INTEGRATING ACCESS CONTROL WITH REAL-TIME ASSESSMENT: ADAPTIVE SECURITY THROUGH THE ACQUISITION, ANALYSIS AND APPLICATION OF CONTEXT DATA

By

HASSAN RASHEED

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

© 2009 Hassan Rasheed

He (the Creator) is the Ever Living, none has the right to be worshipped but He; so invoke Him making your worship pure for Him alone. All the praise and thanks be to Allah, the Lord of all that exists. (Qur'an 40:65)

ACKNOWLEDGMENTS

I thank the Creator for His continuous mercy and favor and my parents for their continual support and self-less care; and I thank all of those who have contributed to my intellectual development throughout my studies.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS 4

LIST OF TABLES 9

LIST OF FIGURES 10

ABSTRACT 15

CHAPTER

1 INTRODUCTION 16

1.1 Motivation: Paradigm Shifts in System Security 16
1.1.1 Changing Nature of Attacks 16
1.1.2 Changing Deployment Environments 17
1.1.3 Greater Emphasis on Distributed Data Analysis 18
1.2 Challenges Faced 19
1.2.1 The Nature of Context Information 19
1.2.2 Applying Security Data for Improved Access Control 20
1.3 Approach 21
1.4 Summary of Results 21
1.5 Significance and Impact 22
1.6 Organization of this Report 23

2 RELATED WORK 24

2.1 Context Information 24
2.1.1 Existing Definitions of Context 24
2.1.2 Redefining Context 24
2.1.3 Context Representation 26
2.2 Systems Integration 26
2.2.1 Horizontal Integration 27
2.2.2 Vertical Integration 29
2.2.3 Summary on Integration 30
2.3 Integration of Security Controls 31
2.4 Use of Threat, Risk and Trust in Access Control 33
2.5 Intrusion Response 33

3 GENERAL APPROACH PART 1: CONTEXT ACQUISITION 35

3.1 Introduction and Design Goals 35
3.2 Survey of Context Acquisition Approaches 36
3.2.1 Closed/Coalition Approach 37
3.2.2 Open/Federation Approach 39
3.3 A Coalition-Based System for Context Acquisition 41

3.3.1 Information Model . . . 41
3.3.1.1 Access control . . . 41
3.3.1.2 Intrusion detection . . . 43
3.3.1.3 Intrusion response . . . 44
3.3.1.4 Model overview . . . 45
3.3.1.5 Security event . . . 45
3.3.1.6 Access request . . . 46
3.3.1.7 Intrusion attempt . . . 46
3.3.1.8 Intrusion response . . . 46
3.3.2 Implementation . . . 47
3.3.3 Summary of the Coalition Approach to Context Acquisition . . . 49
3.4 A Federated System for Context Acquisition . . . 49
3.4.1 Approach . . . 49
3.4.2 The Use of Event Correlation in Federated System . . . 50
3.4.3 Available Correlation Methods: A Taxonomy of Event Correlation Approaches for Security Data . . . 50
3.4.3.1 Taxonomy of alert correlation methods based on outcome . . . 50
3.4.3.2 Taxonomy of alert correlation methods based on means . . . 52
3.4.4 Selecting an Effective Ontology Approach . . . 54
3.4.5 A Hybrid Event Ontology . . . 55
3.4.5.1 The basis for a common ontology . . . 55
3.4.5.2 The proposed ontology . . . 56
3.5 Summary . . . 59

4 GENERAL APPROACH PART 2: CONTEXT ANALYSIS . . . 67

4.1 Introduction and Design Goals . . . 67
4.2 A High-Level Ontology of Security Assessment Information . . . 68
4.2.1 Core Assessments . . . 69
4.2.1.1 Risk . . . 69
4.2.1.2 Trust . . . 69
4.2.1.3 Dependability and importance . . . 70
4.2.2 Composite Assessments . . . 71
4.2.2.1 Threat . . . 71
4.2.2.2 Impact . . . 72
4.3 Summary . . . 72

