAC model with the introduction of attributes, which we also use in the extension of our model. Not only does every entity belong to a role, but also that role can be further customized by the addition of attributes. For instance we can have the role of an employee but an employee, working in the E-Marketing department has attributes to distinguish it from a general employee.

These attributes could be ones allowing the employee to access E-Marketing documents that other employees cannot access like customers’ personal information.
The RBAC model in [21] is constructed as a tree. The root of the tree is the general role, for example, an employee role. The children of the parent have more attributes to further customize the role. These attributes are inherited in the role from parent to child in the form of permissions granted to the parent for that given role. In our model we are going to use some of the math notation regarding the RBAC model as presented in [21].

In [21] each role has a set of attributes that are pre-assigned and provide specific description for that given role. These attributes are inherited by the child from the parent. These attributes are in addition to other roles attributes that the child might have. The attributes are defined by the system administrator at the time of role creation. When an entity is assigned a given role, it is assigned all attributes associated with that role in addition to attributes that are inherited from the entity’s parent. The access control system uses this information to grant or deny the access, as we will describe later in this section.

We introduce the following definitions similar to [21]:

Definition 1: A set of roles is denoted as \( R \) having all possible system roles where \( r \in R \) and organized as a tree where the parent is the most general role and the children are more customized role inheriting its parent general role.

For example: An organization can have the general roles of Marketing, E-Marketing, Employee, Admin, and Secretary. Under the parent general role E-Marketing, for example, we can have a role of E-Mail marketing which has all the attributes of the role E-Marketing in addition to its own role.
Definition 2: Every role $r \in R$ has a set of attributes denoted as $r.\text{Attributes} = \{r.\text{attr}_1, \ldots, r.\text{attr}_n\}$. At the creation of the role $r$ these attributes are defined or inherited from the ancestor for each created $r \in R$.

*For example:* For instance the role Employee can be further customized by adding attributes to it by having an ID and Department to make it an E-Marketing employee. So the E-Marketing employee inherits the attributes of the role employee like the name and ID in addition to having additional attributes of the E-Marketing role.

Next, we want to restrict how far a subject can read down by introducing the maximum lower integrity level difference concept. It is assumed that data at high integrity levels are of high quality and more trust worthy. Our model with LQR does not restrict reads. However, we may want to have some level of control on how far a subject can read down due to the defined role of that subject because data at lower levels are assumed to be of lower quality. A read down can impact the integrity level of that subject by impacting the modification factor, and that can have an impact on the overall integrity level of that subject.

Definition 3: Maximum lower level MLL: for each subject $s \in S$, based on its defined role, it is allowed to access objects for the reason of reading where the object has a maximum lower integrity level difference $N$ from the subject.

*For example:* Subject having the role of E-Marketing has an integrity level of 5 and based on its role definition is allowed to read only three integrity levels down as a maximum. So that subject can read down to objects with integrity level 2, which is at the maximum difference of $N = 3$.

Definition 4: (Reach List): For each subject in our model based on its defined role, that subject is provided with a Reach List $RL$ containing all other objects it can reach for access. Objects in this
list may have different integrity levels, and when the subject submits a request to the policy enforcer it makes the decision to grant the access or deny it based on the integrity level of the subject compared to the object and that the object is in its reach list and also the type of access (read or write).

*For example:* The subject *RL* can be objects in (E-Marketing, Admin).

5.1.1 Read Access

In our model we allow for a subject to read any object as long as the object matches the one in the subject’s *RL* based on its role *r*. The immediate effect of that read may be on the modification factor. However, we do restrict the lowest integrity level a subject can go down to read an object as long as it is at maximum difference MLL. This lowest level is determined by the subject’s integrity level and the MLL. Also, we can place a subject under a HQR, which in terms restrict that the subject reads only from higher or equal integrity level objects due the importance of that subject. The ring policy in case of reads is modified to handle these requirements. When the security policy enforcer receives the read request the following takes place:

1. The policy enforcer checks that $s_{RL}$ matches $o_R$
2. The policy enforcer makes sure that $o_{il}$ is at maximum MLL $N$ defined for that subject $s$.
3. If 1 and 2 are true then the subject $s$ can read the object $o$, otherwise no access is allowed.
Our model rewards for reading from higher level objects. We don't restrict these because the information at higher-level integrity objects has a better quality and can impact positively the quality of data at that reading subject.

Any access taking place at the same level of integrity regardless if read or write has no impact on the integrity level.

We use the modification factor in our model as a metric to guide the dynamic change in integrity levels by assessing the modification a subject has from previous accesses to other entities at different integrity levels.

