Role-Based Access Control Administration of Security Policies and Policy Conflict Resolution in Distributed Systems

by

Stephen Sakawa Kibwage

A Dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

in

Information Systems
Graduate School of Computer and Information Sciences
Nova Southeastern University
2015

We hereby certify that this dissertation, submitted by Stephen S. Kibwage, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Peixiang Liu, Ph.D. Chairperson of the Dissertation Committee	Date
Gurvirender Tejay, Ph.D. Dissertation Committee Member	Date
James Cannady, Ph.D. Dissertation Committee Member	Date

Approved:

Eric S. Ackerman, Ph.D. Dean, Graduate School of Computer and Information Sciences	Date

Graduate School of Computer and Information Sciences

Nova Southeastern University

2015

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Role-Based Access Control Administration of Security Policies and Policy Conflict Resolution in Distributed Systems

by
Stephen S. Kibwage
January 2015

Security models using access control policies have over the years improved from Role-based access control (RBAC) to newer models which have added some features like support for distributed systems and solving problems in older security policy models such as identifying policy conflicts. Access control policies based on hierarchical roles provide more flexibility in controlling system resources for users. The policies allow for granularity when extended to have both *allow* and *deny* permissions as well as weighted priority attribute for the rules in the policies. Such flexibility allows administrators to succinctly specify access for their system resources but also prone to conflict.

This study found that conflicts in access control policies were still a problem even in recent literature. There have been successful attempts at using algorithms to identify the conflicts. However, the conflicts were only identified but not resolved or averted and system administrators still had to resolve the policy conflicts manually. This study proposed a weighted attribute administration model (WAAM) containing values that feed the calculation of a weighted priority attribute. The values are tied to the user, hierarchical role, and secured objects in a security model to ease their administration and are included in the expression of the access control policy. This study also suggested a weighted attribute algorithm (WAA) using these values to resolve any conflicts in the access control policies. The proposed solution was demonstrated in a simulation that combined the WAAM and WAA. The simulation's database used WAAM and had data records for access control policies, some of which had conflicts. The simulation then showed that WAA could both identify and resolve access control policy (ACP) conflicts while providing results in sub-second time. The WAA is extensible so implementing systems can extend WAA to meet specialized needs. This study shows that ACP conflicts can be identified and resolved during authorization of a user into a system.

Acknowledgements

This dissertation has been a long journey from the application to join the program until now. I thank God who created heaven and earth for giving me faith to persevere. For my wife, Debbie, who sacrificed much to accommodate my being a student for all that time. To my kids Joel and Ella who just knew that “daddy had homework” and could not wait for him to be done. To my parents, Samuel and Frida Kibwage, who have yearned for me to complete my dissertation and for always believing it was possible. My other family members have been supportive every step of the way with their prayers to God and their well wishes.

I would like to express my gratitude to my dissertation advisor, Dr. Peixiang Liu, for working with me to the finish line. His guidance, timely encouragement, and responsiveness through the processes were invaluable. I would also like to thank my dissertation committee members, Drs. James Cannady and Gurvirender Tejay, for their help and support in carefully reviewing my work. I would also like to thank Dr. Ackerman for providing guidance in the period transitioning from a student to beginning the dissertation process. I truly am grateful.

I would like to acknowledge my employer Web.com, Inc. for their financial support when called for. Special thanks to my friend Dr. Diana Makombe for being in my corner and cheering me on having gone through the process herself. I would also like to thank the friends who have had encouraging words and kinds deeds to help get me through.

Table of Contents

Abstract iii

Table of Contents v

List of Tables viii

List of Figures ix

Chapters

1. Introduction 1
- Background 1
- Problem Statement 2
- Dissertation Goal 5
- Research Questions 6
- Relevance and Significance 7
  - Groups Affected by ACP Conflict 7
  - Benefits of Resolving ACP Conflicts 7
  - Promise of Resolution 8
  - Addition to Knowledgebase 8
  - Generalizability of Results 9
  - Originality 10
- Barriers and Issues 11
- Limitations and Delimitations 12
  - Limitations 12
  - Delimitations 13
- Definition of Terms 14
- Summary 15

2. Review of the Literature 16
- Introduction 16
- Early RBAC 16
  - Mandatory Access Control 16
  - Discretionary Access Control 16
  - Role-Based Access Control 17
- Improvements over RBAC 17
  - Administrative RBAC ‘97 17

Organizational RBAC 18
Integrated Policy 18
Generalized Temporal RBAC and Geographical RBAC 19
Concrete and Abstract RBAC 19
PolyOrBAC 20
Administrative RBAC 20
NBAC and ZBAC 21
RBAC96 21
Spatially Aware RBAC 22
UARBAC and ACPCDM 23
Residual Problems 24
Policy Conflicts 24
Global Policy 25
ACP Conflict in Collaborative Systems 26
Access Conflict Resolution 28
Adaptation of Results 28
Summary 29

3. Methodology 30
Overview 30
Research Method 30
Designing the Weighted Attribute Algorithm 31
Designing Allowance for Extensible Algorithm 34
Designing the Weighted Attribute Administration Model 35
Consideration for the Distributed System Environment 37
Designing the Simulation Environment 39
Preparing the Seed Data used for Runs through WAA and ACPCDM 41
Collecting Metrics from Runs through WAA and ACPCDM 43
Formats for Presenting Results 47
Resource Specifications 48
Hardware 48
Software 48
Summary 50

4. Results 51

Data Analysis 51

General Observations of the Seed Data used for the Runs 51

Performance of the Algorithms 53

Accuracy in Identifying Conflicts for WAA and ACPCDM 55

Accuracy in Resolving Conflicts using WAA 55

Findings 56

The Delay added by using WAA in User Authorization 57

User Benefits of using WAA 57

Administrator Benefits of using WAA and WAAM 58

Summary of Results 58

5. Conclusions 59

Overview 59

Outcomes 61

Research Question 1 61

Research Question 2 62

Research Question 3 63

Recommendations and Future Research 64

Implication 65

Study Limitations 65

Summary 66

Reference List 67

Appendixes 71

Appendix A: Database Create Tables' Script 72

Appendix B: Database Create View Script 74

Appendix C: Database Create Stored Procedures' Script 75

Appendix D: Tools and Components for Weighted Attribute Algorithm Implementation 84

Appendix E: Authorization Log from Running the Weighted Attribute Algorithm 85

Appendix F: Authorization Log from Runs through ACPCDM 88

Appendix G: List of a User's ACP used by WAA 91

Appendix H: Code to Generate the Data for the Simulation 96

List of Tables

Table 1. The Values Collected from each Run	43
Table 2. The Collection of Metrics from the Simulation	44
Table 3. Example of ACPs in conflict	51
Table 4. The Common Summary Metrics Collected from the Runs	52
Table 5. The Metrics Collected from the Comparing the Algorithms	52
Table 6. Comparison of Delay added to Authorization Process	53
Table 7. The Summary Metrics Collected from the Weighted Attribute Algorithm	54
Table 8. The Delay Added to Authorization by WAA by on User ACP Count	54
Table 9. Accuracy Comparison of ACP Conflict Identification	55
Table 10. Overall Accuracy of ACP Conflict Resolution	55
Table 11. Accuracy of ACP Conflict Resolution by User ACP	56

List of Figures

Figure 1. Authorization Flowchart using WAA for Conflict Resolution 32

Figure 2. ERD of the Weighted Attribute Administration Model 36

Figure 3. Location of WAA in a Distributed System 37

Figure 4. Determining the Source of User ACP 39

Figure 5. Logical Design for the Simulation Environment 40

Figure 6. Design of the Simulation 41

Figure 7. The Configuration Setting when running ACPCDM 42

Figure 8. The Configuration Setting when running the Weighted Attribute Algorithm 42

Chapter 1

Introduction

Background

Role-based access control (RBAC) has served foundationally for newer models that enforce system security at various levels of improvement. Improvements are needed in administering security polices (Pérez, Lòpez, Skarmeta, & Pasic, 2010), and detecting and resolving system authorization conflicts in distributed systems (Juntapremjitt, Fugkeaw, & Manpanpanich, 2008). RBAC is a paradigm commonly accepted in enforcing system security because roles provide a way to group features and their relationships to users of a system (Sandhu, Coyne, Feinstein, & Youman, 1996; Vaidya, Atluri, Warner, & Guo, 2010). It shows improvement over earlier IS security approaches like discretionary access control and mandatory access control. RBAC has been used as a foundation of studies and has been extended into various models such as CABAC (Concrete and Abstract Based Access Control) introduced by Bouzida, Logrippo, and Mankovski (2011), and Generalized Temporal Role-Based Access Control (GTRBAC) by Joshi, Bertino, and Ghafoor (2005). These are also different from Park and Sandhu's (2004) UCON_{ABC} (usage control) which has found success in the business to consumer space. There is El Kalam et al.'s (2003) Organization based access control (OrBAC) which uses RBAC as one of its pillars and has proved useful in geospatial research. These studies show the acceptance of RBAC and the research that has gone to further refine it.

Within RBAC-based research, policy conflict occurs when the policies satisfied by the authorization in a system have actions that are contradictory. If the contradictory access rights are granted to an individual entity, such as a system user, then that user will experience the effects of policy conflict which are anomalous system behavior (Jajodia, Samarati, Sapino, & Subrahmanian, 2001).

Policy conflict has since been researched as shown by Wu, Chen, Zhang, and Dai (2009) and Fan, Liang, Luo, Bo, and Xia (2011). Wu et al. (2009) compare user security policies in *matrix groups* by using an algorithm against the matrices to detect security policy conflicts. According to Wu, et al. (2009), a matrix group is a policy definition represented as a matrix. Whenever policies need to be compared to determine conflict, they perform matrix operations to determine the conflict. Fan, et al. (2011) propose an algorithm, ACPCDM (Access Control Policy Conflict Detection Model), to review the policies and identify those that have conflicts. This study goes further than the identification of ACP (Access Control Policy) conflict done by Wu et al. (2009) and Fan et al. (2011) by finding a way to resolve the policy conflicts using a security administration model and an extended algorithm.

Problem Statement

According to Wu, Chen, Zhang, and Dai (2009), ACP conflicts occur in situations where overlapping event conditions or actions end up being contradictory. Fan, Liang, Luo, Bo, and Xia (2011) identified the conflicts as disaccords in the roles that a user is assigned. The access control policy (ACP) conflicts occurring in RBAC based studies fall into two options; cyclic inheritance and separation of duty (Shafiq, Joshi, Bertino, & Ghafoor, 2005).

RBAC based systems in a distributed environment could end with conflicts in their ACPs which would cause a system to behave erratically as it relates to security

(Shafiq, Joshi, Bertino, & Ghafoor, 2005). Other studies (Fan, Liang, Luo, Bo, & Xia, 2011; Wu, Chen, Zhang, & Dai, 2009) were also in agreement that ACP conflicts occur in systems and should be resolved. However, attempts to resolve ACP conflicts resulted in the automation of detecting conflicts for system administrators to resolve (Fan, et al., 2011, El Kalam, et al., 2003, Wu, et al., 2011). Shafiq, et al. (2005) proposed a solution that would be external to existing systems but would require policy integration and a homogenizing process to resolve conflicts.

Bertino, Catania, Ferrari, and Perlasca (2003) proposed a framework within which their algorithm would resolve conflict. The framework containing their process generally required that a mapping from an existing system be done and the ACP data imported into their tool for comparison to identify ACP conflicts. Once the data was in their framework, the ACPs were then compared using C-Datalog, an object oriented programing language developed by Greco, Leone, and Rullo (1992). So Bertino, et al. (2003) identified the problem as worth researching but provided an external solution that required data mapping from existing security models into their framework. Bertino, et al. (2003) stated that ACP conflict was a problem but did not attempt to solve it in their model; instead they cede to the specifications of the system.

In more recent research, ACP conflicts in RBAC systems have been studied (Fan, et al., 2011; Wu, et al., 2009) showing that ACP conflicts are still worth pursuing. The studies by Fan, et al. (2011) and Wu, et al. (2009) are close to this problem though these studies' results are limited to identifying the conflicts for administrators to correct. Fan, et al. (2011) algorithm identifies conflicts and also provides detail of the conflict to assist the administrators in resolving the conflict. The algorithm proposed by Wu, et al. (2009)

to identify conflict suggested an implementation that is abstracted from the secured system so that updates to the system were made independently from the security algorithm. The algorithms did not offer anything beyond identification of ACP conflict. Perez, Lòpez, Skarmeta, and Pasic (2010) provided a XAML (Extensible Application Markup Language) –based solution to administer the ACP of a system but neglected identifying ACP conflict or providing any options for ACP conflict resolution. This study proposed improvements to access control by extending the administration of policies shown by Pérez, Lòpez, Skarmeta, and Pasic (2010) and Li and Mao (2007); and detecting and resolving conflicts for system authorization in distributed systems (El Kalam, Deswarte, Baïna, & Kaaniche, 2007).

ACP conflicts occur organically in systems over time with cumulative demands for information, features, and data access (Fan, Liang, Luo, Bo, & Xia, 2011). These ACP conflicts though identified must be resolved manually to maintain the usefulness of the system. Resolving conflicts manually is burdensome for administrators who would have other tasks to perform. This study looked at ACP conflict in distributed systems and proposed a solution to automatically resolve the ACP conflict. There is a lack of a unified solution that includes both an administration model and an algorithm in distributed systems (Fan, et al., 2011, Oh, Sandhu, & Zhang, 2006, Wu, et al., 2009). The administration model provided attributes to be used in the algorithm because none of the existing models considered ACP conflict resolution. This study proposed a unified solution with an administration model with hierarchical roles as well as an extended algorithm to find ACP conflicts and calculate weighted priority attribute to use in resolving ACP conflict in distributed systems.

Dissertation Goal

This study brought convergence to the results from previous studies by using an administration model and an extensible algorithm to resolve ACP conflict. It proposed combining hierarchical roles that would be central to the systems at a high level of visibility into an administration model. Additionally, this study proposed that the administration model would include attributes to support a weighted priority attribute (WPA) for the algorithm to use in finding and resolving ACP conflict.

This study also proposed that the conflicts in security policies' permissions and prohibitions were avoidable but would need both an administration model of hierarchical roles with attributes to calculate WPA named *weighted attribute administration model* (WAAM); and an extended algorithm to resolve any ACP conflicts name *weighted attribute algorithm* (WAA). In this study, the values to calculate WPA were calculated using values provided by a system administrator when a user, role, or object record is created. The study by Fan, Liang, Luo, Bo, and Xia (2011) used an algorithm in the attempt to identify ACP conflict but lets the administrators resolve any conflicts. Using the weighted priority attribute would provide an expressive declaration of precedence to assist in averting conflict. However should there still be conflict beyond the weighted priority, a localized algorithm would resolve any conflict by applying prescribed checks against any attributes to be defined within the algorithm.

The new security administration model improved on the ARBAC02 (Administrative RBAC '02) presented by Oh, Sandhu, and Zhang (2006) and bind the resulting administrative model with the algorithm. ARBAC02 does not attempt to resolve ACP conflict or provide support to resolve ACP conflict. Kern, Schaad, and Moffett

(2003) argue that using role hierarchies could be problematic because of directional inheritance and inadequately defined relationships between a user's role and job function. Li and Mao (2007) overcome the issues identified by Kern, Schaad, and Moffett (2003) with comprehensive design requirements to include flexibility, scalability, acceptability, and economy of mechanism. Their solution proposes UARBAC ("Unnamed" Administrative Role-Based Access Control) which separates using a role for access control from administering a role (Li & Mao, 2007). Bruns, Huth, and Avijit (2011) built their study by extending UARBAC by simulating plan synthesis and non-atomic administration. None of these administrative models, ARBAC02 and UARBAC, addresses conflict resolution (Bruns, Huth, & Avijit, 2011; Li & Mao, 2007; Oh, Sandhu, & Zhang, 2006). This study proposed to use UARBAC (Li & Mao, 2007) as foundational research while integrating with the conflict resolution algorithm.

Research Questions

The following research questions were considered in the course of the investigation for this study. The answers to these questions were placed in the outcomes section of the conclusion.

1. What are the advantages and disadvantages of implementing WAAM in a RBAC system?
2. What are the advantages and disadvantages of an RBAC system adopting the WAA as part of their use authorization?
3. Expecting the number of conflicts a systems user has to grow how would the proposed WAA resolve ACP conflicts?

Relevance and Significance

Groups Affected by ACP Conflict

Research has shown that some solutions uncover problems besides the ones being solved. An example of this is the Sandhu, Coyne, Feinstein, and Youman (1996) study which discussed RBAC in the formative stages and resultant benefits yet identified problems such as a lack of analysis in managing role hierarchies on a unified framework. The problems identified show that administrators of systems benefiting from RBAC would be affected positively as would the users of such systems though they would also be exposed to the problems listed (Sandhu, et al. 1996). The conflicts that go unresolved would cause the system to misbehave and would affect both users and administrators as follows:

1. Users with ACP conflicts will have the system misbehave such as by denying a user access where it is expected (El Kalam, Deswarte, Baïna, & Kaaniche, 2007; Shafiq, Joshi, Bertino, & Ghafoor, 2005).
2. Administrators have to spend expensive time researching or troubleshooting their systems to find the conflicts whenever such issues arise (Shafiq, Joshi, Bertino, & Ghafoor, 2005).

Benefits of Resolving ACP Conflicts

For the results of this study wherever implemented, the expectation is that proprietors should benefit from less administrative demands, better user efficacy, and an overall simplification of authorization in the security model. For the affected users, the system would be more useful and their usage more pleasant (El Kalam, Deswarte, Baïna, & Kaaniche, 2007; Shafiq, Joshi, Bertino, & Ghafoor, 2005). The users should also have a better guarantee of access to pertinent data that would otherwise be denied in the event of an unresolved conflict (Kuang & Ibrahim, 2009).

This study anticipates these benefits for the users because the access to secured system features and objects would better reflect their administratively prescribed access to the system. The administrators who prescribe a system's access would benefit (Shafiq, Joshi, Bertino, & Ghafoor, 2005) because they can adjust the ACP proactively in response to monitoring the authorization logs and making any administrative changes without having frustrated users making inquiries.

Promise of Resolution

Reviewing the studies in the context of RBAC-based distributed systems would provide a methodology to better administer security in distributed systems in a way not addressed by studies in the reviewed literature. Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) show that using single-level roles for their ACP, conflicts can be resolved using their algorithm. Many systems use hierarchical policies to manage ACP increasing the complexity needed to administer them and to resolve conflicts that may arise (Damiani & Silvestri, 2008; Muppavarapu, Pereira, & Chung, 2010). This complexity is perhaps why most of the algorithms only go as far as finding the conflict but this study overcomes this complexity by resolving conflicts at the ACP level rather than higher in the hierarchy.

Addition to Knowledgebase

While OrBAC only suggests a solution for resolving conflict using an algorithm based on *possibilistic logic,* it does so without an administration model (El Kalam, et al., 2003). Abdunabi, Ray, and France (2013) propose using spatio-temporal constraints in determining users' ACP but do not address conflict resolution. The solution this study proposes looks to go beyond the literature by resolving conflicts by utilizing both an administration model and a complementing algorithm that also works in distributed systems. The success by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) in resolving conflict through an algorithm, though in a different context—a file system rather than a distributed system—shows that success in resolving ACP conflict is achievable for this study.

The reviewed literature shows attempts in ACP resolving conflicts using algorithms with some success such as (Reeder, Bauer, Cranor, Reiter, & Vaniea, 2009), even in distributed systems (Fan, Liang, Luo, Bo, & Xia, 2011), but lack an administration model. The lack of using both an administration model and an algorithm to resolve conflict from the available literature is worth pursuing. Researching an administration model in the RBAC space that concludes in implementation solution should be a welcome addition.