5 GENERAL APPROACH PART 3: CONTEXT APPLICATION . . . 75

5.1 Introduction . . . 75
5.2 Context-Based Policy Evaluation . . . 75
5.2.1 Access Control Schema Extension . . . 76
5.2.2 Application Scenarios . . . 78
5.3 Context-Based Threat Response . . . 79
5.4 Summary . . . 81

6 ADAPTIVE RISK-AWARE ACCESS CONTROL FOR WEB SERVERS 87

6.1 Introduction 87
6.1.1 Connection Between the Implementation and Previous Chapters . . 87
6.1.2 Implementation Overview 88
6.2 Intrusion Response and Attack Resistance 89
6.2.1 Strategy Selection 89
6.2.2 Response Triggering 90
6.3 Notion of Risk and a Preliminary Risk Assessment Model 90
6.3.1 Analysis Model 90
6.4 Triggering Restricted Permissioning With Risk Data 93
6.5 Abacus Framework Architecture 94
6.6 Updates and Modifications to the Initial Model and Architecture 99
6.6.1 Performance Issues With the Initial Architecture 99
6.6.2 Solution 1: Caching 99
6.6.3 Solution 2: Redesigning the Analysis Algorithm and Refactoring the Architecture 100
6.6.4 Revised Risk Assessment Model 100
6.6.5 Restructured Architecture 101
6.7 Summary 101

7 RESULTS 106

7.1 Testing Setup 106
7.2 Validation of Analysis Model 106
7.3 Web Server Attack Resistance Results 108
7.4 Performance Analysis 111
7.4.1 Performance Testing Methodology 111
7.4.2 Performance of Initial Abacus Framework 112
7.4.3 Performance of Abacus Framework with Recursive Analysis Model . 113
7.4.4 Performance Comparison for ABACUS Framework and Ordinary Apache Web server 113

8 CONCLUSIONS 130

8.1 Conclusions Produced By Examination of the General Approach 130
8.1.1 Data Acquisition 130
8.1.2 Data Analysis 130
8.2 Conclusions On the Implementation and Testing of the Concrete Implementation 131
8.2.1 Data Quality 131
8.2.2 Changes From the General Approach to the Concrete Implementation 132
8.2.3 Effectiveness and Performance 133
8.3 Future Work 133

REFERENCES 135
BIOGRAPHICAL SKETCH 141

LIST OF TABLES

Table page

5-1 Escalation of threat in subsequent requests by two different sources. When the threat is assigned to individual sources seperately, the system is able to distinguish between malicious and non-malicious subjects. 83

5-2 Escalation of threat in subsequent requests by three different hosts on a common target. When the effect of requests from different subjects to the same object are considered in aggregate, the system is able to contextualize individual requests into an overall pattern of interaction with the object. 83

5-3 Selected intrusion response strategies. Each general response strategy is listed along with its appropriate use case, its implementation at the access control level and the contextual properties that constrain its application. 86

7-1 A summary of the simulation results for scenario one simulating an attack from a single source on multiple system resources. 119

7-2 A summary of the simulation results from scenario two while simulating an attack from multiple sources on a single system resource. 120

7-3 A summary of the simulation results from scenario three while simulating an attack from multiple sources on multiple system resources. 121

7-4 Traffic statistics for three top websites in December 2008. 129

7-5 Estimated peak performance for ABACUS framework with current testing constraints. 129

LIST OF FIGURES

Figure page

3-1 Diagram of a security coalition. Each security component has to interact with all of the other components in order to access their data. This architecture is limited in extensibility because each time a new member is added to the coalition, all of the other members must be adapted to use its interface. 60