After a read access takes place, the MF of the subject is recalculated and the following rules are applied to do that:

- A subject having a higher integrity level than the object where the object is at maximum MLL N after reading (LQR) that subject gets its MF changed by:

\[ s_{NEW,MF} = (s_{il} - o_{il}) + s_{Current,MF} \]

- If the new MF is greater than or equal to M, which is the maximum, the integrity level is changed one level down, and the MF is set to the middle on the new integrity level. Otherwise the same integrity level is kept with the new MF until it reaches a maximum or minimum.

- If the subject integrity level reaches the MLL N, then that subject is blocked from further reading down. However if that subject tries to access lower integrity level objects their
$MF$ is increased by one level and it might lead it to lose in terms of integrity level if such behavior persists by having its integrity level falling down.

If the subject accesses an object with the same level of integrity then nothing happens in terms of changes to the $MF$.

If it is the case that the subject $s$ is under HQR only access, then when the security policy enforcer receives the read request the following takes place:

3. The policy enforcer checks that $s_{RL}$ matches $o_R$.
4. For $s \in S$ and $o \in O$, $s$ can read $o$ if and only if $o_{il} \geq s_{il}$
5. If 1 and 2 are true then the subject $s$ can read the object $o$, otherwise no access is allowed.

After reading that object ($HQR$) the subject’s $MF$ changes by decreasing the current $MF$ by one. We do this to make it slow for the subject to recover, forcing it to build trust slowly. If the $MF$ is less than or equal to $-M$ which is the minimum $MF$, we change the integrity level of the subject to one level up and start the $MF$ at the middle.

5.1.2 Write access

For the write access we follow Biba star property. When the policy enforcer receives the write access request it performs the following steps:

2. The policy enforcer checks that $s_{RL}$ matches $o_R$.
3. For $s \in S$ and $o \in O$, $s$ can write to $o$ if and only if $s_{il} \geq o_{il}$
4. If 1 and 2 are true then the subject $s$ can write to the object $o$, otherwise no access is allowed.
Under some cases we might want to monitor objects as well and evaluate if they also can change their integrity levels since they have been written to by a higher integrity level subject like in the open collaborative author systems. In such systems the subjects are the authors and the objects are the documents that are modified. Therefore, we propose an optional procedure where the MF is reduced each time the object is written to by a higher integrity level subject. We use the following formula:

\[
o_{NEW,MF} = o_{NEW,MF} - (s_i - o_i)
\]

If the new MF is less than or equal to -M, which is the minimum, the integrity level is changed one level up, and the MF is reset to the middle MF value with the new integrity level. Otherwise, the same integrity level is kept with the new calculated MF.

We keep track of the number of successful writes W the subject did. The value of W is decided by the security officer based on the role definition when created. This is rewarded by the reduction of the MF by one level after a number of successful W’s. If the new MF is less than or equal to -M, which is the minimum, the integrity level is changed one level up, and the MF starts with middle MF value on the new integrity level. Otherwise, the same integrity level is kept with the newly calculated MF.

Our model studies the behavior of the entities to change the integrity level of an entity dynamically. We reset the MF at each new integrity level, regardless whether the last integrity level change was up or down, to the middle value to avoid another change in integrity level too quickly. Going up in integrity level has to be done slowly to prevent malicious entities from reaching higher integrity level quickly, and this is why our MF is decreased by one level at a time. This is due the fact that data at lower levels have lower quality and less trust. If an entity
manages to change its integrity level quickly then it might be the case that a malicious entity could infect other entities with malicious intent. All the actions conducted are entered into a log to make it easy to trace the changes in integrity levels.
Conclusion and Future work

In this dissertation we presented a dynamic integrity model. The motivation to investigate integrity is the fact that it was not a focus of access control for a while because early research in access control was driven by military requirements. For the military, confidentiality is the focus so the multilevel security solved many of the security requirements.

Commercial requirements had different needs. This led researchers like Biba to introduce the rules to protect integrity. As we have seen, confidentiality is about preventing disclosure where integrity is the prevention of improper modification. The integrity of data can reflect how much trust we can place in it. It also can reflect the trustworthiness an entity has. We have shown that many of the models are static in nature. Web applications and more people using online collaborations presented a challenge to trace the integrity of objects and subjects.

We introduced our model that takes into account the behavior of subjects and the history of these subjects to change their integrity levels dynamically. We use a metric we call the modification factor to trace how much taint (modification) an object or subject has along with defined rules to change the integrity dynamically.

We did point out that Biba presented an audit policy in chapter 1, however it tracks the contamination but does not change the integrity level and the goal is to change the integrity level dynamically. We also presented the low-water mark policy, and we mentioned that subjects
cannot recover after accessing lower level objects. Our model allows for both the trace of the contamination of entities and uses that history information to change the integrity levels dynamically.