Generalizability of Results

The proposed results of this study, an administration model with an algorithm in an inheritable component, was expected to be easy for practitioners to incorporate into their design and architecture. Should they be in the formative stages of implementing their distributed system, the data model of the distributed system could extend its security database objects to include the fields in the data model. Alternatively, they could add database objects as provided in the data model and integrate to the rest of their existing data model. The administrator of the system or the implementer would have to carefully review their existing administration model to ensure that the attributes required by WAAM are accounted for in the database. These would be both in the storage data objects as well as objects for retrieving ACP so the attribute values would be available to the algorithm. The algorithm was then encapsulated in a component that could be used in the system's authentication processes.

For those with a distributed system already implemented but not yet homogenized into a single security model, the results from this study would allow for the adoption of the administration into their existing data model. The changes needed would include creating new database storage object or extending existing ones to accommodate the data model as well as changes to a corresponding administrative interface to manage the ACP data. Implementing the algorithm for existing systems would also include altering components in the security module to use or implement the resultant component of this study.

For systems in formative stages, such as design, the model could be used as the security data model, or extended by adding attributes to existing database objects to meet WAA's needs. Likewise, the algorithm was placed in a compiled component that could be referenced in the system's authorization component or module.

Originality

This study proposed a synergistic approach of both an administration model and a conflict resolution algorithm (using both hierarchical roles and a weighted priority attribute). The studies that have algorithms in resolving policy conflicts such as OrRBAC (El Kalam, et al., 2003) do not use hierarchical roles while others (Fan, Liang, Luo, Bo, & Xia, 2011) only identify the policy conflicts. Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) discuss their specificity precedence improvements for the conflict resolution in a Windows file system simulation. Building on the Reeder, et al. (2009)

study which had conflict resolution was useful for this study to adopt for applications in a distributed system environment (DSE). The administration model in this study borrowed liberally from the study by Dekker, Crompton, and Etalle (2008) and extended it to include the calculation for weighted priority attribute (WPA) to be used in the algorithm for conflict resolution.

Barriers and Issues

Availability of a distributed system to use as a subject for this study may be challenging to find because such systems in practice are usually proprietary. The stakeholders of such proprietary systems may not consider kindly an outsider inspecting their system to see if it is performing satisfactorily. Similar to the articles by Juntapremjitt, Fugkeaw, and Manpanpanich (2008) and Reeder, Bauer, Cranor, Reiter, and Vaniea (2009), this study employed a simulated environment containing distributed applications with online user interfaces implementing the security algorithm for the study.

In order to prove that the solution would work there were iterations of designs resulting in an administration model and algorithm to identify and resolve ACP conflict. The complimentary designs for the administration model and algorithm were used in the simulation to show how they would work where implemented. The component implementing the algorithm was iteratively checked in the simulated environment to ensure performance was acceptable. Generally, systems with specific tasks visible to a user have to perform within acceptable time constraints. Applying the security model to existing systems would add to the tasks to be performed within the time constraints. So performance of the model for authorization is critical to acceptance by practitioners. To overcome the challenge of finding existing systems, building a simulation of a distributed system provided a baseline for the algorithm's ability to solve conflicts when they occur.

In the simulated environment, creating the ACP data required creating records via a generated script; however the script generation could not reliably create ACP conflict. The automated ACP script generation proved challenging to have ACP conflicts because creating each ACP entry script used known record identifiers so no conflict was created for the script. Manually created script entries were appended to the automated script in order to create conflicted ACP.

Without a baseline with which to compare the performance of the conflict resolution portion of the algorithm, the metrics recorded to determine performance were collected over multiple runs through the algorithm. The metrics, such as the number of runs, the time taken for the runs, the average number of runs, and the number of ACP evaluated; were then collected and aggregated to find statistical values to represent performance of the algorithm.

Limitations and Delimitations

Limitations

An administration model with an algorithm containing an inheritable component in the proposed results is expected to be easy for practitioners to incorporate into their system design and architecture. The natural limitation of WAA was how extensively the implementers would like to extend the algorithm.

Using a role hierarchy may not directly conform to an organizations' administrative concept (Kearn, Schaad, & Moffett, 2003). The large systems which apply their organization's administrative concepts may end up modifying an RBAC schema structure so that it fits their situation.

Evaluation as far as resolving conflict could be compared to an algorithm that resolves conflict such as the one discussed by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) though the conflicts there are not based on RBAC. Each system implementing this model would still need a sound design in its security model as well as sound security practices so that the system is not subject to abuse by negligent users (Karp, Haury, & Davis, 2009).

Delimitations

The natural limitation of how extensively the implementers would like to extend the algorithm would also depend on availability of skilled staff or other resources necessary to extend the algorithm. This study showed how the proposed model and algorithm would be implemented but did not go into the implementation of the inherited components of the extended algorithm.

Should the proposed role hierarchy in the model not directly conform to an organizations' administrative concept (Kearn, Schaad, & Moffett, 2003) there would have to be some accommodations to benefit from this study's solution. The model's implementation would be one where the objects used in storing and retrieving the ACP values for users' authorization would have to be altered to provide the attributes needed by the proposed algorithm to resolve conflict. For example, should the roles not have a hierarchy from which to retrieve the role attribute for the algorithm, an alternative attribute would be a ranking of the roles. This would still provide the attribution need for the algorithm to determine which role should get more weight when resolving any conflicts.

There was a challenge to finding the right fit for comparison with the proposed solution because it identified and resolved conflict in systems with RBAC based schema. The other RBAC models and algorithms only identified the conflicts (Fan, Liang, Luo, Bo, & Xia, 2011; Wu, Chen, Zhang, and Dai, 2009). The model by Fan, et al. (2011) proved a close match to compare the conflict identification portion of the solution. The conflict resolution could not directly be compared with what was provided by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) because of the differences in managing access for files in a windows system versus ACP in a RBAC system. Therefore the ACP conflict resolution portion was measured for accuracy and performance without comparison to results from an existing study.

Definition of Terms

ACP – Access Control Policy – this is a systems definition of which object can be accessed by a use and could also define what level of access. A user would generally have a set of these provided from the authorization process.

Authentication – this is the process by which a user identifies himself to the system proving a right to use a system (Karp, Haury, & Davis, 2009); such as by providing a user name (public key) and a password (private key).

Authorization – this is the process by which a system's security module determines which features or parts of the system a user can access as well as the level of access (Karp, et al., 2009) such as read-only or read-write.

Policy Conflict – occurs when policies satisfied by the authorization in a system have actions that are contradictory (Jajodia, Samarati, Sapino, & Subrahmanian, 2001; Wu, Chen, Zhang, & Dai 2009; Fan, Liang, Luo, Bo, & Xia, 2011).

RBAC – Role-based Access Control – this a methodology of controlling access to parts of a large system based on the roles that a user is assigned.

Request – this is the action-reaction when a user performs a gesture on a part of the use interface and the system responds accordingly.

User Session – this is an interactive period beginning when a user authenticates by signing into a system until they are signed out by a process or by signing out themselves.

Summary

RBAC has served as foundational to systems security and newer systems use it at various levels of improvement. Within RBAC-based research, ACP conflict occurs in systems whereby within a user's set of ACPs there are contradictory actions allowed on an object causing the system to misbehave. There have been attempts to automatically identify these conflicts but there is a lack of research showing how to automatically resolve the ACP conflict using an administration model and algorithm. This study proposed a solution to overcome this problem. The users and administrators of systems implementing the proposed solution would benefit by having consistent behavior and fewer demands directed to the administrators to identify and correct system anomalies resulting from ACP conflict.

Chapter 2

Review of the Literature

Introduction

This chapter reviewed published studies in the RBAC space showing the problems solved over time as well as the prevailing problems. This section reviewed some *Early RBAC* studies, some of the *Improvements over RBAC* that have happened over the years, some *Residual Problems in RBAC* that are still lingering in this research space, and *Adaptation of Results* from the previous studies that were useful in formulating the results.

Early RBAC

Some studies such as Bertino, Bettini, Ferrari, and Samarati (1996) considered access control and discussed temporal access and used discretionary access control (DAC) along with mandatory access control (MAC) as foundations. RBAC, depending on the implementation could be either but has elements of both MAC and DAC.

Mandatory Access Control

MAC is where the access is provided by the data being secured and the clearance level of the user (Atluri, Jajodia, & Bertino, 1997). It was generally used in multi-level secure military systems where preference is to security over confidentiality (Ferraiolo & Kuhn, 1992).

Discretionary Access Control

Downs, Rub, Kung, and Jordan (1985) describe DAC as a means of restricting access to objects based on the identity of the subjects or the groups to which they belong.

Users of a system with security based on DAC allowed a user to grant or deny privileges to system objects that they control without an administrator's intervention (Ferraiolo & Kuhn, 1992).

Role-Based Access Control

Organizations have a wide array of needs in terms of security policies of their systems that would be difficult to meet by either MAC or DAC alone (Ferraiolo & Kuhn, 1992). The introduction of RBAC brought about benefits such as ease of administration by matching ACP to a role rather than directly to a user (Sandhu, Coyne, Feinstein, & Youman, 1996). This administration of roles would have to be done by a trained administrator rather than by passing permissions from one user to another in DAC (Ferraiolo & Kuhn, 1992). Sandhu, Coyne, Feinstein, and Youman (1996) defined RBAC as a kind of access control whereby only authorized users are given access to specific data or resources in a system.

Within RBAC the role hierarchy is a constraint in that when a child role is granted access to an object, the parent roles also receive that permission. Also, a user assigned to a particular role automatically receives all the descendant roles in the hierarchy (Sandhu, Coyne, Feinstein, & Youman, 1996). They further identified lack of information regarding both configuration and constraints in RBAC systems as problems worth studying (Sandhu, Coyne, Feinstein, & Youman, 1996).

Improvements over RBAC

Administrative RBAC '97

The study by Sandhu, Bhamidipati, Coyne, Ganta, and Youman (1997) proposed ARBAC97 (Administrative RBAC '97) to administer the access control of an RBAC-based system. They posited that with enterprise size systems, the roles could number into the hundreds or thousands so they would need RBAC to manage the roles in the RBAC (Sandhu, et al., 1997).

Organizational RBAC

Bertino, Bettini, Ferrari, and Samarati (1996) considered dynamism in the model but would still need new algorithms to help with decentralization and periodic authorization. El Kalam, et al. (2003) shows the use of RBAC in an organizational context and calls their resulting model OrBAC. With organization structure usually having a hierarchy, OrBAC accommodates handling the organizational hierarchies in their ACPs. OrBAC has provided the basis for other research studies (Capolsini & Gabillon, 2009; El Kalam, Deswarte, Baïna, & Kaaniche, 2007) around access control rules based on temporal data such as time and location that use OrBAC and are referenced in other research with some improvement. El Kalam, et al. (2003) which proposed OrBAC has improvement over RBAC by adding hierarchy and organizational context to roles. The El Kalam, et al. (2003) study shows OrBAC as an improvement based on how the model is setup though still lacking in administration of policies and enforcement when the policies are violated.

Integrated Policy

Shafiq, Joshi, Bertino, and Ghafoor (2005) proposes an integrated policy for RBAC systems in what they describe as multi-domain environment, similar to this study's distributed system, because different applications collaborate as they perform their tasks. Shafiq, et al. (2005) describe their approach using an integer programming to homogenize the ACP across the different applications. The administrators would have to make trade-offs to find a balance between integration into homogeneous polices or autonomy of the distributed systems (Shafiq, et al., 2005).

Generalized Temporal RBAC and Geographical RBAC

The RBAC study by Joshi, Bertino, and Ghafoor (2005) shows that roles are important though still inadequate so they proposed Generalized Temporal Role-Based Access Control (GTRBAC) that brought some improvements over RBAC. Joshi, et al. (2005) augmented RBAC with time as a dynamic component in the GTRBAC model alongside the user context to determine access control in a hospital system. The GTRBAC model would also improve over RBAC in expressiveness and usability (Joshi, et al., 2005). The GTRBAC would be further improved with GEORBAC (GEOgraphical Role Based Access Control) which takes into account the location of the subject and object when considering access control (Damiani, Bertino, Catania, & Perlasca, 2007). The ACP incorporates the location of the user as the subject, as well as the bound location of the object (Damiani, Bertino, Catania, & Perlasca, 2007). Cruz, Gjomemo, Lin, and Orsini (2008) describe a similar access control system that uses global positioning system to determine location and derive permissions based on other role attributes the user may have.

Concrete and Abstract RBAC

CABAC, presented by Bouzida, Logrippo, and Mankovski (2011), uses predicate logic that results in granting and revoking access to users based on the changes in static data and a dynamic data context. CABAC attempts to combine rules into contexts though it has trouble when there are conflicts in the policies (Bouzida, Logrippo, & Mankovski, 2011).

PolyOrBAC

Damiani and Sylvestri (2008) mention challenges in accommodating systems with distributed architecture as they developed the GEORBAC model. However, El Kalam, Deswarte, Baïna, and Kaaniche (2007) accommodate systems with distributed architecture by employing web services to manage access control for distributed systems in a collaborative context using *PolyOrBAC*. An improvement to PolyOrBAC in expressiveness is the *distributed-RBAC* that provides single-sign-on to distributed systems using XACML (eXtensible Access Control Markup Language) which is based on XAML (Juntapremjitt, Fugkeaw, & Manpanpanich, 2008). Introducing distributed-RBAC brings dynamism in ACP by emphasizing a decentralized implementation (Juntapremjitt, Fugkeaw, & Manpanpanich, 2008). Similarly, Pérez, Lòpez, Skarmeta, and Pasic (2010) bring about a concept whereby the system administrators delegate some of the administrative tasks to agents at the application level in a distributed system. Fan, Liang, Luo, Bo, and Xia (2011) propose an algorithm to find conflicts for the administrators to solve—providing a benefit for the administrators to resolve the conflicts whenever identified. The studies that describe the newer models provide improvement over the older RBAC showing a maturing process for the access control discipline though there is still some space worth investigating.

Administrative RBAC

Some of the improvements in the RBAC administration by Oh, Sandhu, and Zhang (2006) propose ARBAC02 (Administrative RBAC '02), which improves over ARBAC97 by using bottom-up inheritance. ARBAC02 introduces *user pool* and *permission pool*; user pools are special roles that contain permissions at the central system and permission pools that are roles at the distributed system (Oh, Sandhu, & Zhang, 2006). Users in systems using ARBAC02 would have to be assigned to both a user pool and a permission pool. ARBAC02 supports hierarchical roles so permissions are assigned to the lower policies and inherited up the hierarchy (Oh, Sandhu, & Zhang, 2006). Further improvements in RBAC administration are discussed by Li and Mao (2007) in UARBAC which uses parameters and units in defining its permissions so they can be delegated to improve scalability. Dekker, Crompton, and Etalle (2008) discuss the administration of RBAC in distributed systems. They discuss administration in heterogeneous distributed systems and use a method that begins with applying a security policy in each of the distributed systems in a distributed systems environment (Dekker, Crompton, & Etalle, 2008).

NBAC and ZBAC

In the study by Karp, Haury, and Davis (2009), the older models were presented as NBAC (autheNtication-Based Access Control) which has some of the RBAC solutions but also brings up its own issues. One of the NBAC issues they (Karp, Haury, & Davis, 2009) identify is role explosion. Role explosion is a situation whereby a system ends up with overly granular roles or roles that are very similar. They present a solution to the issues they present in ZBAC (authoriZation-Based Access Control) which restricts users to specific domains (Karp, Haury, & Davis, 2009).

RBAC96

Jiong and Chen-hua (2012) base their study on RBAC96 and propose a consistent constraint schema to help administrators in RBAC systems and categorize the constraint conflicts as either external or internal. Their study defines external conflicts as those occurring “when the configuration of RBAC does not satisfy the constraints defined in the system” (Jion and Chen-hua, 2012, p2) and internal conflicts occurring when “two or more constraints are deemed incompatible with each other” (Jion and Chen-hua, 2012, p2). The approach that Jiong and Chen-hua (2012) offered identified the constraint for resolution but would need to be automated into an algorithm. The solution from this study considered that assigning conflicting permissions to a role would be prevented in a user interface that limits the administrator to the selection of a single permission when creating an ACP so internal conflicts according to Jiong and Chen-hua (2012) would not occur. The external conflict is what would be considered in this study because a user could have direct permissions to an object and be assigned roles that have conflicting permissions to the same object. This study goes beyond what Jiong and Chen-hua (2012) proposed because the administration model for this study overcomes the internal conflict because one role may not have multiple permissions for the same object.

Spatially Aware RBAC

Damiani, Bertino, Catania, and Perlasca (2007) proposed improvements to RBAC by adding roles and location constraints in determining the access to be granted to the user by a system. They discussed location-based services and mobile applications creating a demand for spatially aware systems. *They extend role-based access control (RBAC) by*. This extension forms GEO-RBAC (Geographic RBAC) which added a geographically derived spatial role to the user. Rather than just use the user’s location, they split position into logical and real whereby the real position is based on geography and the logical position is computed from the real position to provide some extension to the spatial location. They also separated the duties of the user by role and also employer hierarchies to simplify role definition. This article does not cover the administrative operations or moving spatiotemporal extents (Damiani, Bertino, Catania, & Perlasca, 2007).

As use of mobile access to systems is expected to grow, the use of derived contextual attributes such as geographic location of a user, the local time of the system or of the user could be part of future studies related to resolving ACP conflict (Abdunabi, Ray, & France, 2013). A future study could find the possibility of involving attributes available from a mobile user such as location to determine how to resolve any ACP conflicts.

UARBAC and ACPCDM

Li and Mao (2007) introduced UARBAC essentially as a way to use RBAC to administer RBAC systems. RBAC was presented as policy neutral, meaning it could be configured to enforce different kinds of policies in simultaneously. UARBAC, as discussed by Li and Mao (2007), described a tuple as follows: $\langle C, OBJS, AM \rangle$ where: C is a finite set of object classes the system supports, for example, {user, role}; OBJS is a mapping function for C that returns a set of object names such that OBJS(user) returns a set of all possible user names; and AM which is a function that maps each class to a predefined set of access modes, for example AM(user) returns {empower, admin} (Li & Mao, 2007).

Fan, Liang, Luo, Bo, and Xia (2011) showed that policy conflict was an existing problem in current systems. They proposed an algorithm to review an entire authorization policy and pointing out the discrepancy which they called ACPCDM (ACP Conflict Detection Model). They used XACML for their policy expression (Fan, et al., 2011). ACPCDM identified two types of policy conflict: separation of duty and cyclic inheritance. Both of these conflicts were identified using ACPCDM which they proposed. They identify conflict using ACPCDM model which ran against ACP listed in two files. Each file represented ACP from different domains that need to be merged. Their conflict detection includes: removing duplicates, reasoning which determines the conflicts, and analyzing the results (Fan, et al., 2011). For this study we considered different systems in a DSE.

Residual Problems

Policy Conflicts

Moffett and Sloman (1994) raise issues regarding management of policies and point out a need to analyze resulting conflicts in the policies but only present a theoretical model leaning toward automated management. A later study (Schaad & Moffett, 2002) posited that resolving the conflict of polices was in delegating authority to the decentralized applications within the distributed system. Shu, Yang, and Arenas (2009) declare that the problem of policy conflicts exists. Their article proposes a method for a conflict detection solution which they implement in a prototype (Shu, Yang, & Arenas, 2009). Policy conflicts were studied more recently by Wu, Chen, Zhang, and Dai (2009) who contract security policies into matrix groups and then use their algorithm against the matrices in order to detect ACP conflicts. The checks they used are computationally intense and could be improved in reducing the calculations used. Fan, Liang, Luo, Bo, and Xia (2011) propose an algorithm to review the policies and identify the policies that have conflicts; however they defer to the administrator to resolve the conflict.