3-2 The anatomy of a security component in an open architecture. The core decision mechanism is responsible for placing new events into the mechanisms event data store which subsequently provides the data to other consumers. The core decision mechanism also pulls data from the component's event consumer module in order to enforce policy based on external event information. The event consumer module includes a policy describing the different types of events that should be drawn from the event provider (this interaction is depicted in Figure 3-3). 61

3-3 Security components with a common event provider. Rather than having to interact with each other member of the system, the components can now access data through a common event provider. 62

3-4 The anatomy of a security component in an open architecture. The core decision mechanism is responsible for placing new events into the common event provider that is now an external service instead of the previous data store that was contained in the mechanism itself. The core decision mechanism also pulls data from the component's event consumer module in order to enforce policy based on external event information. The event consumer module includes a policy describing the different types of events that should be drawn from the event provider (this interaction is depicted in Figure 3-3). 63

3-5 Security event information model 64

3-6 The flow of data between an IDS and a web server under the coalition-based implementation. 64

3-7 Taxonomy of the means used to achieve alert correlation. 65

3-8 Ontology for inter-domain event correlation 66

4-1 Core assessment classes. Importance, trust and dependability assessments for entities. Threat and impact assessments for access requests. Value and risk assessments for the action of an access request. 73

4-2 Assessment factors for three assessment types: trust, risk and dependability. The class Assessment is also a subclass of Assessment Factor because of the composite assessments that are derived from other assessments. The assessment factors for threat are the risk of the request and the trust granted to the subject. The assessment factors for the impact are the dependability and importance of the object and the risk of the request. 74

5-1 XACML rule including source-centered threat. This rule demonstrates the extension of the XACML schema with a new property *total-source-threat.* This property is designated as an attribute of the subject of the request. An integer function is used to compare the value returned for this property with the designated value of 20. If the total-source-threat property is greater than or equal to this value, then the rule has the effect of causing the request to be denied. . . . 83

5-2 XACML rule including target-centered threat. This rule demonstrates the extension of the XACML schema with a new property *total-target-threat.* This property is designated as an attribute of the resource being accessed. An integer function is used to compare the value returned for this property with the designated value of 30. If the total-target-threat property is greater than or equal to this value, then the rule has the effect of causing the request to be denied. . . 84

5-3 XACML rule including an attribute indicating that a resource is locked. This rule demonstrates the extension of the XACML schema with a new property *resource-lock-status.* This property is designated as an attribute of the resource being accessed. A boolean function is used to compare the value returned for this property with the designated value of 'true'. If the resource-lock-status property is true, then the rule has the effect of causing all requests to this resource to be denied. 84

5-4 XACML rule including an attribute indicating that a user account is locked. This rule demonstrates the extension of the XACML schema with a new property *resource-lock-status.* This property is designated as an attribute of the subject initiating the request. A boolean function is used to compare the value returned for this property with the designated value of 'true'. If the user-account-lock-status property is true, then the rule has the effect of causing all requests from this source to be denied. 85

5-5 XACML rule including a property to restrict a specific permission . This rule demonstrates the extension of the XACML schema with a new property *user-write-prohibit.* This property is designated as an attribute of the subject initiating the request. The Policy Decision Point will be extended with a new module that provides the logic to provide a current value for this property. An boolean function is used to compare the value returned for this property with the designated value of 'true'. If the user-write-prohibit property is true, then the rule has the effect of causing the request to be denied. 85

6-1 Sample risk progression for an intruder executing intrusive requests of moderate severity (5) . . . 103

6-2 Architecture for the ABACUS framework. . . . 103

6-3 Apache configuration directive that establishes a *SourcePermissionRestrict* access handler to evaluate all requests to resources in the directory '/s'. The directive also establishes four risk thresholds, each for a different action. These thresholds are subsequently used by the access handler to compare against the current risk evaluation for the source of the request, with the request being denied if the source's risk exceeds the threshold. The final variable *SourceLockoutThreshold* establishes that one the risk attached to the source exceeds 41, all requests from that source will be denied. . . . 104