We presented an application that is a collaboration website where a user creates a project and then invites others to work with him/her on the project. We used the expertise level as an indication of the integrity level. We tested and saw that the user levels changes according to their actions.

Our experiments did show that the behavior of the subject does impact their integrity level. Our model does make it hard for a subject to reach higher integrity levels quickly in order to hinder malicious users from corrupting higher-level objects. It also captures bad access behavior quickly to prevent corruption from happening to other objects.

We introduced an extension based on the RBAC approach. We extended our model by combining the generic model and an RBAC model presented in [21]. We were able to customize the roles to fit certain requirements. We did this to point out that our model can be implemented with other access control models and not to suggest that our model works only with RBAC.

For the future work we plan to implement the RBAC extension as presented in chapter 5. We also want to test our model in a cloud environment setting and in a distributed systems environment. Also we would like to explore the inclusion of our model into other existing access control models.

We also plan to study the human behavior since our model keeps track of the history of an entity. We want to use this history in order to spot possible malicious intent and prevent an attack.
References:


[52] Smith, Sean W. "Security and Cognitive Bias: Exploring the Role of the


APPENDIX A

List of definitions

- **Integrity:** Refers to keeping data consistent, accurate, and trustworthy over its life cycle in the system. It also refers to protecting data against improper modification from unauthorized entities and against change during transit.

- **Confidentiality:** Protect sensitive information from improper disclosure. Only entities with necessary security credentials can view the sensitive information.

- **Availability:** Maintain the hardware and software by performing required maintenance and keeping the system up to date to prevent possible failures. It also involves placing alternative plans to keep the system running in case of failure to allow access to it.

- **Security Policy:** A document that describes how an organization plans to protect its IT infrastructure from security threats. This document is updated to keep up with changing security requirements.

- **Access Control:** Security features that describe who can and cannot access resources inside the system. It describes the procedures to allow or deny access in order to protect the system from unauthorized access.
• Access Control List (ACL): A list with access control entries (ACE) (Objects). For each ACE it identifies which subject is trusted with access and what access rights it has over that object.

• Access Matrix: An access matrix has rows representing subjects and columns representing objects with the rights of subjects over each object as the mapping in each cell.

• Intrusion Detection: An IDS collects and analyzes information within a computer network to spot possible security breaches.

• Access Control Service: Provides protection to system resources against unauthorized access.

• User: A person, organization entity or process that accesses a system whether that access is authorized or not.
APPENDIX B

Figures from the implemented model

![Sign Up Form](image)

**Figure 16** Sign up form to create an account

A user must register first to create and invite people to join their project. This is the form used as presented in our website. The information provided is stored in our database.
After the account creation, the user has to login to start using the website by joining a project if invited or creating another new project.
After clicking on the New Project link, the user can create a project by providing a project name in the system.

**Figure 18** Creating a new project
The user can update their Email and password by clicking on the account update link.

**Figure 19** Account information update
Figure 20 Requesting account password via Email

In the case of password reset, the user needs to click on the forgot password link and provide the Email used in the registration process.
Figure 21 The Manager creating a task for a member of the team

This is the screen where Tom the manager creates tasks for his team members. He is the only person that can creates and deletes the tasks.
The user John can view the tasks assigned to him and it is visible that he can view the tasks only and he is not allowed to modify or delete the task.
After the user login, when they click on the projects tab they are provided with a list of projects they manage. Next to each project name is an invite link where they provide the Email to invite another user.
APPENDIX C

List of abbreviation used

- S: Subject.
- O: Object.
- I: Integrity.
- IL: Integrity level.
- RL: Reach List.
- MF: Modification Factor.
- MAC: Mandatory Access Control.
- DAC: Discretionary Access Control.
- RBAC: Role Based Access Control.
- Or-BAC: Organizational Based Access Control.
- MLL: Maximum Lower Level.
- ACL: Access Control List.
- HQR: High Quality Read.
- LQR: Low Quality Read.
- CD: Company Dataset.
- COI: Conflict of Interest.
- CDI: Constrained Data Items.
- UDI: Unconstrained Data Items.
- IVPs: Integrity Verification procedure.
- TPs: Transformation Procedure.
- SIP: Strict Integrity Policy.
- CIs: Critical infrastructure.
- SLOs: Single Level Objects.
- MLOs: Multi Level Objects.
- TE: Type Enforcement.
- DDT: Domain Definition Table.
- DIT: Domain Interactive Table.
- TACSIF: Trust-Oriented Access Control Based on Source of Information Flow.
- DIF: Destination of Information Flow.
- TIL: Threshold Integrity Level.
- CIL: Current Integrity Level.
- TAAC: Trust-Aware Access Control Model.
- IDS: Intruder Detection System.