The RBAC based studies, while providing improvement, either avoid addressing policy conflict resolution or work within the prescriptions of a prior model. Conflicts, according to Wu, Chen, Zhang, and Dai (2009), occur in a situation where overlapping events conditions or actions and any two or more of the actions end up being contradictory. Fan, Liang, Luo, Bo, and Xia (2011) identifies the conflicts as disaccords in the roles between permissions and the representative schema representation of a system's ACPs. Their proposed ACP conflict detection model (ACPCDM) contributes identification of the policy conflicts but do not address policy conflict resolution.

The study by Schaad and Moffett (2002) posited that solving the conflict of polices was in separation of duty controls to the decentralized applications within the distributed system and integration of administrative mechanisms. The ability to centralize the administrative mechanisms would make it easier to administer the system from a single place though decentralizing some administrative function to the systems where they are relevant would be useful to consider for this study. Abdunabi and Ray (2010) suggested that developers of the systems were more likely to use technical concepts that were easier to understand and administrators more likely to use automated approaches. This could involve running the ACP conflict identification algorithm off hours or when system usage is low (Abdunabi & Ray, 2010)

Global Policy

The homogenous unification into a global policy as shown by Shafiq, Joshi, Bertino, and Ghafoor (2005) is largely a process implementation to pool together the policies of the systems in a multi-domain environment. A multi-domain environment is a collection of cooperating single domain systems (Shafiq, et al., 2005). The study also states that the underlying systems must be RBAC based before it could be useful.

The multi-domain environment (Shafiq, Joshi, Bertino, & Ghafoor, 2005) was very similar to this study's distributed system environment and some elements were incorporated such as integrating the ACP names across the applications in the system. Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) discuss specificity precedence for the conflict resolution in their Windows file system simulation that could be useful in this study if adopted for applications in a distributed system.

ACP Conflict in Collaborative Systems

The studies reviewed show some improvement but still have some problems worth investigating: Bertino, Catania, Ferrari, and Perlasca (2003) state that ACP conflict is a problem but do not attempt to solve it in their model; instead they cede conflict resolution to the specification of the system. The PolyOrBAC model needs to improve in detecting and resolving conflicts in security policies in collaborative systems (El Kalam, et al, 2007). Pérez, Lòpez, Skarmeta, and Pasic (2010) introduced delegated administration using XACML but without support for distributed systems (Pérez, Lòpez, Skarmeta, & Pasic, 2010). D-RBAC, presented by Juntapremjitt, Fugkeaw, and Manpanpanich (2008) still does not address conflict resolution over PolyOrBAC. CABAC attempts to combine rules into contexts though it has trouble when there are conflicts in the policies (Bouzida, Logrippo, & Mankovski, 2011).

According to Shafiq, Joshi, Bertino, and Ghafoor (2005), the conflicts in ACP are classified into four types:

1. Modality conflicts – where positive and negative policies exist in an authorization;

2. Multiple management – occurs when administrators specify conflicting authorizations for the same roles.
3. Cyclic inheritance – occurs when a subject lower in the hierarchy ends up with permissions of a subject higher in the hierarchy.
4. Separation of duty – prevents access of an object when there would be conflict of interest.

Each of these has different causes and may be addressed separately and independently. The conflicts that occur under RBAC are cyclic inheritance and separation of duty (Shafiq, Joshi, Bertino, & Ghafoor, 2005) and are the conflicts covered by this study.

Damiani and Sylvestri (2008) mentioned challenges in accommodating systems with distributed architecture as they extended the GEORBAC model to be motion-aware. The article still struggled with the separation of duty conflicts. This was an attempt to mitigate the conflicts by adding constraints in the creation of ACP but admit they were not able to resolve the conflict (Damiani & Sylvestri, 2008).

The study by Pérez, Lòpez, Skarmeta, and Pasic (2010) proposed a concept of *administrative delegation* whereby the system administrators of a distributed system delegate some of the administrative tasks to users responsible at the application level who only have to deal with ACPs at the application level. The concept of administrative delegation could be useful for this study as administrative tasks are performed by administrators who are functionally closer to the tasks or objects being secured.

Access Conflict Resolution

The studies based on RBAC that recognize conflicted ACP as a problem only went as far as identifying the conflict. The study by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) showed that conflict in access policies could be resolved using an algorithm. In their study, they simulated the file access policies on Windows and chose to improve them because of conflicts that came out of divergent polices. They described ACP as consisting of a set of rules under which users are allowed to access system resources (Reeder, et al., 2009).

ACP conflict occurs when the user cannot get access to a resource when an action is allowed in one policy but denied in another (Reeder, et al., 2009). Without conflict resolution a system would behave unexpectedly. Their study used a Windows environment which, to mitigate the potentially erratic behavior during ACP conflict, gave precedence to the deny policies should there be a conflict.

Adaptation of Results

The study by Schaad and Moffett (2002) posited that solving the conflict of polices would be in separation of duty controls to the decentralized applications within a distributed system and integration of administrative mechanisms. The ability to centralize the administrative mechanisms would make it easier to administer the system from a single place. Abdunabi and Ray (2010) suggested that developers of the systems more likely used technical concepts that were easier to understand and administrators more likely to use automated approaches. This involved running the ACP conflict identification algorithm off hours or when system usage is low (Abdunabi & Ray, 2010)

The multi-domain environment (Shafiq, Joshi, Bertino, & Ghafoor, 2005) was very similar to this study's distributed system though some elements were incorporated such as integrating the ACP names across the applications in the system. Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) discussed specificity precedence for the conflict resolution in their Windows file system simulation that could be useful in this study if adopted for applications in a distributed system.

The study by Pérez, Lòpez, Skarmeta, and Pasic (2010) proposes a concept of *administrative delegation* whereby the system administrators of a distributed system delegate some of the administrative tasks to users responsible at the application level who only have to deal with ACPs at the application level. The concept of administrative delegation could be useful for this study as administrative tasks are performed by administrators who are functionally closer to the tasks or objects being secured.

Summary

This chapter reviewed available literature from various periods in the development of access control in systems. The improvements in access control from various studies were reviewed from MAC (Ferraiolo & Kuhn, 1992) and DAC (Downs, Rub, Kung, & Jordan, 1985), which formed the foundation of RBAC, and continuing the review with UARBAC (Li & Mao, 2007) and ACPCDM (Fan, Liang, Luo, Bo, & Xia, 2011) which were foundational for this study. The residual problems that were identified from the literature like ACP conflict in collaborative systems of which this study proposed a solution and how the results could be adapted.

Chapter 3

Methodology

Overview

This chapter discussed the algorithm proposed by this study to answer the identified research questions. This quantitative study used metrics measuring performance of the weighted attribute algorithm and weighted attribute administration model for ACP operations during authorization into a system. The metrics collected from the simulation that implemented both WAA and WAAM included the number of ACP evaluated by WAA; the number of runs through the algorithm where each run represented a user being authorized into a RBAC-based system; the delay added to authorization by the algorithm; and the number of ACP conflicts resolved vis a vis the known conflicts for each user. The delay added to the authorization was considered for performance which the comparison of the number of known ACP conflicts with those identified was WAA's identification accuracy. This chapter then discussed the procedures employed to evaluate WAA's accuracy and performance, the performance metrics used, and how these metrics were obtained from an implementation of the algorithm using a simulation.

Research Method

This study proposed the *weighted attribute algorithm* (WAA) and employed the quantitative method to help answer the research questions identified in Chapter 1. The overall method was to analyze ACP conflicts in RBAC-based systems then design, develop, and implement a simulation of WAA as a way to collect evidence supporting the ability to identify and resolve ACP conflict accurately during user authorization.

WAA needed some attributes be added to the ACP described by Li and Mao (2007) in order to resolve ACP conflicts. Implementing WAA required introducing attributes available to the ACP resulting in the weighted attribute administration model (WAAM). The design for WAAM, discussed below, encapsulates the attributes added to UARBAC (Li & Mao, 2007). To evaluate WAA and WAAM, metrics for accuracy and performance were collected to see the viability.

The procedures employed during this investigation follow the patterns used by Fan, et al. (2011), Reeder, et al. (2009), and by Li and Mao (2007). This study used the following procedures in designing WAA to identify and resolve ACP conflict along with its supporting WAAM:

- to identify the metrics needed to evaluate the performance of the algorithm
- to design the simulation environment containing WAA and ACPCDM (Fan, et al., 2011) to collect evidence for performance evaluation in accuracy and efficiency
- to create the seed dataset, and
- to report metrics for performance evaluation in accuracy and efficiency.

Designing the Weighted Attribute Algorithm

The design for WAA had three primary objectives in providing RBAC-based hierarchical systems a solution for ACP conflict resolution: first identify the conflicts to be resolved, second was to resolve the ACP conflicts, and third was to provide simple implementation for administrators who choose WAA for their conflict resolution. These objectives, as accommodated in WAA, are in the flowchart in Figure 1 below. The basis of the conflict resolution is the *weighed priority attribute* (WPA) which was derived from the ACP attributes: user position and role hierarchy. Figure 1 below is broken down into numbered steps such as 1.0 and 2.0 with steps at a lower level of granularity numbered as 4.4 or 4.6.1.

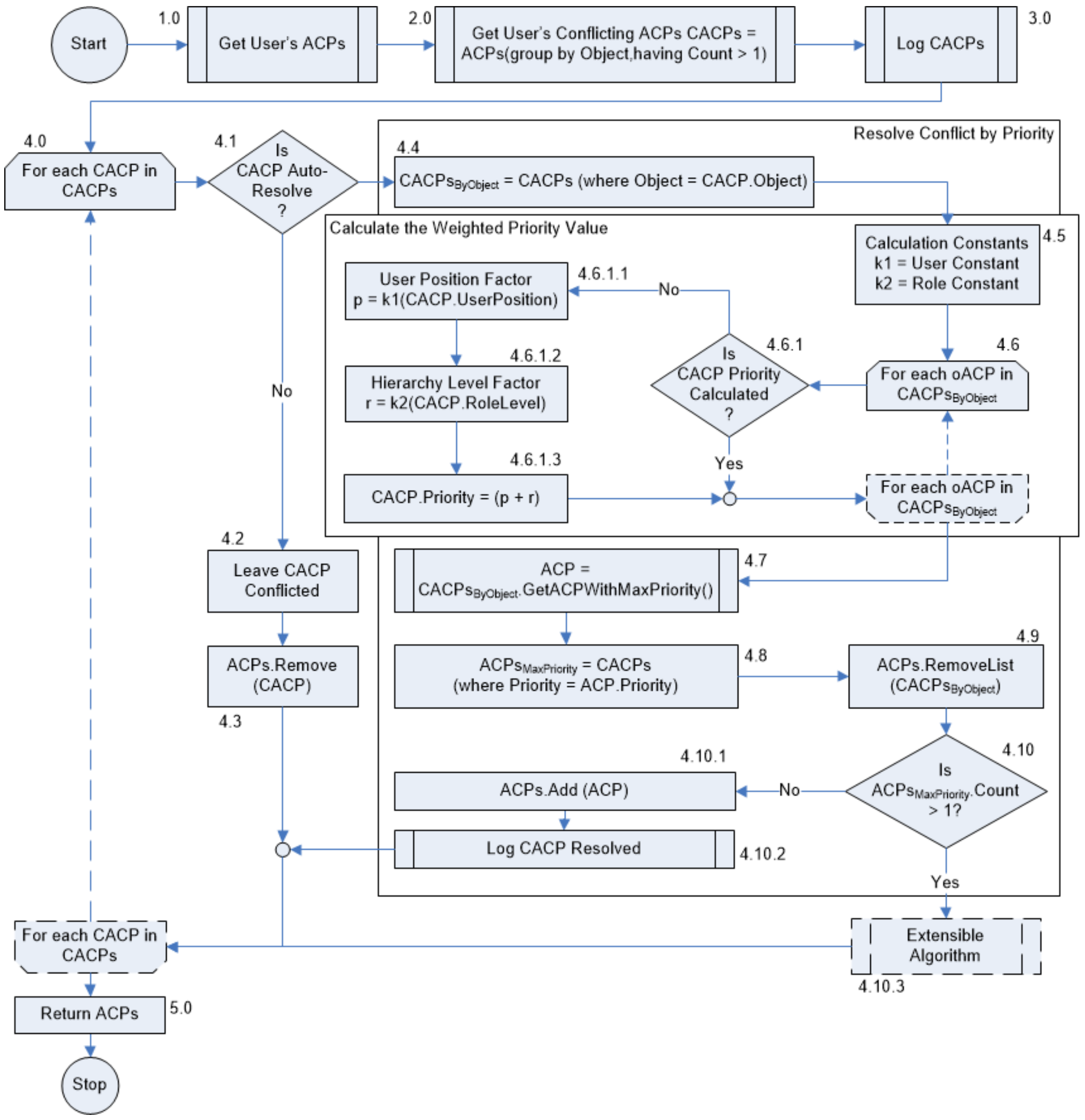
Start
1.0
Get User's ACPs
2.0
Get User's Conflicting ACPs CACPs = ACPs(group by Object,having Count > 1)
Log CACPs
3.0
4.0
For each CACP in CACPs
4.1
Is CACP Auto-Resolve ?
Resolve Conflict by Priority
4.4
CACPs_ByObject = CACPs (where Object = CACP.Object)
Calculate the Weighted Priority Value
Calculation Constants k1 = User Constant k2 = Role Constant
4.5
User Position Factor p = k1(CACP.UserPosition)
4.6.1.1
No
4.6.1.2
Hierarchy Level Factor r = k2(CACP.RoleLevel)
4.6.1
Is CACP Priority Calculated ?
4.6
For each oACP in CACPs_ByObject
No
4.6.1.3
CACP.Priority = (p + r)
Yes
For each oACP in CACPs_ByObject
4.2
Leave CACP Conflicted
ACP = CACPs_ByObject.GetACPWithMaxPriority()
4.7
ACPs.Remove (CACP)
4.3
ACPs_MaxPriority = CACPs (where Priority = ACP.Priority)
4.8
4.9
ACPs.RemoveList (CACPs_ByObject)
4.10.1
4.10
ACPs.Add (ACP)
No
Is ACPs_MaxPriority.Count > 1?
Log CACP Resolved
4.10.2
Yes
For each CACP in CACPs
Extensible Algorithm
4.10.3
Return ACPs
5.0
Stop

Figure 1. Authorization Flowchart using WAA for Conflict Resolution

Designing WAA required that each ACP contain attributes to derive WPA. These attributes: user position, role hierarchy level, and is-auto-resolve flag are used in the comparative portions of the algorithm. The is-auto-resolve flag is an indicator to tell WAA whether to attempt to resolve the ACP conflict. This allows for administrators to deem ACP conflicts on specific objects not be resolved by the algorithm. This is useful if ACPs of an object containing sensitive data, like access to monetary transactions are in conflict, they would be left in conflict and an administrator' intervention needed to correct the ACPs in conflict.

UARBAC, as discussed by Li and Mao (2007), described a tuple as follows: $\langle C, OBJS, AM \rangle$ where: C is a finite set of object classes the system supports, for example, {user, role}; OBJS is a mapping function for C that returns a set of object names such that OBJS(user) returns a set of all possible user names; and AM which is a function that maps each class to a predefined set of access modes, for example AM(user) returns {empower, admin} (Li & Mao, 2007). For this study WAA required that the expected tuple of ACP be extended to: $\langle C, OBJS, AM, WPAF \rangle$ where WPAF is the function to determine the weighted priority attribute (WPA). For instance, WPAF(role, user) would return a numeric value for WPA. The WPA is derived as shown in step 4.6.1.3 of Figure 1 from the user position and role hierarchy level attributes discussed further in the design for WAAM below. The WPAF(role, user) function used the following formula to calculate WPA:

$$Priority = k1(UserPosition) + k2\ (RoleLevel)$$

For the calculation, *k1* and *k2* are positive non-zero configurable constants where *k1* ≠ *k2*. Based on their knowledge of a system's usage, the administrator would determine that *k1* which modifies the user weight is of higher importance than *k2*.

The WAA is designed to easily integrate into an existing system where a set of user ACP is provided for user authorization as shown in Step 1.0 of Figure 1. Getting the set of user ACP would be from the local database if the user is authenticated locally or from a database in another system in the environment. This achieved enough flexibility to work for both regardless of where the user's ACP are obtained from so long as they have the attributes required to calculate WPA for conflict resolution.

Designing Allowance for Extensible Algorithm

WAA was designed to be extensible so that implementers could augment the conflict resolution for those ACPs that are still in conflict after running WAA. This option which is the *extensible algorithm* in Step 4.10.3 of Figure 1 provided flexibility to the implementers who could, by adding or deriving other attributes from the ACP. This option allowed for implementing systems to add further logic to resolve any ACP conflicts that WAA cannot resolve. For example, an implementer could have an attribute on the ACP based on the time the role was created, and could use this attribute in their implementation of extending WAA.

The design to integrate with an extensible algorithm called for the implementation of the extensible algorithm to subscribe to a WAA interface and be added to the configuration as an available extension to WAA. It would also be limited to being compiled similarly to the implementation of WAA, so that if Java was used the extensible algorithm could be extended seamlessly if it were implemented in Java as well. It would still be possible to use different technologies like Microsoft .Net and Java together but would add a logical layer of interoperability between the different technologies. This design recommended that any extensible algorithm be implemented in like technologies, such that a Microsoft .Net implementation of WAA would be extended by an implementation in Microsoft .Net. The simulation used for this study was compiled using Microsoft .Net.

Designing the Weighted Attribute Administration Model

This design for Weighted Attribute Administration Model (WAAM) was made to complement WAA by providing attribution to the set of ACPs used to authorize a user into a system. The WAAM extended the UARBAC model discussed by Li and Mao (2007) as well as hierarchical roles (Damiani, Bertino, Catania, & Perlasca, 2007). This study combined these concepts into an administration model including adding attributes from which the weighted priority attribute (WPA) would be derived. Two options were considered to add attributes for WPA: add new objects with the desired attributes, or extend existing database objects.

The option to add new objects while plausible would be more challenging to maintain making it less desirable for administrators. Adding new objects to a database to contain the attributes for WPA would allow existing related and dependent client objects to work as before though access to the new objects would require new stored procedures or access queries. Extending existing objects with the ACP attributes with default values would allow for backwards compatibility though existing access stored procedure or queries would have to be altered to accommodate the new attributes. Adding new objects would also require three new objects that have a 1:1 relationship with existing objects; for instance, the entity containing the *RoleHierearchy* would have a 1:1 relationship with the Role entity and may have to be updated in concert. For this study the attributes that WAAM provided were added to existing database objects so that queries for user ACP would only be extended by adding the additional fields rather than extending authorization by joining additional objects. The entity relation diagram representing the WAAM is presented in Figure 2 below.