6-4 Psuedocode for the access control model that performs restriction of source permissions based on a risk assessment obtained from an analysis server. It retrieves a risk assessment for the source from the analysis server and then compares it with the appropriate threshold for the action being performed. . . . 104

6-5 Apache configuration directive for a custom authentication handler. Three different thresholds, or properties are established which could be used to trigger the use of authentication. A value is also set for *AuthExpiration* which ensures that, one authenticated, users are only re-authenticated every 300 second (five minutes) at most. . . . 105

6-6 Pseudo code for authentication module. Authentication is required if any of the established risk thresholds are exceeded. . . . 105

7-1 Simulation results from the validation of the analysis model showing risk estimates for the sources detected as intrusive. A) using only concrete vulnerability filtering B) using concrete vulnerability filtering and configuration verification. . . 118

7-2 Simulation results from the validation of the analysis model showing risk estimates for targets being attacked by intrusive requests. A) using only concrete vulnerability filtering B) using concrete vulnerability filtering and configuration verification. . . . 119

7-3 Access control policies for the two servers during scenario one while simulating an attack from a single source on multiple system resources. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses source risk data and sets a threshold of 45 for the source risk, beyond which, requests from that source will be denied. . . . 120

7-4 The growth of the risk from the intruder in scenario one. . . . 120

7-5 Access control policies for the two servers during scenario two while simulating an attack from multiple sources on a single system resource. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses target risk data and sets a threshold of 45 for the target risk, beyond which, requests to that target will be denied. 121

7-6 The growth of risk for the targeted resource in scenario two. 121

7-7 Access control policies for the two servers during scenario three while simulating an attack from multiple sources on multiple system resources. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses three different risk properties to trigger the requirement of authentication. The system risk threshold is 65, the source risk threshold is 33 and the target risk threshold is 45. A time limit for the expiration of a valid authentication is set at 300 seconds using the *AuthExpiration* property. 122

7-8 Statistics for ABACUS framework version 1 during a simulation with ten concurrent users, one of which was an intruder. Graphs show time to serve requests for different breakdowns of the set of requesting users. A) requests from all users B) requests from the intruder C) requests from non-intrusive users. These graphs establish that the time to process requests was increasing throughout the simulation and that this was due to the increased time in took to process requests from the intruder that required more data to be aggregated and analyzed in order to produce a risk assessment. 123

7-9 Statistics for ABACUS framework version two. A) time to serve requests for the web server B) time to serve requests for the analysis server. The graphs correspond to a simulation with 100 concurrent users for the entire duration of the test 10 minute stress test. 124

7-10 Statistics for ABACUS framework version two. A) alert processing time B) alert receiving time. The graphs correspond to a simulation with 100 concurrent users for the entire duration of the test 10 minute stress test. 125

7-11 Additional stress test statistics for ABACUS framework version two. A) using 110 concurrent clients B) using 175 concurrent clients C) using 200 concurrent clients. 126

7-12 Web server comparison using a randomized delay from 0 and 1 second between requests. A) response time B) concurrency C) transaction rate 127

7-13 Web server comparison using a randomized delay from 0 and 10 seconds between requests. A) response time B) concurrency C) transaction rate. 128

7-14 Summary of the factor increase in web server response time for the ABACUS framework version two compared to the performance of an unmodified web server. 129

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

INTEGRATING ACCESS CONTROL WITH REAL-TIME ASSESSMENT: ADAPTIVE SECURITY THROUGH THE ACQUISITION, ANALYSIS AND APPLICATION OF CONTEXT DATA

By

Hassan Rasheed

May 2009

Chair: Randy Y.C. Chow
Major: Computer Engineering

The need for adaptive security mechanisms is growing, driven by the increasing automation and modularity of attack tools, the prevalence of dynamic service-oriented architectures and the greater availability of network analysis data. In order to facilitate the evaluation and enforcement of access control policies based on real-time analysis data, a framework for the collection, analysis and dissemination of security data is proposed. In demonstrating its implementation, the framework is integrated with a web server and is used to provide a quantitative risk assessment based on data from vulnerability exploitation attempts. While maintaining high availability for non-affected entities, the percentage of denied intrusive requests is increased by triggering more restrictive permissioning in the face of escalating risk from external nodes and to system resources. A detailed performance analysis is also conducted that compares the proposed framework with an ordinary webserver and demonstrates the ability of the framework to handle high request loads in excess of one million transactions per day.