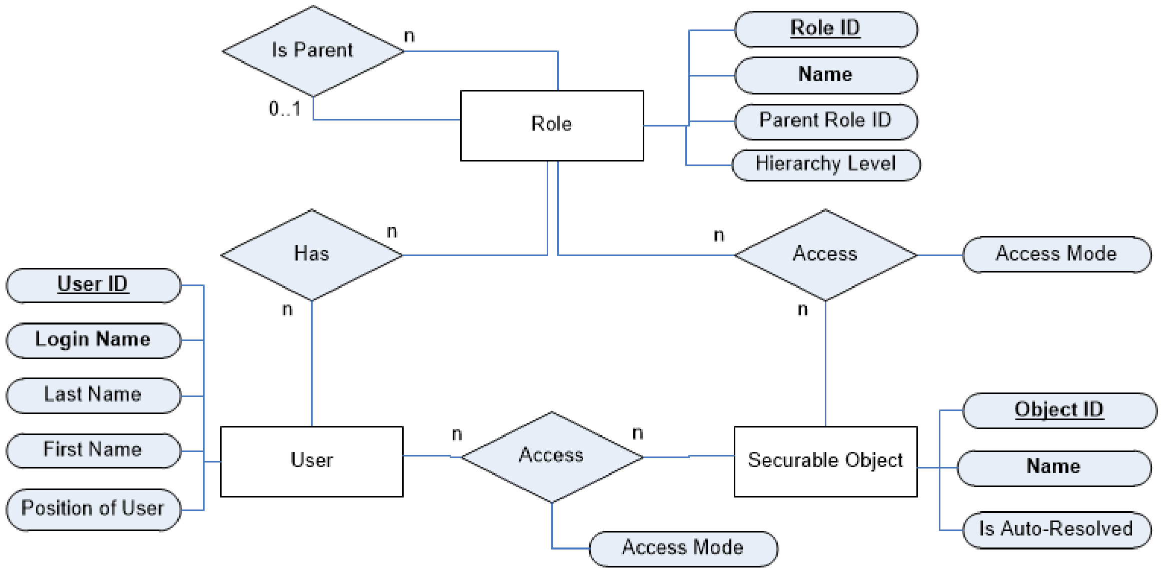

Figure 2. ERD of the Weighted Attribute Administration Model

The design for WAAM called for three fields to be added to an existing RBAC-based database implementation. These fields would allow for easy implementation for an existing system because existing stored procedures and queries just need to access a new field rather than build new queries. These are the three attributes that were used by WAA:

1. The *is-auto-resolved* attribute which tells the algorithm whether to consider the ACP for resolution when there is a conflict.
2. The *hierarchy level* used in calculating the role hierarchy factor in determining WPA.
3. The *position of user* used in calculating the user position factor in determining WPA.

The is-auto-resolved flag only needs to be updated along with a record in the *SecurableObject* object and all ACP associated with the securable object would share that attribute. WAA used the is-auto-resolved flag to determine whether to calculate WPA.

Consideration for the Distributed System Environment

A distributed system environment (DSE) is one consisting of multiple systems that a user can authenticate once and be authorized to perform tasks on any of the systems based on their ACP. The layout of the distributed system considered is shown in Figure 3 which shows a DSE with WAA implemented in the satellite systems. The primary and secondary systems are shown to have extended WAA while Node-1 is without an extension. The Node-n represents any other system that is part of the DSE that implements WAA and has an option to extend WAA at its administrators' discretion and implementation of their choosing.

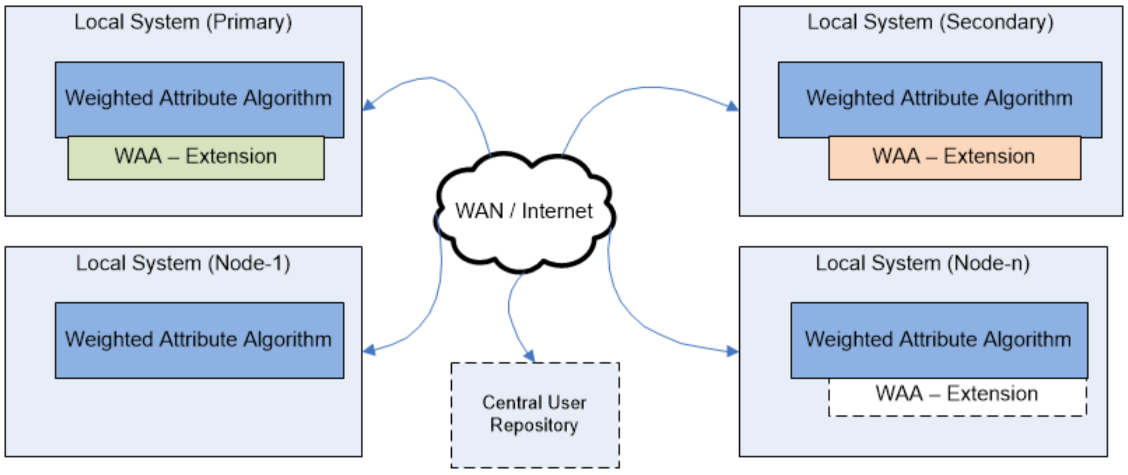

Figure 3. Location of WAA in a Distributed System

A local system, for this study, is one where the user record of the authenticating user exists. For example, a user attempting to use the system in Node-1 would make Node-1 the local system for that user's session. When authorizing at Node-1 in Figure 3 where the user already exists, the ACPs would be retrieved from the local system database in Node-1. The central user repository would contain all user records for the systems in the DSE, if so configured. When no central repository is configured in the DSE, the primary node would function as the central user repository for authentication purposes. The secondary node could be any node in the system but provides redundancy for the primary node. The extension is available only where implemented by the administrators and could differ from one local system to the next depending on additional attributes and logic as they see fit.

For a DSE to implement WAA fully, each system would have to extend their ACP to include the attributes to support WAA by also implementing WAAM. If a DSE only has some of its constituent systems with WAA, the benefits of ACP conflict resolution would only apply when both the authenticating system and authorizing system have WAA and WAAM. The systems where the ACPs are obtained that do not implement WAAM would be unable to provide the needed attributes to calculate WPA.

The logic to identify the system from which to obtain a user’s ACPs for authorization in a distributed environment were encapsulated in Step 1.0 of the flowchart in Figure 1. The scenarios considered were:

1. Authorizing a user in the local system where the user’s ACPs exist
2. Authorizing where a user is not in the local system but:
    a. is in the primary node of the DSE
    b. is in a central user repository of the DSE

These scenarios were covered in Figure 4 below:

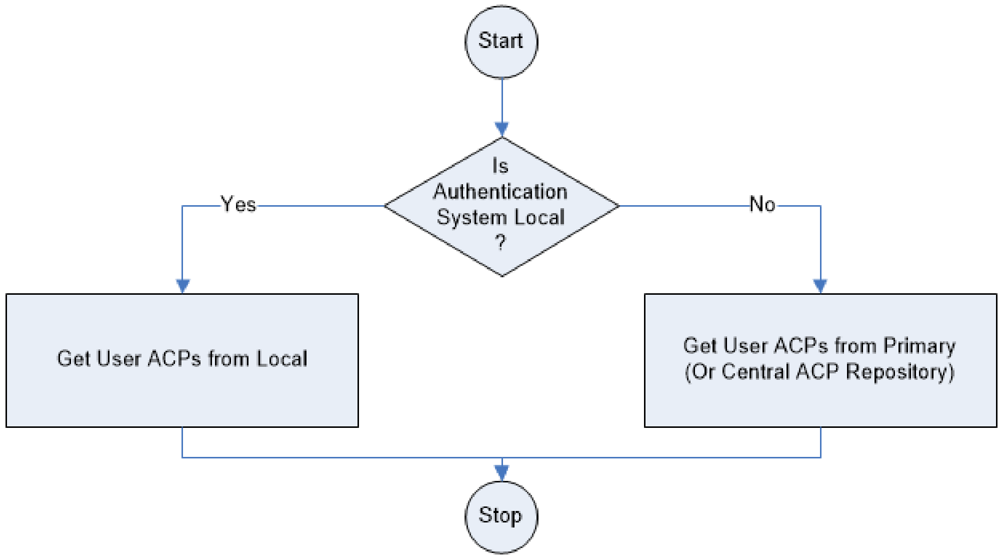
Start
Is Authentication System Local ?
Yes
No
Get User ACPs from Local
Get User ACPs from Primary (Or Central ACP Repository)
Stop

Figure 4. Determining the Source of User ACP

The ACP retrieval, as shown in Figure 4, would obtain all the ACP from the system where the authentication was attempted. When considering the second scenario where authentication against a local system fails, authentication would then be attempted against the primary node (or central user repository, depending on how the DSE is constituted). Upon successful authentication at the primary node—or central user repository, the ACPs would be retrieved from there and authorization performed at the local system.

Consider the same steps in the flow for retrieving the user's ACP shown in Figure 4 (Step 1.0 of Figure 1); they are all outside of the WAA. This study determined that where a user's set of ACP lies is of little significance to WAA in calculating WPA because the presence of the attributes determined whether WAA could be used.

Designing the Simulation Environment

The simulation environment was designed to accommodate both WAA from this study and ACPCDM proposed by Fan et al. (2011). ACPCDM was included to establish a baseline for identifying ACP conflicts of a user during authorization. Figure 5 below (corresponding with Figure 1 beginning in Step 2.0) shows how the logical components were arranged to fit in the simulation environment.

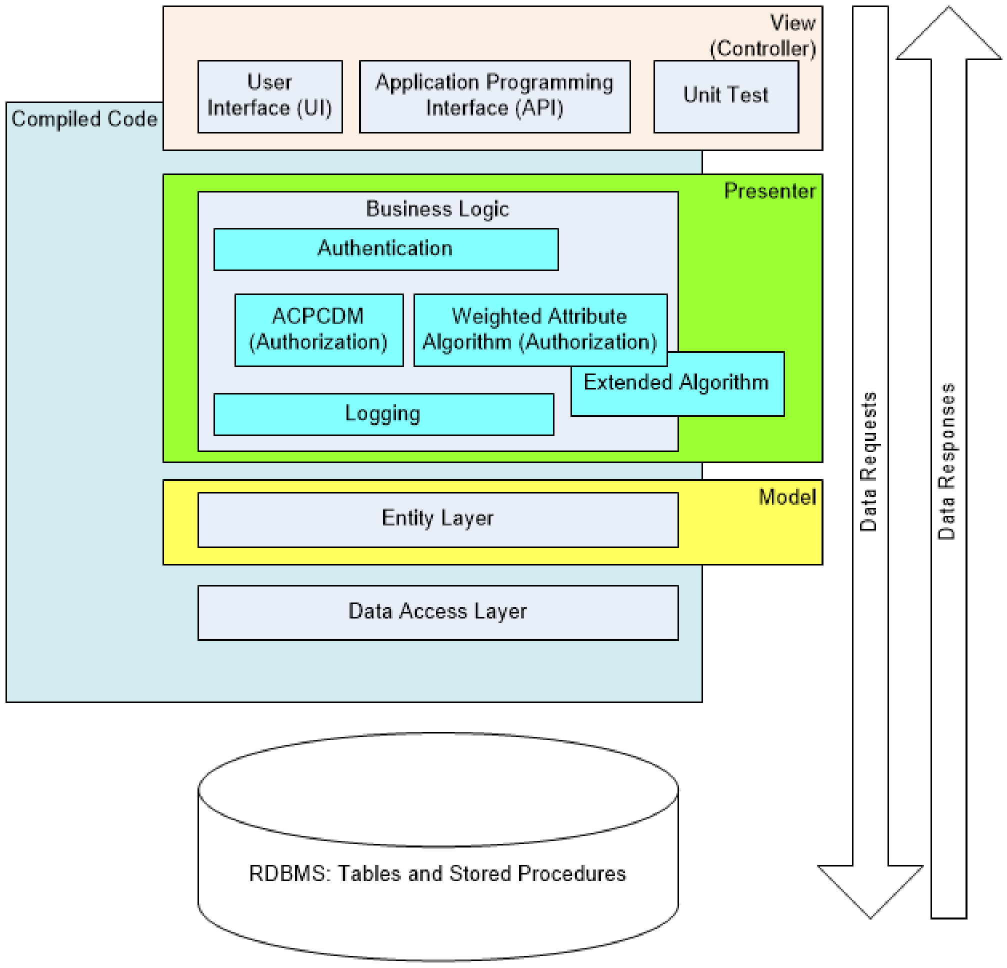

Figure 5. Logical Design for the Simulation Environment

The implementation of the DSE is limited to a single node on the DSE. Having both algorithms in the simulation facilitated use of common data in the user, object, and role entities which constituted ACP. Each run through the algorithm had a user to obtain a set of ACP from the common database where the metrics were also collected. These algorithms were implemented into a combined logic pattern as shown in Figure 6.

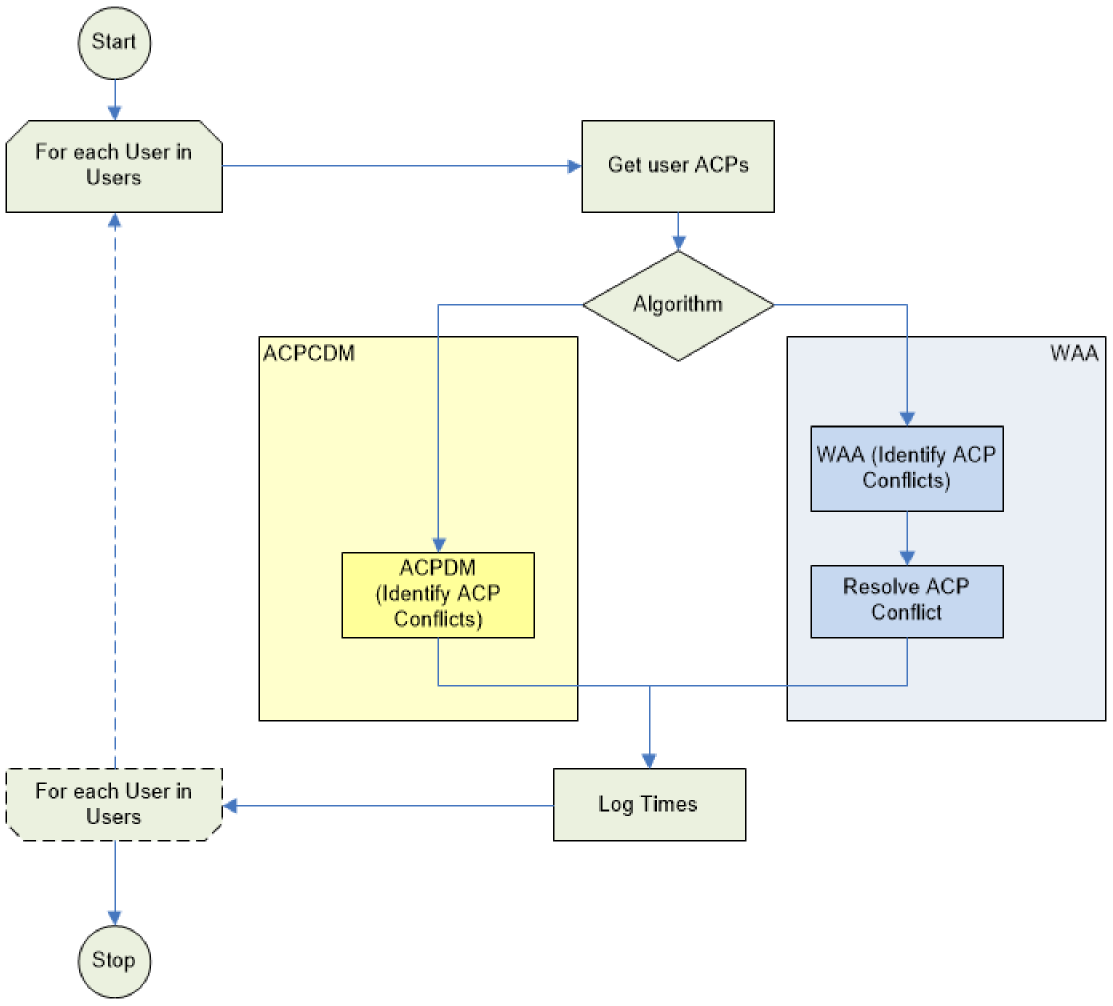

Figure 6. Design of the Simulation

This simulation environment contained single database implementing WAAM for all the ACP data for runs through the algorithms. The script used to create the tables is in Appendix A:. Appendices B and C contain the scripts used to implement the view and stored procedures respectively. Appendix D contains the list of tools and related components as shown in Figure 6 that facilitated executing the simulation application.

Preparing the Seed Data used for Runs through WAA and ACPCDM

In order to create the data to support the ACPs for the simulation, the code in Appendix H was used to create all the data in the objects needed to support creating users, roles, objects, and their relationships for the simulation. This was run as a console application to prepare the simulation environment for the algorithm runs to collect the metrics needed for the study. The program created a script containing *SQL Insert* statements for all the data used for the runs. The approach populated objects in the compiled code corresponding to database objects and then serialized the objects into SQL Insert statements. The statements were then run into the database to create the user, roles, objects, and ACP data to support the runs through either algorithm.

The same sets of ACP were used for identifying conflicts using ACPCDM, and for both identifying and resolving conflicts using WAA. For each run, a different system user was used providing a different set of ACPs. When changing the algorithm the same set of users was used guaranteeing that metrics collected are from the same sets of ACP for the runs. The decision point during the runs to determine which algorithm to use was placed in the configuration file where the value was edited to the desired algorithm. The configuration was placed as follows in order to run the ACPCDM algorithm:

```
<appSettings>
  <add key="Algorithm" value="Acpcdm"/>
</appSettings>
```

Figure 7. The Configuration Setting when running ACPCDM

The algorithm was set as follows for the weighted attribute algorithm:

```
<appSettings>
  <add key="Algorithm" value ="WeightedAttribute"/>
</appSettings>
```

Figure 8. The Configuration Setting when running the Weighted Attribute Algorithm

This configuration option was simple enough to toggle the algorithm to use for a set of runs through an algorithm in the simulation.

Collecting Metrics from Runs through WAA and ACPCDM

There were 100 users selected from the database to be used for runs through each algorithm. From the runs through the algorithms performance and accuracy metrics were collected to facilitate comparison and analysis of the algorithms. Each user record had its own set of ACP data based on the roles assigned to the user and the objects accessible to those roles and their descendants. There were also ACP records that were obtained from a user record having direct access to a secured object. So obtaining ACP for the 100 different users provided variety in the sets of user ACP, in agreement with what Kothari (2004) called the principle of local control.

Each run collected these values for use as inputs to derive the performance and accuracy metrics:

Table 1. The Values Collected from each Run

	Value Collected from Run	**WAA**	**ACPCDM**
1.	Number of ACP for the user evaluated in the run	x	x
2.	Known number of conflicts	x	x
3.	Number of conflicts identified	x	x
4.	Number of conflicts resolved	x	
5.	Time taken to identify conflict	x	x
6.	Time taken to resolve conflict	x	

These collected values had the expectation that for each run the authenticated user had their own set of ACP to use for authorization. The author tracked the number of ACP evaluated which was expected to vary by user. From the user's ACPs we also obtained the number of known conflicts among the ACPs. The expectation was that the number of known conflicts is greater than or equal to the number of the conflicts resolved. The time taken to identify conflict was also tracked for both algorithms to facilitate comparison between the two algorithms. The time taken to resolve conflict and the number of conflicts resolved could only be collected from WAA because ACPCDM does not resolve ACP conflict. The time for identifying and resolving ACP conflict were added to show the delay that running either algorithm would add to a system that places either algorithm in their authorization module.

The values as shown in Table 1 were collected from runs through both algorithms. Once collected for both WAA and ACPCDM algorithms, they were aggregated into the performance metrics shown in Table 2. These metrics collected from the runs through the algorithms are as follows:

Table 2. The Collection of Metrics from the Simulation

	Metric Collected	**WAA**	**ACPCDM**
1.	Number of runs	x	x
2.	Total time taken for the runs (T)	x	x
3.	Average time per run – average delay added to the authorization	x	x
4.	Number of ACP evaluated (N)	x	x
5.	Average number of ACPs per run	x	x
6.	Known number of conflicts (C_a)	x	x
7.	Number of conflicts identified (C_i)	x	x
8.	Total time taken to identify conflicts (T_i)	x	x
9.	Average time to identify conflicts per ACP	x	x

Table 2. The Collection of Metrics from the Simulation

	Metric Collected	WAA	ACPCDM
10.	Number of conflicts resolved (C_r)	x	
11.	Time taken to resolve ACP conflict (T_r)	x	
12.	Average time taken per ACP conflict	x	
13.	Average Resolve time taken per run	x	

The simulation was configured to use one algorithm during execution. For either algorithm the configuration was set for one of the algorithms as shown in Figure 7 or Figure 8. The application for the simulation ran in a MS Windows environment from the command prompt. Then following the logic in Figure 6 the metrics for Table 2 were collected as follows:

1. *Number of runs* – this metric is a count of how many times an algorithm was executed during a simulation run. Each of the runs consisted of a single user's authorization process.
2. *Total time taken for the runs (T)* – this is a sum of the time taken for all the runs beginning after retrieving the set of ACP of the user and running through the algorithm.
3. *Average time per run (average delay added to the authorization)* – this was determined by the dividing the total time (T) by the number of runs.
4. *Number of ACP evaluated (N)* – this was the total number of ACPs evaluated from each run through the algorithm.
5. *Average number of ACPs per run* – this is the number of ACPs evaluated divided by the number of runs through the algorithm.

6. *Known number of conflicts (C_a)* – this is the number of ACPs created with conflicts to represent the conflicts that would occur in systems.
7. *Number of conflicts identified (C_i)* – this represents the number of conflicted ACPs that the algorithm identified during the runs.
8. *Total time taken to identify conflicts (T_i)* – this is the total of all the time that the algorithm took to identify the ACP conflicts for each run.
9. *Average time to identify conflicts per ACP* – this metric considered the time it took to identify ACP conflicts over the number of ACPs considered for all the runs.
10. *Number of conflicts resolved (C_r)* – this counted how many ACP conflicts were resolved.
11. *Time taken to resolve ACP conflict (T_r)* – this tracked all the time taken to resolve ACP conflict for all the runs.
12. *Average time taken per ACP conflict* – this considered the time taken to resolve conflicts divided by the number of ACP.
13. *Average Resolve time taken per run* – this was calculated from the T_r over the number of runs to determine how the average time taken for ACP conflict resolution for each run.

From these metrics in Table 2 the study obtained the following: *accuracy* of the algorithm and the *performance* of the authorization process while using the algorithm. The performance of the algorithm was from tracking the number of conflicts resolved per run as well as the delay each run added to authorizing a user. The accuracy of the algorithm compared the number of conflicts identified to that of known conflicts for each run as well as the conflicts resolved relative to those identified. Thus, we have the following working definitions:

1. Accuracy for conflict identification: $A_i = \frac{C_i}{C_a}$ Where:
   a. A_i is the accuracy for identifying ACP conflict
   b. C_i is the number of conflicts identified
   c. C_a is the number of known conflicts
2. Accuracy for conflict resolution: $A_r = \frac{C_r}{C_i}$ Where:
   a. A_r is the accuracy for resolving ACP conflict
   b. C_i is the number of conflicts identified
   c. C_r is the number of conflicts resolved
3. Performance of the algorithm $P = \frac{T_i+T_r}{N}$ where:
   a. T_i is the time taken to identify conflict
   b. T_r is the timer taken to resolve conflict
   c. N is the number of runs

Formats for Presenting Results

The results of the study proved that the WAAM and WAA provided an approach that would solve ACP conflict in RBAC systems. The performance and accuracy results were presented comparing ACPCDM with WAA for all data points relevant to identifying ACP conflict. Further results for performance and accuracy of resolving ACP conflict only pertained to WAA. The administrator interaction with the WAAM was presented in narrative form to show the sequence of events required to create and update the weighted attributes, and logic of the conclusion.

Resource Specifications

In order to collect performance and accuracy metrics from the runs through the algorithms the author set up a computing environment with a RDBMS to implement WAAM and containing user and ACP records, and an IDE to contain the WAA logic and necessary components to communicate with the database. The simulation for a node in a DSE was represented on one PC containing the user data and ACP for authorization. (A DSE with multiple nodes would require that upon user authentication, the ACP for the user would be retrieved from the authenticating node using the DSE connectivity. The connectivity between the nodes could be achieved over a LAN/WAN. The ACP for the user could be retrieved either directly from the remote database or via API on the authenticating node.) For this effort a personal computer with the following features was used.

Hardware

One ASUS Q550L Notebook PC with a single Intel's Core i7-4500 2.39 GHz processor was used for this study. It had 8.0 GB of RAM and over 700GB of free space on the hard drive. The runs in the simulation were performed while the PC was plugged into direct current rather than using battery power.

Software

The software used comprised of the following:

- Operating system: Microsoft's Windows 8.1 running as a 64-bit system
- The languages used were T-SQL for the database and C# for the compiled code

- The relational database was Microsoft SQL Server 2012 - 11.0.2218.0 (X64) Express Edition (64-bit). This also provided the SQL Server Management Studio which is an IDE (integrated development environment)
- The integrated development environment (IDE) consisted of MS Visual Studio 2012 (Shell Integrated) version 11.0.20727.1 RTMREL. This was used for all the programming tasks for the compiled portion of the simulation.
- These available components used to perform required tasks for the simulation to run as an application as well as to simulate runs through the algorithm:
    - MS .Net Runtime 4.5.51641 – necessary to use the tools needed for rapid development using the Visual Studio IDE.
    - System Data – is a dynamic link library provided by Microsoft to connect to the database. This was used to encapsulate all the requests to the database to read and write data.
    - MS Quality Tools Unit Test Framework – these provided a unit testing feature which was used to target the necessary parts of the simulation for the algorithm to collect data.
- A logging mechanism was created as part of this study because to collect the data to meet the objective of calculating performance and accuracy metrics. The data in this log also had to be easily retrievable and used in a spreadsheet for analysis. So part of the implementation of WAA and WAAM in this environment was to collect data at selected points (such as after ACP conflict identification and at completion) during each run through the authorization process and logged into one record at the conclusion of each run. These data collected in the

AuthenticationLog table (see Appendix E for database implementation) corresponded with the performance and accuracy metrics in Table 1.

- Documentation of the results was done using Microsoft Office 2010 by copying the SQL query results into a spreadsheet in MS Excel.

Summary

The research methodology described the procedures essential to this research process and how the WAA and supporting WAAM were designed. Following the procedures also provided the performance and accuracy metrics for qualitative analysis for WAA and ACPCDM. Following the procedures provided was enough to achieve the same results from the algorithm in either a single system or in a distributed system environment.

Chapter 4

Results

This chapter presents the results obtained from following the procedures outlined in Chapter 3. Following those procedures, including making runs through the algorithms, this study collected performance metrics from runs through WAA and ACPCDM. This chapter discusses these metrics along with the findings from analyzing them.

Data Analysis

The performance values collected from each run through the algorithm provided details from which we could draw some conclusions when analyzing summary of the metrics from the entire dataset. This section discussed general observations from the results, performance of the algorithms, accuracy, and ease of use from the administrator.

General Observations of the Seed Data used for the Runs

The data used for the runs was common for both algorithms so that any comparisons in performance and accuracy were from the same inputs. An example of a user's set of ACP used for one of the runs is shown in Appendix G:. It shows most of the ACPs for a user who was assigned roles which in turn had access to objects, as well as access assigned directly to the user where there was no role. From the user's ACP in Appendix G:, a sample of conflicted ACP was extracted and shown in Table 3 below:

Table 3. Example of ACPs in conflict

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
98	Object - 98	Deny	1	2	Executive	Role - 35	1
98	Object - 98	Full	1	2	Executive	Role - 75	3

The data in Table 3 was from the ACP list of a single user instance that that was subscribed to multiple roles. The expectation here was that the actual person assigned this user instance only had one user account to authenticate into the system. The ACP in Table 3 showed that two of the roles that the user is subscribed to have access to the same object, however the access to the object are not in agreement causing ACP conflict.

There were metrics that were common to the runs of both the algorithms in the investigation. These are listed in Table 4 below.

Table 4. The Common Summary Metrics Collected from the Runs

Metric	**Value**
Number of Runs	100
Total number of ACP processed	23,644
Average ACP per run	236
Minimum ACP count in the runs	69
Maximum ACP count for the runs	964
Median of the ACP count for the runs	182

The values collected according to Table 1 are in Appendices E and F and are summarized in Table 5 below.

Table 5. The Metrics Collected from the Comparing the Algorithms

	Metric Collected	**WAA**	**ACPCDM**
1.	Number of runs	100	100
2.	Total time taken for the runs (T)	36.5122s	1.672s
3.	Average time per run	0.3651s	$1.672\text{x}10^{-2}$s
4.	Number of ACP Evaluated (N)	23,644	23,644

Table 5. The Metrics Collected from the Comparing the Algorithms

	Metric Collected	WAA	ACPCDM
5.	Known number of conflicts (C_a)	5,866	5,866
6.	Number of conflicts identified (C_i)	5,866	5,866
7.	Total time taken to identify conflicts (T_i)	0.2201s	0.2117s
8.	Average time to identify conflicts per ACP	3.7521×10^{-5}s	3.6089×10^{-5}s
9.	Number of conflicts resolved (C_r)	5,683	
10.	Total time taken to resolve ACP conflict (T_r)	34.7956s	
11.	Average time taken per ACP conflict	6.1227×10^{-3}s	
12.	Average resolve time taken per run	0.3479s	
13.	Average delay added to authorization	0.3502s	2.12×10^{-3}s

These metrics show how the runs through the two algorithms compared.

Performance of the Algorithms

The algorithms were designed to be part of a system's user authorization process so any time taken to run through the algorithm was considered an additional delay to the authorization. Table 6 represents the delay when the algorithms were used.

Table 6. Comparison of Delay added to Authorization Process

Metric Collected	WAA	ACPCDM
Total time taken to identify conflicts (T_i)	0.2201s	0.2117s
Average time to identify conflicts per run	2.201×10^{-3}s	2.12×10^{-3}s
Average time to identify conflicts per ACP	3.7521×10^{-5}s	3.6089×10^{-5}s
Total time taken to resolve ACP conflict (T_r)	34.7956s	
Average time to resolve ACP conflict per run	0.348s	

Table 6. Comparison of Delay added to Authorization Process

Metric Collected	WAA	ACPCDM
Average delay added to authorization	0.3502s	$2.12\text{x}10^{-3}$s

Comparing the delay added to authorization when all that is required is identifying the conflicts shows that on average ACPCDM ($2.12\text{x}10^{-3}$s) is marginally faster than WAA ($2.201\text{x}10^{-3}$s) by $8.1\text{x}10^{-5}$s. When considering the conflict resolution which WAA provided the average overall delay added to authorization is 0.3502s.

Table 7. The Summary Metrics Collected from the Weighted Attribute Algorithm

	ACPs in Run	Conflicts Found	Conflicts Found and Resolved	Delay Added to Authorization Time
Average	236	58.66	56.83	0.3502s
Minimum	69	7	7	0.0395s
Maximum	964	199	197	1.4162s
Median	182	47	45	0.2866s

Looking at these same metrics but classifying the user based on how many ACP the user had, there was a direct correlation between the number of ACP that a user had and the delay that would be added to authorization. The delay added to authorization is the sum of the time taken to identify the conflict and the time to resolve the conflict.

Table 8. The Delay Added to Authorization by WAA by on User ACP Count

User ACP Count	Number of Users	ACP	Conflict Actual	Conflict Resolved	Average Delay per User
<= 100	12	1050	164	155	0.1066s
101 - 200	44	6672	1538	1467	0.2112s
201 - 300	29	7073	1970	1897	0.4429s
301 - 400	8	2668	801	785	0.5437s
>400	7	6181	1393	1379	1.2495s

Accuracy in Identifying Conflicts for WAA and ACPCDM

The target for accuracy in identifying ACP conflicts was 100%. The accuracy was obtained by comparing the actual conflict and the conflicts identified. Using the values from Table 5, the accuracy results for WAA and ACPCDM are shown below.

Table 9. Accuracy Comparison of ACP Conflict Identification

Formula for Accuracy of Identifying Conflicted ACP	**WAA**	**ACPCDM**
Known number of conflicts (C_a)	5,866	5,866
Number of conflicts identified (Ci)	5,866	5,866
$A_i = \frac{C_i}{C_a}$	100%	100%

The accuracy from both algorithms is comparable and shows that WAA was able to identify the ACP conflicts as well as ACPCDM so systems adopting WAA would still obtain the desired accuracy.

Accuracy in Resolving Conflicts using WAA

The accuracy score desired was 100% resolution for all conflicts identified though the results as shown in Table 10 below are different.

Table 10. Overall Accuracy of ACP Conflict Resolution

Accuracy of Identifying Conflicted ACP	**WAA**
Number of conflicts identified (C_i)	5,866
Number of conflicts resolved (C_r)	5,683

Accuracy of Identifying Conflicted ACP	**WAA**
Conflict resolution accuracy $A_r = \frac{C_r}{C_i}$	96.98%

The accuracy shows that about 3% of conflicts were not resolved by WAA. The ACP from objects with the *is-auto-resolved* flag set to false were ignored from resolution by WAA and but were logged for the administrator.

Looking into the accuracy of ACP conflict resolution further suggested that ACP conflicts would be more prominent with users with a lower number of ACP as shown in Table 11 below. The table has the number of ACP a user would have classified into bands of 100.

Table 11. Accuracy of ACP Conflict Resolution by User ACP

User ACP Count Range	Number of Users	ACP Count	Conflict Actual	Conflict Identified	Conflict Resolved	Resolve Accuracy
0 - 100	12	1050	164	164	155	94.512%
101 - 200	44	6672	1538	1538	1467	95.384%
201 - 300	29	7073	1970	1970	1897	96.294%
301 - 400	8	2668	801	801	785	98.002%
>400	7	6181	1393	1393	1379	98.995%

From the accuracy metrics collected, the users with significantly higher ACP had higher conflict resolution accuracy.

Findings

After following the procedures for the study and performing the analysis above, the findings are: that there is a delay added to the authorization process, there are benefits of implementing WAA. The benefits of having ACP conflicts resolved were considered against the delay experienced in the authentication process.

The Delay added by using WAA in User Authorization

Using WAA for ACP conflict resolution added delay of 0.3502s on average to user authorizations. The expectation for this delay is that it would continue to decrease as the administrators correct the ACP conflicts. Because of this average delay that WAA would add to a system's authorization, the author recommended that WAA be applied at the beginning of a session rather than at each request. For research question (RQ) 1, we found that implementing WAAM in isolation did not provide any benefits to a system and that WAA would also need to be implemented.

As the system grows, there is an expectation that there will be new objects, roles, and users causing the ACP in the system to grow. Looking at RQ 2, WAA from the study would still be usable because, the ACPs can be processed by WAA at 2.201 x10^{-3}s per ACP. The ACP sets ranging in size from 69 to 964 per user were processed during authorization with the median ACP set being 182. This wide range shows that there is room for ACP counts to grow and still provide ample performance using the current WAA design and configuration. If there are too many ACP for WAA to process in time that is acceptably for the users, then the option to consider would be to run WAA as part of administrative tasks instead as part of each authentication.

User Benefits of using WAA

The systems that implement WAA in their authorization process will provide ACP sets that are free of conflict. The benefit to the users would be expected behavior with the removal of system anomalies resulting from ACP conflict. Using WAA provides users with a working set of ACP without conflicts as any ACP conflicts encountered were either resolved or removed from the user's ACPs.

Administrator Benefits of using WAA and WAAM

The administrator of the systems that has implemented WAA will benefit from knowing which ACPs need correcting by monitoring the ACP conflict log. Responding to the ACP conflicts recorded in the logs provided by WAAM could be an additional task for the administrators who did not have any ACP conflict monitoring. Considering RQ 2, WAA logs the conflicts from the user authorizations so that the administrators would have access to the ACPs that are constituted in conflict. The administrators can then formulate options to correct the configurations to be free from conflict. The administrators can then formulate options to correct the configurations to be free from conflict.

Another anticipated gain from using WAA was to have fewer users requesting their access to be corrected by the administrator. Correcting the conflict would improve the system because users will have their ACP conflicts corrected providing access to the system without having the system behavior altered by ACP conflict. Correcting the ACP conflicts proactively by the administrator would reduce the number of ACP conflict for each user.

Summary of Results

This chapter reviewed the findings from the analysis performed in this study from the metrics that were collected from runs through WAA. There were comparisons to ACPCDM in ACP conflict identification in which both algorithms scored 100% in accurately identified each ACP conflict. The difference in performance for WAA in ACP conflict identification from that of ACPCDM, taking 8.1×10^{-5}s longer per user authorization, was negligible.

Chapter 5

Conclusions

Overview

This chapter begins with a review of the idea and goals of the study along with the investigation of the research questions. Also provided is a summary of the analysis and conclusions from the study. The chapter concludes with recommendations for future studies, implications of the study, and summary of contributions.

The study began with the idea that as systems get larger administrators of these systems would, in the course of their administrative activities, inadvertently create conflicts in the ACP of the system. The conflicts then cause the system to misbehave when the users affected by the ACP conflict attempt to use the system and their ACPs have conflict. This study looked at ACP conflict in RBAC-based distributed systems and proposed a solution to automatically resolve the ACP conflict. The literature available was lacking a unified solution that includes both an administration model and an algorithm in distributed systems. The study by Oh, Sandhu, & Zhang (2006) showed an administration model but did not use hierarchical roles like WAAM. The studies by Fan, Liang, Luo, Bo, and Xia (2011) and Wu, Chen, Zhang, and Dai (2009) show use of algorithms to identify ACP conflict but do not resolve them like WAA. This study sought a solution to this problem and documented the results of the investigation of WAAM and WAA.

With studies that showed that ACP conflict detection was possible in RBAC (Fan et al., 2011) and would be logged for administrators to review and take appropriate action. The study by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009) showed that access conflicts are resolvable in a file access setting so the possibility of having ACP conflicts resolved in the RBAC-based system would be next step in the natural progression in studying access control.

This study sought a solution to resolve ACP conflict in RBAC systems in a DSE where conflicts were expected as part of the growth in usage systems over time. The approach used a simulation to prove that implementing WAA and WAAM would be sufficient to both identify and resolve ACP conflict. This approach of using a simulation to show results was used before by Reeder, Bauer, Cranor, Reiter, and Vaniea (2009). Further this study also included in its simulation ACPCDM (Fan, Liang, Luo, Bo, & Xia, 2011) which was an algorithm to identify ACP conflict. Including the ACPCDM in the simulation allowed for metrics to be collected from the same set of seed data and the metrics for analysis and comparison.

The results observed from the simulation showed that as far as identifying ACP conflict, WAA was comparable to ACPCDM with seemingly negligible difference in performance. The accuracy for identifying the ACP conflicts was identical so the metrics collected for conflict resolution distinguished WAA from other algorithms like ACPCDM (Fan, et al., 2011) and (Wu, Chen, Zhang, and Dai, 2009) which only identified ACP conflict.

This study also proposed that the conflicts in security policies' permissions and prohibitions were avoidable but would need both an administration model of hierarchical roles with a weighted priority attribute, and an extended algorithm to resolve resulting conflicts. In this study, the WPA was calculated using values provided by a system administrator when a user, role, or object record was created. The study by Fan, Liang, Luo, Bo, and Xia (2011) used an algorithm in the attempt to identify ACP conflict but lets the administrators resolve any conflicts. Using the weighted priority attribute provided an expressive declaration of precedence to assist in averting conflict. However if there still was any conflict beyond the weighted priority, an algorithm extending WAA could be used in a manner localized to the implementing system in the DSE.