CHAPTER 1
INTRODUCTION

One of the most significant challenges in the security domain is the development of security mechanisms with the capabilities of dynamic, context aware behavior. The word context awareness means different things in various domains, but here we refer to the general ability of a software device to adjust its behavior based on its perception of the environment it operates in. This is still a very broad concept, but will be further specified in the course of the discussion.

Security mechanisms often offer assessments or evaluations of various requests and events, for example: evaluating the validity of a request for authorization or analyzing an event for intrusive characteristics. Any assessment is based on assumptions (both implicit and explicit) about the current environment: explicit assumptions may often be modeled in a policy whereas implicit assumptions are subsumed in the underlying model used for decision making. The role of context data is then to either keep those explicit assumptions accurate if they exist, or to introduce important parameters in the decision-making process if they have been left out. There are three primary motivations for the current approach: the environments in which security mechanisms are deployed are changing, the attacks themselves that must be guarded against are changing, and the emerging opportunity to leverage data sources providing valuable security data.

1.1 Motivation: Paradigm Shifts in System Security

1.1.1 Changing Nature of Attacks

There are two primary motivating sub-factors towards producing security mechanisms with an improved ability to deal with a changing environment: the changing nature of attacks, and move towards utilizing security mechanisms in service oriented architectures. The changing nature of attacks was noted in [1] and has since been confirmed in various other reports. Amongst the factors noted in the report were the following:

1. *The automation and speed of attack tools* - each of the four common phases of automated attacks (scanning, compromising, propagating and coordinated management)

are being done more quickly and effectively. Attack tools use exploits in the midst of scanning and automatically initiate attack cycles. Coordinated management is facilitated by widely used communications protocols such as instant messaging. As a result, the window of response before an attack moves on to the next stage is no longer based on the response time of a human attacker and therefore easily outpaces a human administrators ability to respond.

2. *The increasing sophistication of attack tools* - attackers increasingly use techniques to conceal the nature of the tools they use. Tools themselves are more modular and exhibit more dynamic behavior. Because of the anti-forensic behavior, previous detection methods using low-level or isolated indicators may fail to detect attacks that otherwise might be evident using multiple related pieces of evidence.

3. *Faster discovery of vulnerabilities* - the number of new vulnerabilities reported more than doubles each year, often due to examination of existing code for newly discovered vulnerability classes. This implies a wider number of available attack vectors at any given point in time. It also implies that the potential for publicizing vulnerabilities will create more occurrences of widespread exploitation of the same vulnerability. In its 2007 annual report IBM's Internet Security Systems (ISS) group reported that vulnerabilities in 2007 were down five percent compared to 2006. However, the number of those vulnerabilities that were classified as severe (high impact) rose by 28 percent [2]. High impact vulnerabilities are those that "allow immediate remote or local access, or immediate execution of code or commands with unauthorized privileges". The report also noted that of all of the vulnerabilities newly discovered in 2007, that only 50% of them are correctable with a vendor patch, meaning that more and more vulnerabilities are of a severe, uncorrectable nature.

4. *Increasingly asymmetric threat* - attacks can now be launched using large numbers of distributed systems, meaning that the traditional one-to-one relationship between victim and attacker will increasingly be a one-to-many relationship.