The goal of the study was to propose improvements to the administration of ACP and to aid in detecting and resolving ACP conflicts in RBAC systems. This was accomplished by introducing attributes to the role hierarchy and user position objects used to create the ACP; and having the values from those new attributes used to calculate the WPA when there is conflict in a user's ACP. To that end WAAM and WAA in the simulation environment showed that while the conflicts were identified, not all conflicts are adequately resolvable using the proposed WAA using only the WPA. Also, for security reasons some conflicts were configured so that the authorization would not delegate the conflict resolution for WAA to resolve by setting the *is-auto-resolved* flag to false.

Outcomes

Investigating the data from the simulation provided the following results for the research questions from Chapter 1. These questions formed the basis of WAA to resolve ACP conflicts in large RBAC systems.

Research Question 1

The question was: What are the advantages and disadvantages of implementing WAAM in a RBAC system?

1. Implementing WAAM in an RBAC system provides the system with the advantage of being able to use WAA to identify and resolve ACP conflict.
2. A system implementing WAAM would allow its administrators to quickly react to fixing the underlying issue of ACP conflicts proactively as they are encountered.
3. One disadvantage of WAAM is that administrators of implementing systems must spend the time needed to implement WAAM into their administration model.
4. Another disadvantage is that there would be an additional item the administrators have to monitor to ensure the best use experience for their users. They have to monitor the conflict logs produced by WAA.

Research Question 2

The question was: What are the advantages and disadvantages of an RBAC system adopting the WAA as part of their use authorization?

1. The main advantage of adopting WAA is that the system will be able to resolve ACP conflicts for users as they get authorization to use the system. This would let users into the system while avoiding the anomalies that result from conflicted ACP.
2. The other advantage is that administrators will have a list of the ACP conflicts from the logs generated by WAA. This would allow the administrators to use the data proactively instead of waiting for user complaints before correcting the conflicted ACPs.

3. A disadvantage of using WAA is that it adds a delay to the authorization process of 0.3502s on average. This delay of 0.3502s was observed using the simulation environment used for this study. It would be possible that the delay is negligible if the servers used for implementing systems are more powerful than the computer used for the simulation.
4. The other disadvantage of using WAA is the administrators or implementers would have to include WAA into their system design and architecture which could prove challenging for systems already in use.

Research Question 3

The question was: Expecting the number of conflicts a systems user has to grow how would the proposed WAA resolve ACP conflicts?

1. The WAAM allows for a growing number of users, roles, securable objects and the resultant ACP. The values used to calculate the WPA will be set up when adding roles or positions of users. WAA will use the available attributes during authorization which is limited to a user's ACP. So while a system's number of users and objects grow it is possible that the average number of ACPs a user has in the system may not grow as fast. The expectation is that WAA would continue to process ACPs without noticeable degradation in system performance during user authorization.
2. The administrator would not be required to perform additional tasks beyond the attributes used to calculate WPA. So as the number of objects grows each new object will be assigned to at least one role in the hierarchy or at least one user will gain access to it. The ACP related to the new object will obtain its

attributes from the related user and role records to use for WPA. WAA has no restriction on the number of ACP it will process once implemented.

3. The administrators have a choice in determining when to authorize a user: on each request, or once when the user authenticates. For best performance using WAA, authorizing a user once per session after authentication would be the recommendation but the administrator would have to decide what option best meets their security needs. The hope would be that as conflicts are identified for the administrator, corrective action is taken because the largest delay from WAA is in resolving the conflicts.
4. The WAA is designed to work in compiled code and should be abstracted from the storage of the ACP. This allows for the horizontal scaling so that should there be a need to process more ACP an additional server instance could be acquired to provide the added computation without additional stress on the storage beyond basic retrieval for the ACPs.

Recommendations and Future Research

This WAA could be run as-is buying administrators time to resolve any identified conflict while providing ACP free of conflicts from their system's user authorization process. Implementing the WAAM and WAA would provide these benefits to the users and administrators of an RBAC-based system. The ACP conflicts encountered such as shown in Table 3 shows that the user was assigned roles with conflicting permissions to the same object. Some of the conflict had roles having the same hierarchy so WAA could not resolve the conflict because the WPA values were identical for the conflicted ACP. Looking at the data from the accuracy of resolving conflict, additional research would be required to determine other options, including if different attributes could be used to increase the accuracy of the resolved conflicts from 96.98% closer to 100%. Such attributes to be considered could be: time when the ACP was granted, or the physical location of a user. It may also be possible to study the use of the additional attributes and rules in an extension module to be used if an identified conflict cannot be resolved, or pass the conflicted ACP to a separate algorithm altogether.

Implication

The empirical data gathered from this investigation succinctly confirms that ACP conflicts in RBAC systems can be resolved by using WAA to resolve ACP conflicts. The structure of the data in the role hierarchy could influence the accuracy of resolving the conflict. If the hierarchical roles allow multiple root nodes, or have a flat hierarchy, the hierarchy attribute would be similar in many of the roles represented in a user's ACPs. The administrators would have to alter the configurable WAA constants as appropriate and further, the role attributes in order to achieve desired WPA to resolve ACP conflict.

The WAA is designed to be part of the authorization. This would cause a delay in the authorization when the algorithm runs but still benefits the users and administrators. If the delay added by implementing WAA in the authorization process is unacceptable, it could still be used periodically to check a system's ACP health by identifying the conflicts. This would be similar to the automated approach to security analysis by Abdunabi and Ray (2010) whereby the algorithm is run separately from the authorization.

Study Limitations

Three limitations were identified in this study. The first limitation is that the results were produced from randomized data created via script. The results, if collected from an actual system could vary what has been show in this study. The second limitation is that the simulation environment used only ran WAA during user authorization. Collecting the metrics from a fully functioning system would likely produce different results depending on availability of resources. Third, the simulation environment housed both WAA and WAAM on the same physical computer so the results may not be generalizable to DSE with nodes over a LAN/WAN.

Summary

This dissertation has shown that it is indeed possible to have a solution that combines an administration model and an algorithm in resolving ACP conflict for large RBAC-based systems with hierarchical roles in a DSE. Following the studies by Fan et al. (2013) which provided the ACPCDM for conflict identification and Reeder et al. (2009) with conflict resolution in a file access environment without role hierarchy, the natural progression in combining the results of those two studies would bring conflict resolution to administering RBAC-based systems.

Further, this study has shown that ACP conflicts can indeed be resolved in RBAC systems although not without challenges because 100% resolution accuracy was not achieved in all the runs. The algorithm could be extended to increase the resolution accuracy. It could be worth studying if 100% accuracy is an achievable limit in resolving ACP conflicts and whether the delay observed in resolving the conflicts can be lowered further.

Reference List

Abdunabi, R. and Ray, I. (2010). A Comparison of Security Analysis Techniques for RBAC Models. *Proceedings of the 2nd Annual Colorado Celebration of Women in Computing, Golden-Colorado, USA, November 4-5, 2010.* Retrieved February 23, 2014. http://www.cs.colostate.edu/~rabdunab/CCWIC2.pdf

Abdunabi, R., Ray, I., and France, R. (2013). Specification and Analysis of Access Control Policies for Mobile Applications. *Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, 173-184.

Atluri, V., Jajodia, S., and Bertino, E. (1997). Transaction Processing in Multilevel Secure Databases with Kernelized Architecture: Challenges and Solutions. *IEEE Transactions on Knowledge and Data Engineering, 9*(5), 697-708.

Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. (1996). A Temporal Access Control Mechanism for Database Systems. *IEEE Transactions on Knowledge and Data Engineering, 8*(1), 67-80.

Bertino, E., Catania, B., Ferrari, E., and Perlasca, P. (2003). A Logical Framework for Reasoning About Access Control Methods. *ACM Transactions on Information and System Security*, *6*(1), 71–127.

Bouzida, Y., Logrippo, L., and Mankovski, S. (2011). Concrete- and abstract-based access control. *International Journal of Information Security, 10*(4), 223-238.

Bruns, G., Huth, M., and Avijit, K. (2011). Program Synthesis in Administration of Higher-Order Permissions. *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, 41-50.

Capolsini, P. and Gabillon, A. (2009). Security policies for the visualization of Geo Data. *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, 2-11.

Cruz, I. F., Gjomemo, R., Lin, B., and Orsini, M. (2008). A location aware role and attribute based access control system. *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems,* 1-2.

Damiani, M. L., Bertino, E., Catania, B., and Perlasca, P. (2007). GEO-RBAC: A Spatially Aware RBAC. *ACM Transactions on Information Systems and Security, 10*(1), 1-42.

Damiani, M. L. and Silvestri, C. (2008). Towards movement-aware access control. *SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS,* 39-45.

Dekker, M. A. C., Crompton, J., and Etalle, S. (2008). RBAC Administration in Distributed Systems. *SAMCAT '08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies,* 93-101.

Downs, D. D., Rub, J., R., Kung, K. C., and Jordan, C. S. (1985). Issues in Discretionary Access Control. *IEEE Symposium on Security and Privacy*, 208-218.

El Kalam, A. A., Deswarte, Y., Baïna, A., and Kaaniche, M. (2007). Access Control for Collaborative Systems: A Web Services Based Approach. *IEEE International Conference on Web Services*, 1064-1071.

El Kalam, A. A., et al (2003). Organization based access control. *IEEE International Workshop on Policies for Distributed Systems and Networks*, 1-12.

Fan, B., Liang, X., Luo, Y., Bo, Y., and Xia, C. (2011). Conflict Detection Model of Access Control Policy in Collaborative Environment. *2011 International Conference on Computational and Information Sciences*, 377-381.

Ferraiolo, D. F. and Kuhn, D. R. (1992). Role Based Access Control. *15th National Computer Security Conference*, 554-563.

Greco, S., Leone, N., and Rullo, P. (1992). Complex: An Object-Oriented Logic Programming System. *IEEE Transactions on Knowledge and Data Engineering, 4* (4), 344–359.

Jajodia, S., Samarati, P., Sapino, M. L., Subrahmanian, V. S. (2001). Flexible support for multiple access control policies. *Transactions on Database Systems (TODS), 26* (2), 214-260.

Jiong, Q., and Chen-hua, M. (2012). Detecting and Resolving Constraint Conflicts in Role-Based Access Control. *2012 International Conference on Affective Computing and Intelligent Interaction, 10, 1-7.*

Joshi, J.B.D., Bertino, E., and Ghafoor, A. (2005). An Analysis of Expressiveness and Design Issues for the Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Dependable and Secure Computing,* 157-175.

Juntapremjitt, S., Fugkeaw, S., and Manpanpanich, P. (2008). An SSO-capable Distributed RBAC Model with High Availability across Administrative Domain. *22nd International Conference on Advanced Information Networking and Applications,* 121-126.

Karp, A. H., Haury, H., and Davis, M. H. (2009). From ABAC to ZBAC: The Evolution of Access Control Models. Technical report HPL-2009-30, HP Labs, February, 2009. Retrieved June 3, 2013, http://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf

Kern, A., Schaad, A., and Moffett, J. (2003). An Administration Concept for the Enterprise Role-Based Access Control Model. *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, 3-11.

Kothari, C. R. (2004). *Research Methodology: Methods and Techniques*. (2nd ed.). New Dheli, India: New Age International.

Kuang, T. P. and Ibrahim, H. (2009). Security privacy access control for policy integration and conflict reconciliation in health care organizations collaborations. *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, 750-754.

Li, N. and Mao, Z. (2007). Administration in Role-Based Access Control. *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security,* 1-12.

Moffett, J. D. and Sloman, M. S. (1994). Policy Conflict Analysis in Distributed System Management. *Journal of Organizational Computing, 4*(1), 1-22.

Muppavarapu, V., Pereira, A. L., and Chung, S. M. (2010). Role-based access control for a Grid system using OGSA-DAI and Shibboleth. *Journal of Supercomputing, 54* (2), 154-179.

Oh, S., Sandhu, R., and Zhang, X. (2006). An Effective Role Administration Model Using Organization Structure. *ACM Transactions on Information and System Security*, *9*(2), 113–137.

Park, J. and Sandhu, R. (2004). The UCON-ABC Usage Control Model. *ACM Transactions on Information Systems Security*, *7*(1), 128-174.

Pérez, M. G., Lòpez, G., Skarmeta, A. F. G, and Pasic, A. (2010). Advanced Policies for the Administrative Delegation in Federated Environments. *Third International Conference on Dependability,* 76-82.

Potel, M. (1996). MVP: Model-View-Presenter: The Taligent Programming Model for C++ and Java. Retrieved September 28, 2013, http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Reeder, R. W., Bauer, L., Cranor, L. F., Reiter, M. K., and Vaniea, K. (2009). Effects of Access-Control Policy Conflict-Resolution Methods on Policy-Authoring Usability. CMU-CyLab-09-006. Retrieved February 4, 2012, *http:www.ece.cmu.edu/~lbauer/papers/2009/cylabtr09-semantics.pdf.*

Sandhu, R. S., Bhamidipati, V., Coyne, E., Ganta, S., and Youman, C. (1997). The ARBAC97 Model for Role-Based Administration of Roles: Preliminary Description and Outline. *Proceedings of the Second ACM Workshop on Role-based Access Control*, 41-50.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-Based Access Control Models. *IEEE Computer, 29*(2) 38-47.

Schaad, A. and Moffett, J. D. (2002). A Lightweight Approach to Specification and Analysis of Role-based Access Control Extensions. *SACMAT 2002: 7th ACM Symposium on Access Control Models and Technologies*, 1-12.

Shafiq, B., Joshi, J. B. D., Bertino, E., and Ghafoor, A. (2005). Secure Interoperation in a Multi-domain Environment Employing RBAC Policies. *IEEE Transactions on Knowledge and Data Engineering, 17*(11), 1557-1577.

Shu, C., Yang, E. Y., and Arenas, A. E. (2009). Detecting Conflicts in ABAC Policies with Rule-Reduction and Binary-Search Techniques. *IEEE International Workshop on Policies for Distributed Systems and Networks,* 182-185.

Vaidya, J., Atluri. V., Warner, J., and Guo, Q. (2010). Role Engineering via Prioritized Subset Enumeration. *IEEE Transactions on Dependable and Secure Computing*, 300-314.

Wu, B., Chen, X., Zhang, Y., and Dai, X. (2009). An Extensible Intra Access Control Policy Conflict Detection Algorithm. *International Conference on Computational Intelligence and Security*, 483-488.

Yu-Cheng, H., and Gwan-Hwan, H. (2013). Chinese Wall Security Model for Workflow Management Systems with Dynamic Security Policy. *Journal of Information Science & Engineering*, *29*(3), 417-440.

Appendixes

Appendix A: Database Create Tables' Script

```
CREATE TABLE AcpConflictLog (
  LogTimeStamp DATETIME NOT NULL Default GetUtcDate(),
  UserId INTEGER NOT NULL,
  SecurableObjectId INTEGER NOT NULL,
  RoleId INTEGER NULL
);

CREATE TABLE AuthenticationLog (
  LogTimeStamp DATETIME NOT NULL DEFAULT GetUtcDate(),
  AcpCount INTEGER ,
  ConflictCount INTEGER,
  ResolveSeconds money,
  AuthenticationSeconds money
);

CREATE TABLE LuAccessMode (
  Id INTEGER NOT NULL,
  Name VARCHAR NULL,
  PRIMARY KEY(Id)
);

CREATE TABLE Position (
  Id INTEGER NOT NULL Identity(1,1),
  [Rank] INTEGER NULL,
  Name VARCHAR NULL,
  PRIMARY KEY(Id)
);

CREATE TABLE [Role] (
  RoleId INTEGER NOT NULL Identity(1,1),
  ParentRoleId INTEGER NOT NULL,
  Name INTEGER NULL,
  HierarchyLevel INTEGER NULL,
  PRIMARY KEY(RoleId)
);

CREATE TABLE RoleObject (
  LuAccessModeId INTEGER NOT NULL,
  SecurableObjectId INTEGER NOT NULL,
  RoleId INTEGER NOT NULL,
  IsAutoResolved Bit Default 0,
  PRIMARY KEY(RoleId, SecurableObjectId, LuAccessModeId)
);

CREATE TABLE SecurableObject (
  SecurableObjectId INTEGER NOT NULL Identity(1,1),
  Name VARCHAR NULL,
  PRIMARY KEY(SecurableObjectId)
);

CREATE TABLE UserRole (
  RoleId INTEGER NOT NULL,
  UserId INTEGER NOT NULL
);
```

```
CREATE TABLE UserSecurableObject (
  IsAutoResolved bit NOT NULL Default 0,
  UserId INTEGER NOT NULL,
  SecurableObjectId INTEGER NOT NULL,
  LuAccessModeId INTEGER NOT NULL,
  PRIMARY KEY(IsAutoResolved, UserId, SecurableObjectId)
);

CREATE TABLE [User] (
  UserId INTEGER NOT NULL Identity(1,1),
  PositionId INTEGER NOT NULL,
  UserName VARCHAR NULL,
  FirstName VARCHAR NULL,
  LastName VARCHAR NULL,
  [Password] VARCHAR NULL,
  PRIMARY KEY(UserId)
);

go
```

Appendix B: Database Create View Script

```
CREATE view [dbo].[AccessControlPolicyVw]
as
	select u.UserId, u.UserName, p.PositionId, p.[Rank] 'PositionRank',
so.SecurableObjectId, so.Name 'SecurableObjectName', am.Name 'AccessMode',
uso.IsAutoResolved, 0 'HierarchyLevel',
		0 'RoleId', p.Name 'PositionName', null 'RoleName'
	from dbo.[User] u
		join dbo.Position p on u.PositionId = p.PositionId
		join dbo.UserSecurableObject uso on u.UserId = uso.UserId
		join dbo.LuAccessMode am on uso.LuAccessModeId = am.LuAccessModeId
		join dbo.SecurableObject so on uso.SecurableObjectId =
so.SecurableObjectId

	union

	select u.UserId, u.UserName , p.PositionId, p.[Rank], so.SecurableObjectId,
so.Name 'SecurableObjectName', am.Name 'AccessMode', ro.IsAutoResolved,
rh.HierarchyLevel,
		rh.DescendantRoleId, p.Name 'PositionName', r.Name
		--, ur.RoleId
	from dbo.[User] u
		join dbo.Position p on u.PositionId = p.PositionId
		join dbo.UserRole ur on u.UserId = ur.UserId
		join dbo.[RoleHierarchy] rh on ur.RoleId = rh.RoleId
		join dbo.RoleObject ro on rh.DescendantRoleId = ro.RoleId
		join dbo.LuAccessMode am on ro.LuAccessModeId = am.LuAccessModeId
		join dbo.SecurableObject so on ro.SecurableObjectId =
so.SecurableObjectId
		join dbo.[Role] r on rh.DescendantRoleId = r.RoleId

GO
```

Appendix C: Database Create Stored Procedures' Script

```
CREATE procedure [dbo].[Role_readList]
       @ParentRoleId int = null
as
       select RoleId, Name, ParentRoleId, HierarchyLevel
       from [Role]
       where ParentRoleId = ISNULL(@ParentRoleId, ParentRoleId)
GO

CREATE proc [dbo].[Role_read]
(
       @RoleId Int
)
as

       select [RoleId], [ParentRoleId], [HierarchyLevel], [Name]
       from [Role]
       where RoleId = @RoleId
GO

CREATE proc [dbo].[Role_insert]
(
       @RoleId Int,
       @ParentRoleId Int = null,
       @HierarchyLevel Int = null,
       @Name Varchar(20)

)
as

       INSERT INTO [dbo].[Role]
       (
              ParentRoleId ,
              HierarchyLevel ,
              Name
       )
       VALUES
       (
              @ParentRoleId ,
              @HierarchyLevel ,
              @Name
       )

       select SCOPE_IDENTITY() 'RoleId'
GO

CREATE proc [dbo].[LuAccessMode_read]
(
       @LuAccessModeId Int
)
as



       select [LuAccessModeId], [Name]
       from [LuAccessMode]
```

```
	where LuAccessModeId = @LuAccessModeId
GO

CREATE proc [dbo].[LuAccessMode_insert]
(
	@LuAccessModeId Int,
	@Name Varchar(20)

)
as

	INSERT INTO [dbo].[LuAccessMode]
	(
		Name
	)
	VALUES
	(
		@Name
	)

	select SCOPE_IDENTITY() 'LuAccessModeId'
GO

CREATE proc [dbo].[Position_read]
(
	@PositionId Int
)
as

	select [PositionId], [Rank], [Name]
	from [Position]
	where PositionId = @PositionId
GO

CREATE proc [dbo].[Position_insert]
(
	@PositionId Int,
	@Rank Int = null,
	@Name Varchar(20) = null

)
as

	INSERT INTO [dbo].[Position]
	(
		Rank ,
		Name
	)
	VALUES
	(
		@Rank ,
		@Name
	)

	select SCOPE_IDENTITY() 'PositionId'
GO

CREATE proc [dbo].[AuthenticationLog_read]
```

```
(
	@LogTimeStamp Datetime,
	@AcpCount Int = null,
	@ConflictCount Int = null,
	@ResolveSeconds Money = null,
	@AuthenticationSeconds Money = null

)
as



	select [LogTimeStamp], [AcpCount], [ConflictCount], [ResolveSeconds],
[AuthenticationSeconds]
	from [AuthenticationLog]
GO

CREATE proc [dbo].[AuthenticationLog_insert]
(
	@LogTimeStamp Datetime,
	@AcpCount Int = null,
	@ConflictCount Int = null,
	@ResolveSeconds Money = null,
	@AuthenticationSeconds Money = null

)
as

	INSERT INTO [dbo].[AuthenticationLog]
	(
		LogTimeStamp ,
		AcpCount ,
		ConflictCount ,
		ResolveSeconds ,
		AuthenticationSeconds
	)
	VALUES
	(
		@LogTimeStamp ,
		@AcpCount ,
		@ConflictCount ,
		@ResolveSeconds ,
		@AuthenticationSeconds
	)
GO

CREATE proc [dbo].[AcpConflictLog_read]
(
	@LogTimeStamp Datetime,
	@UserId Int,
	@SecurableObjectId Int,
	@RoleId Int = null

)
as
```

```
        select [LogTimeStamp], [UserId], [SecurableObjectId], [RoleId]
        from [AcpConflictLog]
GO

CREATE proc [dbo].[AcpConflictLog_insert]
(
        @LogTimeStamp Datetime,
        @UserId Int,
        @SecurableObjectId Int,
        @RoleId Int = null

)
as


        INSERT INTO [dbo].[AcpConflictLog]
        (
                LogTimeStamp ,
                UserId ,
                SecurableObjectId ,
                RoleId
        )
        VALUES
        (
                @LogTimeStamp ,
                @UserId ,
                @SecurableObjectId ,
                @RoleId
        )
GO

CREATE proc [dbo].[RoleHierarchy_read]
(
        @RoleId Int,
        @DescendantRoleId Int,
        @HierarchyLevel Smallint

)
as

        select [RoleId], [DescendantRoleId], [HierarchyLevel]
        from [RoleHierarchy]
GO

CREATE proc [dbo].[RoleHierarchy_insert]
(
        @RoleId Int,
        @DescendantRoleId Int,
        @HierarchyLevel Smallint

)
as

        INSERT INTO [dbo].[RoleHierarchy]
        (
                RoleId ,
                DescendantRoleId ,
                HierarchyLevel
```

```
	)
	VALUES
	(
		@RoleId ,
		@DescendantRoleId ,
		@HierarchyLevel
	)
GO

CREATE proc [dbo].[SecurableObject_read]
(
	@SecurableObjectId Int
)
as

	select
		[SecurableObjectId], [Name]
	from [SecurableObject]
	where SecurableObjectId = @SecurableObjectId
GO

CREATE proc [dbo].[SecurableObject_insert]
(
	@SecurableObjectId Int,
	@Name Varchar(20)

)
as


	INSERT INTO [dbo].[SecurableObject]
	(
		Name
	)
	VALUES
	(
		@Name
	)

	select SCOPE_IDENTITY() 'SecurableObjectId'
GO

CREATE proc [dbo].[RoleObject_read]
(
	@LuAccessModeId Int,
	@SecurableObjectId Int,
	@RoleId Int,
	@IsAutoResolved Bit = null

)
as

	select
		[LuAccessModeId], [SecurableObjectId], [RoleId], [IsAutoResolved]
	from [RoleObject]
GO

CREATE proc [dbo].[RoleObject_insert]
```

```
(
	@LuAccessModeId Int,
	@SecurableObjectId Int,
	@RoleId Int,
	@IsAutoResolved Bit = null

)
as

	INSERT INTO [dbo].[RoleObject]
	(
		LuAccessModeId ,
		SecurableObjectId ,
		RoleId ,
		IsAutoResolved
	)
	VALUES
	(
		@LuAccessModeId ,
		@SecurableObjectId ,
		@RoleId ,
		@IsAutoResolved
	)
GO

create proc [dbo].[User_readByUsername]
	@UserName varchar(20)
as
	SELECT [UserId]
	      ,[PositionId]
	      ,[UserName]
	      ,[FirstName]
	      ,[LastName]
	      ,[Password]
	  FROM [dbo].[User]
	  where UserName = @UserName
GO

CREATE proc [dbo].[User_read]
(
	@UserId Int
)
as



	select
[UserId], [PositionId], [UserName], [FirstName], [LastName], [Password]
	from [User]
	where UserId = @UserId
GO

CREATE proc [dbo].[User_insert]
(
	@UserId Int,
	@PositionId Int,
	@UserName Varchar(20),
	@FirstName Varchar(20),
	@LastName Varchar(20),
	@Password Varchar(20)

)
as


	INSERT INTO [dbo].[User]
	(
		PositionId ,
		UserName ,
		FirstName ,
		LastName ,
		Password
	)
	VALUES
	(
@PositionId ,
@UserName ,
@FirstName ,
@LastName ,
@Password
	)

	select SCOPE_IDENTITY() 'UserId'
GO

CREATE proc [dbo].[UserSecurableObject_read]
(
	@IsAutoResolved Bit,
	@UserId Int,
	@SecurableObjectId Int,
	@LuAccessModeId Int

)
as



	select
[IsAutoResolved], [UserId], [SecurableObjectId], [LuAccessModeId]
	from [UserSecurableObject]
GO

CREATE proc [dbo].[UserSecurableObject_insert]
(
	@IsAutoResolved Bit,
	@UserId Int,
	@SecurableObjectId Int,
	@LuAccessModeId Int

)
as


	INSERT INTO [dbo].[UserSecurableObject]
	(
		IsAutoResolved ,
		UserId ,
		SecurableObjectId ,
		LuAccessModeId
	)
	VALUES
	(
@IsAutoResolved ,
@UserId ,
@SecurableObjectId ,
@LuAccessModeId
	)
GO

CREATE proc [dbo].[UserRole_read]
(
	@RoleId Int,
	@UserId Int

)
as

	select [RoleId], [UserId]
	from [UserRole]
GO

CREATE proc [dbo].[UserRole_insert]
(
	@RoleId Int,
	@UserId Int

)
as


	INSERT INTO [dbo].[UserRole]
	(
		RoleId ,
		UserId
	)
	VALUES
	(
@RoleId ,
@UserId
	)
GO

CREATE proc [dbo].[AccessControlPolicyVw_read]
(
	@UserId int
	, @SecurableObjectId int = null
)
as
	select
		UserId, UserName, SecurableObjectId, SecurableObjectName,
AccessMode, IsAutoResolved
	from AccessControlPolicyVw
	where UserId = @UserId
```

```
              and SecurableObjectId = IsNull(@SecurableObjectId,
SecurableObjectId)
GO
```

Appendix D: Tools and Components for Weighted Attribute Algorithm Implementation

1. A Microsoft Windows computer with the following specifications:
    a. Operating system: Windows 8.1 Pro
    b. Memory: 8GB RAM
    c. Processor: 4GHz (Intel CORE i7)
    d. Model: ASUS Q550L
2. Microsoft SQL Server Management Studio using SQL Server Express to create manage the relational database and run T-SQL scripts
3. Visual Studio 2012 IDE for the development environment using C# programming language.
4. Microsoft System components:
    a. ADO.Net containing System.Data and System.Data.SqlClient for data access.
    b. System.Generic.Collection for management of object lists
    c. System.Linq for lambda expressions on the lists
    d. Microsoft.VisualStudio.QualityTools.UnitTestFramework – for creating and running the unit tests for the runs through the simulation.

Appendix E: Authorization Log from Running the Weighted Attribute Algorithm

ACP Count	Known Conflicts	Conflicts Identified	Conflicts Resolved	Time to Identify Conflicts (Seconds)	Time to Resolve Conflicts (Seconds)
69	7	7	7	0.0002	0.0552
77	8	8	8	0.0003	0.0713
80	12	12	12	0.0003	0.0957
81	12	12	12	0.0003	0.1049
86	16	16	15	0.0001	0.0775
86	11	11	10	0.0004	0.0906
88	11	11	11	0.0001	0.0394
93	18	18	18	0.0001	0.067
95	17	17	13	0.0002	0.1219
95	11	11	10	0.0004	0.0888
100	19	19	19	0.0005	0.146
100	22	22	20	0.0008	0.1751
103	18	18	16	0.0002	0.0574
104	14	14	13	0.0005	0.114
104	21	21	17	0.0002	0.0645
105	17	17	17	0.0009	0.1659
108	16	16	16	0.0002	0.0516
112	17	17	17	0.0004	0.0598
119	18	18	18	0.0002	0.0607
124	25	25	25	0.0007	0.1987
125	24	24	24	0.0011	0.1909
131	36	36	34	0.0004	0.1022
132	18	18	17	0.0003	0.0535
132	28	28	28	0.0004	0.0786
135	31	31	30	0.0003	0.1126
135	31	31	30	0.0008	0.2573
137	28	28	25	0.0003	0.0973
138	27	27	22	0.0009	0.1356
141	33	33	32	0.0009	0.2679
143	33	33	32	0.0003	0.2403
146	35	35	35	0.001	0.2952
146	33	33	30	0.0003	0.1185
148	41	41	41	0.001	0.3395
150	34	34	32	0.0004	0.0976
150	33	33	31	0.0009	0.2763
158	38	38	37	0.0006	0.1204
159	37	37	36	0.0011	0.2906
164	47	47	44	0.0012	0.1693

ACP Count	Known Conflicts	Conflicts Identified	Conflicts Resolved	Time to Identify Conflicts (Seconds)	Time to Resolve Conflicts (Seconds)
165	40	40	38	0.0004	0.1827
165	37	37	36	0.0012	0.2989
168	43	43	42	0.0012	0.3502
171	36	36	34	0.0006	0.1206
172	40	40	40	0.0013	0.3343
172	45	45	44	0.0004	0.1349
172	41	41	39	0.0013	0.2885
175	47	47	46	0.0014	0.256
179	41	41	38	0.0004	0.145
179	45	45	42	0.0022	0.1701
180	42	42	40	0.0004	0.1248
182	44	44	38	0.0005	0.1718
182	46	46	44	0.0015	0.34
188	55	55	50	0.0019	0.513
189	48	48	47	0.0004	0.3767
193	48	48	46	0.0004	0.3331
195	55	55	53	0.0006	0.4292
196	52	52	51	0.0005	0.1556
202	52	52	50	0.0008	0.1758
203	53	53	52	0.0005	0.2829
204	53	53	53	0.0006	0.2978
204	55	55	52	0.0005	0.424
209	62	62	58	0.0033	0.4511
210	60	60	58	0.0005	0.1705
211	51	51	49	0.0019	0.2266
219	58	58	53	0.0025	0.4448
220	57	57	56	0.0006	0.4195
221	60	60	57	0.0006	0.3612
224	56	56	55	0.0074	0.3327
228	62	62	59	0.0028	0.5021
239	65	65	63	0.0008	0.4076
240	70	70	68	0.0013	0.2306
244	66	66	62	0.0007	0.4061
245	67	67	63	0.0008	0.4149
249	65	65	64	0.0036	0.5126
252	70	70	70	0.0007	0.5885
252	72	72	68	0.0009	0.567
253	77	77	75	0.0028	0.5155
270	74	74	72	0.003	0.5435
272	81	81	78	0.0034	0.5702
273	84	84	83	0.0017	0.4215
280	77	77	77	0.0043	0.38

ACP Count	Known Conflicts	Conflicts Identified	Conflicts Resolved	Time to Identify Conflicts (Seconds)	Time to Resolve Conflicts (Seconds)
282	73	73	71	0.0032	0.5721
288	84	84	78	0.0009	0.6423
290	87	87	84	0.0036	0.4969
290	90	90	83	0.001	0.7252
299	89	89	86	0.0011	0.2755
312	95	95	92	0.0044	0.528
318	96	96	92	0.0042	0.6575
320	96	96	96	0.005	0.4001
323	98	98	94	0.0012	0.3054
339	96	96	96	0.0012	0.4922
340	104	104	102	0.0049	0.5949
356	102	102	100	0.0048	0.6007
360	114	114	113	0.0084	0.5417
816	199	199	197	0.0075	1.1521
839	199	199	197	0.0062	1.41
854	199	199	197	0.0253	1.2242
880	199	199	197	0.0106	1.0236
900	199	199	197	0.0065	1.3274
928	199	199	197	0.0067	1.169
964	199	199	197	0.0316	1.1348

Appendix F: Authorization Log from Runs through ACPCDM

ACP Count	Known Conflicts	Conflicts Identified	Time Taken to Identify Conflicts (Seconds)
69	7	7	0.0003
77	8	8	0.0003
80	12	12	0.0001
81	12	12	0.0005
86	16	16	0.0004
86	11	11	0.0004
88	11	11	0.0001
93	18	18	0.0001
95	17	17	0.0001
95	11	11	0.0004
100	19	19	0.0001
100	22	22	0.0002
103	18	18	0.0001
104	21	21	0.0001
104	14	14	0.0002
105	17	17	0.0001
108	16	16	0.0009
112	17	17	0.0003
119	18	18	0.0002
124	25	25	0.0007
125	24	24	0.0007
131	36	36	0.0002
132	18	18	0.0012
132	28	28	0.0002
135	31	31	0.0008
135	31	31	0.0002
137	28	28	0.0002
138	27	27	0.0008
141	33	33	0.0002
143	33	33	0.0009
146	35	35	0.0009
146	33	33	0.0002
148	41	41	0.0002
150	33	33	0.0016
150	34	34	0.0004
158	38	38	0.0003
159	37	37	0.0011
164	47	47	0.0015
165	40	40	0.0004
165	37	37	0.0012
168	43	43	0.0012

ACP Count	Known Conflicts	Conflicts Identified	Time Taken to Identify Conflicts (Seconds)
171	36	36	0.002
172	45	45	0.0019
172	40	40	0.0003
172	41	41	0.0004
175	47	47	0.0015
179	41	41	0.0004
179	45	45	0.0004
180	42	42	0.0004
182	46	46	0.0004
182	44	44	0.0018
188	55	55	0.0004
189	48	48	0.0015
193	48	48	0.0066
195	55	55	0.0005
196	52	52	0.0016
202	52	52	0.0017
203	53	53	0.0005
204	53	53	0.0017
204	55	55	0.002
209	62	62	0.0011
210	60	60	0.0027
211	51	51	0.0023
219	58	58	0.002
220	57	57	0.0021
221	60	60	0.0006
224	56	56	0.003
228	62	62	0.0007
239	65	65	0.0034
240	70	70	0.0006
244	66	66	0.0025
245	67	67	0.0012
249	65	65	0.0031
252	70	70	0.0007
252	72	72	0.0009
253	77	77	0.0028
270	74	74	0.0034
272	81	81	0.0009
273	84	84	0.001
280	77	77	0.004
282	73	73	0.001
288	84	84	0.0009
290	87	87	0.0034

ACP Count	**Known Conflicts**	**Conflicts Identified**	**Time Taken to Identify Conflicts (Seconds)**
290	90	90	0.0034
299	89	89	0.001
312	95	95	0.0053
318	96	96	0.0041
320	96	96	0.0042
323	98	98	0.0011
339	96	96	0.0046
340	104	104	0.0012
356	102	102	0.0019
360	114	114	0.0055
816	199	199	0.0057
839	199	199	0.0059
854	199	199	0.006
880	199	199	0.0064
900	199	199	0.0295
928	199	199	0.0284
964	199	199	0.0072

Appendix G: List of a User's ACP used by WAA

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
2	Object - 2	Read	1	2	Executive	Role - 81	2
5	Object - 5	Read	1	2	Executive	Role - 45	1
6	Object - 6	Deny	1	2	Executive	Role - 71	1
6	Object - 6	Deny	1	2	Executive	Role - 85	4
6	Object - 6	Full	1	2	Executive	Role - 86	1
9	Object - 9	Deny	1	2	Executive	Role - 35	1
10	Object - 10	Full	1	2	Executive	Role - 75	3
10	Object - 10	Read	1	2	Executive	Role - 62	1
12	Object - 12	Full	1	2	Executive	Role - 86	1
15	Object - 15	Deny	1	2	Executive	Role - 91	3
16	Object - 16	Read	1	2	Executive	Role - 33	1
19	Object - 19	Read	1	2	Executive	Role - 96	1
20	Object - 20	Read	1	2	Executive	Role - 21	1
21	Object - 21	Deny	1	2	Executive	Role - 61	1
22	Object - 22	Full	1	2	Executive	Role - 45	1
23	Object - 23	Read	1	2	Executive	Role - 85	4
24	Object - 24	Deny	1	2	Executive	Role - 82	3
26	Object - 26	Read	1	2	Executive	Role - 91	3
28	Object - 28	Full	1	2	Executive	Role - 15	1
28	Object - 28	Full	1	2	Executive	Role - 72	2
29	Object - 29	Read	1	2	Executive	Role - 55	1
30	Object - 30	Deny	1	2	Executive	Role - 25	1
30	Object - 30	Deny	1	2	Executive	Role - 82	3
30	Object - 30	Deny	1	2	Executive	Role - 95	5
31	Object - 31	Full	1	2	Executive	Role - 65	2
32	Object - 32	Full	1	2	Executive	Role - 4	1
34	Object - 34	Full	1	2	Executive	Role - 15	1
35	Object - 35	Read	1	2	Executive	Role - 55	1
36	Object - 36	Deny	1	2	Executive	Role - 52	1
36	Object - 36	Deny	1	2	Executive	Role - 95	5
37	Object - 37	Full	1	2	Executive	Role - 92	4
38	Object - 38	Full	1	2	Executive	Role - 21	1
39	Object - 39	Read	1	2	Executive	Role - 61	1
41	Object - 41	Full	1	2	Executive	Role - 81	2
44	Object - 44	Deny	1	2	Executive	Role - 86	1
45	Object - 45	Read	1	2	Executive	Role - 71	1
48	Object - 48	Deny	1	2	Executive	Role - 75	3
48	Object - 48	Read	1	2	Executive	Role - 35	1
48	Object - 48	Read	1	2	Executive	Role - 91	3
49	Object - 49	Full	1	2	Executive	Role - 62	1
53	Object - 53	Read	1	2	Executive	Role - 52	1

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
54	Object - 54	Deny	1	2	Executive	Role - 92	4
55	Object - 55	Full	1	2	Executive	Role - 33	1
58	Object - 58	Full	1	2	Executive	Role - 96	1
60	Object - 60	Deny	1	2	Executive	Role - 62	1
61	Object - 61	Deny	1	2	Executive	Role - 45	1
62	Object - 62	Full	1	2	Executive	Role - 85	4
62	Object - 62	Read	1	2	Executive	Role - 82	3
63	Object - 63	Read	1	2	Executive	Role - 25	1
64	Object - 64	Deny	1	2	Executive	Role - 65	2
65	Object - 65	Full	1	2	Executive	Role - 35	1
65	Object - 65	Full	1	2	Executive	Role - 91	3
66	Object - 66	Deny	1	2	Executive	Role - 72	2
66	Object - 66	Read	1	2	Executive	Role - 75	3
67	Object - 67	Deny	1	2	Executive	Role - 15	1
68	Object - 68	Full	1	2	Executive	Role - 55	1
69	Object - 69	Read	1	2	Executive	Role - 95	5
70	Object - 70	Deny	1	2	Executive	Role - 21	1
71	Object - 71	Deny	1	2	Executive	Role - 4	1
71	Object - 71	Full	1	2	Executive	Role - 61	1
77	Object - 77	Deny	1	2	Executive	Role - 21	1
78	Object - 78	Full	1	2	Executive	Role - 61	1
78	Object - 78	Full	1	2	Executive	Role - 71	1
80	Object - 80	Full	1	2	Executive	Role - 25	1
80	Object - 80	Full	1	2	Executive	Role - 82	3
81	Object - 81	Read	1	2	Executive	Role - 65	2
82	Object - 82	Deny	1	2	Executive	Role - 62	1
84	Object - 84	Read	1	2	Executive	Role - 15	1
84	Object - 84	Read	1	2	Executive	Role - 72	2
85	Object - 85	Deny	1	2	Executive	Role - 55	1
86	Object - 86	Full	1	2	Executive	Role - 52	1
86	Object - 86	Full	1	2	Executive	Role - 95	5
87	Object - 87	Deny	1	2	Executive	Role - 33	1
87	Object - 87	Full	1	2	Executive	Role - 91	3
87	Object - 87	Read	1	2	Executive	Role - 92	4
88	Object - 88	Read	1	2	Executive	Role - 4	1
90	Object - 90	Deny	1	2	Executive	Role - 96	1
93	Object - 93	Read	1	2	Executive	Role - 45	1
94	Object - 94	Deny	1	2	Executive	Role - 85	4
95	Object - 95	Deny	1	2	Executive	Role - 71	1
96	Object - 96	Deny	1	2	Executive	Role - 96	1
98	Object - 98	Deny	1	2	Executive	Role - 35	1
98	Object - 98	Full	1	2	Executive	Role - 75	3
99	Object - 99	Read	1	2	Executive	Role - 62	1

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
101	Object - 101	Full	1	2	Executive	Role - 82	3
102	Object - 102	Full	1	2	Executive	Role - 25	1
103	Object - 103	Read	1	2	Executive	Role - 65	2
104	Object - 104	Deny	1	2	Executive	Role - 91	3
104	Object - 104	Read	1	2	Executive	Role - 33	1
105	Object - 105	Read	1	2	Executive	Role - 72	2
106	Object - 106	Read	1	2	Executive	Role - 15	1
107	Object - 107	Deny	1	2	Executive	Role - 55	1
107	Object - 107	Full	1	2	Executive	Role - 52	1
108	Object - 108	Full	1	2	Executive	Role - 95	5
108	Object - 108	Read	1	2	Executive	Role - 96	1
108	Object - 108	Read	1	2	Executive	Role - 92	4
109	Object - 109	Read	1	2	Executive	Role - 21	1
110	Object - 110	Deny	1	2	Executive	Role - 61	1
110	Object - 110	Read	1	2	Executive	Role - 4	1
111	Object - 111	Full	1	2	Executive	Role - 45	1
112	Object - 112	Deny	1	2	Executive	Role - 82	3
112	Object - 112	Read	1	2	Executive	Role - 81	2
112	Object - 112	Read	1	2	Executive	Role - 85	4
113	Object - 113	Deny	1	2	Executive	Role - 25	1
114	Object - 114	Full	1	2	Executive	Role - 65	2
116	Object - 116	Deny	1	2	Executive	Role - 71	1
116	Object - 116	Full	1	2	Executive	Role - 72	2
117	Object - 117	Full	1	2	Executive	Role - 15	1
118	Object - 118	Read	1	2	Executive	Role - 55	1
119	Object - 119	Deny	1	2	Executive	Role - 25	1
119	Object - 119	Deny	1	2	Executive	Role - 35	1
119	Object - 119	Deny	1	2	Executive	Role - 95	5
120	Object - 120	Full	1	2	Executive	Role - 65	2
120	Object - 120	Full	1	2	Executive	Role - 75	3
121	Object - 121	Read	1	2	Executive	Role - 62	1
125	Object - 125	Deny	1	2	Executive	Role - 52	1
126	Object - 126	Full	1	2	Executive	Role - 92	4
126	Object - 126	Read	1	2	Executive	Role - 33	1
127	Object - 127	Full	1	2	Executive	Role - 21	1
127	Object - 127	Read	1	2	Executive	Role - 61	1
129	Object - 129	Read	1	2	Executive	Role - 96	1
132	Object - 132	Full	1	2	Executive	Role - 45	1
132	Object - 132	Full	1	2	Executive	Role - 62	1
133	Object - 133	Read	1	2	Executive	Role - 85	4
134	Object - 134	Read	1	2	Executive	Role - 71	1
136	Object - 136	Read	1	2	Executive	Role - 35	1
137	Object - 137	Deny	1	2	Executive	Role - 75	3

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
137	Object - 137	Read	1	2	Executive	Role - 91	3
138	Object - 138	Full	1	2	Executive	Role - 62	1
138	Object - 138	Full	1	2	Executive	Role - 72	2
142	Object - 142	Full	1	2	Executive	Role - 4	1
143	Object - 143	Full	1	2	Executive	Role - 33	1
146	Object - 146	Full	1	2	Executive	Role - 96	1
148	Object - 148	Full	1	2	Executive	Role - 21	1
149	Object - 149	Read	1	2	Executive	Role - 61	1
150	Object - 150	Deny	1	2	Executive	Role - 45	1
150	Object - 150	Full	1	2	Executive	Role - 85	4
151	Object - 151	Read	1	2	Executive	Role - 82	3
152	Object - 152	Deny	1	2	Executive	Role - 65	2
152	Object - 152	Read	1	2	Executive	Role - 25	1
154	Object - 154	Full	1	2	Executive	Role - 91	3
155	Object - 155	Deny	1	2	Executive	Role - 72	2
156	Object - 156	Deny	1	2	Executive	Role - 15	1
157	Object - 157	Full	1	2	Executive	Role - 55	1
157	Object - 157	Read	1	2	Executive	Role - 95	5
160	Object - 160	Deny	1	2	Executive	Role - 4	1
163	Object - 163	Read	1	2	Executive	Role - 52	1
164	Object - 164	Deny	1	2	Executive	Role - 92	4
166	Object - 166	Full	1	2	Executive	Role - 71	1
168	Object - 168	Deny	1	2	Executive	Role - 81	2
170	Object - 170	Deny	1	2	Executive	Role - 62	1
172	Object - 172	Read	1	2	Executive	Role - 86	1
175	Object - 175	Full	1	2	Executive	Role - 35	1
175	Object - 175	Full	1	2	Executive	Role - 91	3
176	Object - 176	Deny	1	2	Executive	Role - 33	1
176	Object - 176	Read	1	2	Executive	Role - 75	3
179	Object - 179	Deny	1	2	Executive	Role - 96	1
182	Object - 182	Read	1	2	Executive	Role - 45	1
183	Object - 183	Deny	1	2	Executive	Role - 85	4
187	Object - 187	Deny	1	2	Executive	Role - 21	1
187	Object - 187	Deny	1	2	Executive	Role - 91	3
188	Object - 188	Deny	0	2	Executive		0
188	Object - 188	Full	1	2	Executive	Role - 61	1
188	Object - 188	Read	1	2	Executive	Role - 62	1
190	Object - 190	Full	1	2	Executive	Role - 25	1
190	Object - 190	Full	1	2	Executive	Role - 82	3
191	Object - 191	Read	1	2	Executive	Role - 65	2
192	Object - 192	Deny	1	2	Executive	Role - 62	1
193	Object - 193	Deny	1	2	Executive	Role - 91	3
194	Object - 194	Read	1	2	Executive	Role - 72	2

Object Id	Object Name	Access Mode	Is Auto Resolved	Position Rank	Position Name	Role Name	Hierarchy Level
195	Object - 195	Deny	1	2	Executive	Role - 55	1
195	Object - 195	Read	1	2	Executive	Role - 15	1
196	Object - 196	Full	1	2	Executive	Role - 52	1
196	Object - 196	Full	1	2	Executive	Role - 95	5
197	Object - 197	Deny	1	2	Executive	Role - 33	1
197	Object - 197	Read	1	2	Executive	Role - 92	4
198	Object - 198	Read	1	2	Executive	Role - 21	1
199	Object - 199	Deny	1	2	Executive	Role - 61	1
199	Object - 199	Read	1	2	Executive	Role - 4	1

Appendix H: Code to Generate the Data for the Simulation

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using KibwageDis.SqlData.Entities;

namespace KibwageDis.DataCreator
{
    class Maxes
    {
        public int AcpRole = 800;
        public int AcpUser = 200;
        public int Objects = 200;
        public int Roles = 100;
        public int Users = 500;
    }

    class Program
    {
        static void Main(string[] args)
        {
            string fileName =
System.Configuration.ConfigurationManager.AppSettings["output"]  + "gr.sql";
            if (File.Exists(fileName)) File.Delete(fileName);
            File.AppendAllText(fileName, TableNameDeletes());
            var mo = new MasterObject() { Max = new Maxes() };
            mo.Objects = DataObjects(fileName, mo.Max.Objects); // 200 records
            mo.Roles = DataRoles(fileName, mo.Max.Roles);     // 100
            DataAcpsByRole(fileName, mo);                // 950
            mo.Users = DataUsers(fileName, mo.Max.Users);     // 500
            DataUserRoles(fileName, mo);
            DataAcpsByUser(fileName, mo);
        }

        private static void DataAcpsByUser(string fileName, MasterObject mo)
        {
            File.AppendAllText(fileName, Environment.NewLine + "-- ACPs by user" +
Environment.NewLine);
            int acpCount = 0;
            var rnd = new Random();
            while (acpCount++ < mo.Max.AcpUser)
            {
                int uid = rnd.Next(1, mo.Users.Count);
                var user = mo.Users.Find(u => u.UserId == uid);
                int oid = rnd.Next(1, mo.Objects.Count);
                var secObject = mo.Objects.Find(o => o.SecurableObjectId == oid);
                var userObject = new UserSecurableObject
                {
                    IsAutoResolved = (acpCount % 5 != 0),
                    LuAccessModeId = (acpCount % 3 + 1),
                    SecurableObjectId = secObject.SecurableObjectId,
                    UserId = user.UserId
                };
                string line = string.Format(@"INSERT INTO
[dbo].[UserSecurableObject]
([IsAutoResolved],[UserId],[SecurableObjectId],[LuAccessModeId]) VALUES({0},
{1},{2},{3});
",
userObject.IsAutoResolved ? 1 : 0, userObject.UserId,
userObject.SecurableObjectId, userObject.LuAccessModeId);
                File.AppendAllText(fileName, line);
            }

            if (acpCount < 50)
                foreach (var user in mo.Users.FindAll(u => u.PositionId < 3))
                {
                    foreach (var secObject in mo.Objects)
                    {
                        var userObject = new UserSecurableObject
                        {
                            IsAutoResolved = (acpCount % 5 != 0),
                            LuAccessModeId = (acpCount % 3 + 1),
                            SecurableObjectId = secObject.SecurableObjectId,
                            UserId = user.UserId
                        };
                        string line = string.Format(@"INSERT INTO
[dbo].[UserSecurableObject]
([IsAutoResolved],[UserId],[SecurableObjectId],[LuAccessModeId]) VALUES({0},
{1},{2},{3});
",
     userObject.IsAutoResolved ? 1 : 0, userObject.UserId,
userObject.SecurableObjectId, userObject.LuAccessModeId);
                        File.AppendAllText(fileName, line);
                        if (acpCount++ > 50)
                            return;
                    }

                }
        }

        private static void DataUserRoles(string fileName, MasterObject mo)
        {
            var rnd = new Random();
            int acpCount = 0;
            var urs = new List<UserRole>();
            foreach (var user in mo.Users)
            {
                // set roles for the user
                int roleCount = rnd.Next(6, 12);
                for (int i = 0; i < roleCount; i++)
                {
                    var role = mo.Roles[rnd.Next(mo.Roles.Count)];
                    var ur = new UserRole { RoleId = role.RoleId, UserId =
user.UserId };
                    if (!urs.Exists(u => u.RoleId == ur.RoleId && u.UserId ==
ur.UserId))
                    {
                        urs.Add(ur);
                        string line = string.Format("INSERT INTO [dbo].[UserRole]
([RoleId],[UserId]) VALUES ({0},{1});\n", ur.RoleId, ur.UserId);
                        File.AppendAllText(fileName, line);
                        acpCount++;
                        Console.WriteLine("Acp {0}", acpCount);
                    }
                }
            }

        }

        class MasterObject
        {
            public List<Role> Roles { get; set; }
            public List<SecurableObject> Objects { get; set; }
            public List<User> Users { get; set; }

            public Maxes Max { get; set; }
        }

        private static List<SecurableObject> DataObjects(string fileName, int max)
        {
            File.AppendAllText(fileName, TableNameIdStart(Ts.SecObj));
            // create objects
            var items = new List<SecurableObject>();
            for (int i = 1; i < max + 1; i++)
            {
                var role = new KibwageDis.SqlData.Entities.SecurableObject {
SecurableObjectId = i, Name = NameGet("Object", i) };
                items.Add(role);
                string line = string.Format("INSERT INTO [dbo].[SecurableObject]
(SecurableObjectId, [Name]) VALUES ({0}, '{1}');\r",
                    role.SecurableObjectId, role.Name);
                File.AppendAllText(fileName, line);

            }
            File.AppendAllText(fileName, TableNameIdStop(Ts.SecObj));
            return items;
        }

        private static List<Role> DataRoles(string fileName, int max)
        {
            var items = new List<Role>();
            File.AppendAllText(fileName, TableNameIdStart(Ts.Role));
            // create roles
            for (int i = 1; i < max + 1; i++)
            {
                var role = new KibwageDis.SqlData.Entities.Role { RoleId = i,
HierarchyLevel = 1, ParentRoleId = 0, Name = NameGet("Role", i) };
                if (i > 1)
                    role.ParentRoleId = 1;
                if (i > 14)
                    role.ParentRoleId = i - 10;
                if (i > 30)
                    role.ParentRoleId = i - 20;
                if (i > 40)
                    role.ParentRoleId = (i % 10) + 10;
                if (i > 60)
                    role.ParentRoleId = i - 10;
                items.Add(role);
```

```
                string line = string.Format("INSERT INTO [dbo].[Role] (RoleId,
[ParentRoleId],[HierarchyLevel] ,[Name]) VALUES ({0}, {1}, {2}, '{3}');\r",
                    role.RoleId, role.ParentRoleId == 0 ? "null" :
role.ParentRoleId.ToString(), role.HierarchyLevel, role.Name);
                File.AppendAllText(fileName, line);
            }
            File.AppendAllText(fileName, TableNameIdStop(Ts.Role));
            return items;
        }

        private static List<User> DataUsers(string fileName, int max)
        {
            var items = new List<User>();
            File.AppendAllText(fileName, TableNameIdStart(Ts.User));
            // create roles
            for (int i = 1; i < max + 1; i++)
            {
                var role = new KibwageDis.SqlData.Entities.User { FirstName = "Fn
" + i.ToString(), LastName = "Ln " + i.ToString(),
                    Password= "qweNM<123", UserName = NameGet("User", i), UserId =
i, PositionId = 5 };
                if (i % 20 == 4)
                    role.PositionId = 3;
                if (i % 40 == 6)
                    role.PositionId = 4;
                if (i % 50 == 2)
                    role.PositionId = 2;
                if (i % 100 == 5)
                    role.PositionId = 1;
                items.Add(role);
                string line = string.Format("INSERT INTO [dbo].[User] (UserId,
[PositionId] ,[UserName] ,[FirstName] ,[LastName] ,[Password]) VALUES ({0}, {1},
'{2}', '{3}', '{4}', '{5}');\r",
                    role.UserId, role.PositionId == 0 ? "null" :
role.PositionId.ToString(), role.UserName, role.FirstName, role.LastName,
role.Password);
                File.AppendAllText(fileName, line);
            }
            File.AppendAllText(fileName, TableNameIdStop(Ts.User));
            return items;
        }

        private static void DataAcpsByRole(string fileName, MasterObject mo)
        {
            File.AppendAllText(fileName, Environment.NewLine + "-- ACPs by role" +
Environment.NewLine);
            // create roles
            var acps = new List<KibwageDis.SqlData.Entities.RoleObject>();
            foreach (var role in mo.Roles)
            {
                // get how many objects the role should have access to
                //int maxObjects = Math.Min(15, (new Random()).Next(1,
mo.Objects.Count));
                int maxObjects = (new Random()).Next(4, 12);
                Console.WriteLine("Role {0} MaxObjects {1}", role.RoleId,
maxObjects);
                int objectCount = 0;
                while(objectCount < maxObjects)
```

```
                    {
                        int oid = (new Random()).Next(1, mo.Objects.Count);
                        var obj = mo.Objects.Find(o => o.SecurableObjectId == oid);
                        var acp = new KibwageDis.SqlData.Entities.RoleObject
                        {
                            IsAutoResolved = true,
                            LuAccessModeId = (acps.Count % 3 +1), // get only Read,
Write, Deny

                            RoleId = role.RoleId,
                            SecurableObjectId = obj.SecurableObjectId
                        };

                        if (acps.Exists(ro => ro.RoleId == acp.RoleId &&
ro.SecurableObjectId == acp.SecurableObjectId))
                            continue; // skip because the ACP is already in the system
and insert will fail

                        string line = string.Format(@"INSERT INTO [dbo].[RoleObject]
([LuAccessModeId], [SecurableObjectId], [RoleId],[IsAutoResolved]) VALUES ({0},
{1}, {2}, {3});
",
                            acp.LuAccessModeId, acp.SecurableObjectId, acp.RoleId,
acp.IsAutoResolved ? 1 : 0);

                        if (acps.Count < mo.Max.AcpRole + 1)
                        {
                            objectCount++;
                            acps.Add(acp);
                            File.AppendAllText(fileName, line);
                        }
                        else break;
                    }
                }
                File.AppendAllText(fileName, Environment.NewLine);
            }

            private struct Ts
            {
                public const string
                    Role = "Role",
                    SecObj = "SecurableObject",
                    User = "[User]";
            }
            private static string TableNameDeletes()
            {
                string retVal = @"
delete from dbo.RoleObject
delete from dbo.[UserRole]
delete from [dbo].[UserSecurableObject]
delete from dbo.SecurableObject
delete from dbo.[Role]
delete from dbo.[User]

";
                return retVal;
            }

            private static string TableNameIdStart(string tableName)
```

```
        {
            string retVal = string.Format(@"
set IDENTITY_INSERT dbo.{0} on
", tableName);
            return retVal;
        }

        private static string TableNameIdStop(string tableName)
        {
            string retVal = string.Format(@"set IDENTITY_INSERT dbo.{0} off
", tableName);
            return retVal;
        }

        private static string NameGet(string tableName, int i)
        {
            string retVal = string.Format("{0} -{1,4}", tableName, i);
            return retVal;
        }
    }

}
```