1.1.2 Changing Deployment Environments

The second motivating factor is the need for security tasks implemented under a service abstraction that can be used in distributed service oriented architectures. Service oriented computing has grown due to the growth of the Web and Web Services. With this growth, has come a greater need for data security. In [3] several challenges to providing security services are discussed. A number of these challenges are related to the notion of context-awareness and context data sharing. The challenges in this regard fall in three

main areas: context data acquisition, analysis and application. Among the questions raised are the following:

- Should decision point to decision point interactions for shared decision making be direct or mediated
- How will shared context information be managed
- How can we build services that can make use of past context history and get around the stateless input/output model for services
- How can inherently context-dependent services such as intrusion detection be implemented as services

All of these challenges, however, lead to the conclusion that context information and consequently mechanisms adapted to use context data are critical to the development of secure service oriented architectures. As one of the three primary system security functions noted in [4], the requirements placed on access control systems, and consequently their need to exhibit context awareness are among the highest of any security mechanism. Access control, more than perhaps any other security mechanism, is also in need of this capability as it often serves as the first line of defense against attacks and intrusions both at the application and network level.

1.1.3 Greater Emphasis on Distributed Data Analysis

As the realization has grown that there are some fundamental limitations with individual intrusion detection systems [5], interest has grown in leveraging multiple intrusion detection systems with the expectation that more data will lead to more accurate analyses. These efforts began with correlation research [6, 7, 8, 9, 10, 11, 12] which focused on relatively small sets of data contributors. It has also recently grown to include wide-area or global data analysis efforts that utilize large sets of intrusion detection systems [13, 14, 15, 16] in looking for new and emerging threats. The question of what to do with these vast amounts of data once we have them has largely gone ignored however - the primary assumption being that administrator notification is the only dependable way

to make use of such data. There is, however, an opportunity for the design of security mechanisms suited to using such data that can perform some limited tasks to improve response time.

1.2 Challenges Faced

There are challenges on two main levels to the construction of effective context-aware security systems. The first type relate to all types of systems in which context-awareness is a design goal. These are difficulties related to the acquisition, analysis and application of the context information. These problems arise from the nature of the data being collected and the sensors providing the data: the fact that they are largely autonomous, heterogeneous and distributed.

The second type of challenge are those arising from the particular application domain. These problems include the rate of incoming data, the high rate of inaccuracies with security data and the resulting uncertainty for affecting responses to intrusions. Another challenge at this level - because we are seeking to utilize context data at the access control level - is ensuring that adapting the system to rely on contextual information does not prove prohibitively slow from a performance perspective. The challenges for applying context data also include preventing and mitigating the negative effects of intrusions while maintaining a high level of service availability.

1.2.1 The Nature of Context Information

Dealing with representations of context data to make such responsiveness possible, is inherently complex. There are several inherent difficulties dealing with context information discussed in [17] including the following: the range of temporal characteristics exhibited, high degree of interrelatedness, inaccuracies or imperfections in the data and large number of alternate representations. Add to this, the fact that services to deal with context are application specific [18] and the task of managing context data becomes even more imposing. These difficulties, however, have only been confronted in the realm of security in a very limited way. Some approaches to alert correlation deal with receiving

heterogeneous, interdependent and possibly imperfect pieces of data, but the goal has yet to be making that data usable by other security components.

To further specify the challenges in this arena, we break the task of context-awareness down into three sub-tasks: acquisition, analysis and application. Context data acquisition is complicated by the fact that the sources for context data in this domain are very diverse, sometimes possessing their own specialized domain-specific standards. This is a somewhat unique challenge as the approaches to acquiring context data in a pervasive computing environment rely on data sources that are passive sensors producing relatively low-level data. More specifically, an effective approach towards context data acquisition amongst security services must deal with the inherent autonomy, heterogeneity and distribution present in current security mechanisms.

As a result of the intricacies and uniquenesses in security data, efficient context analysis techniques suited to security data must also be designed to derive key security measures from data that has significant semantic heterogeneities. These context analysis techniques must also be focused on producing *actionable data*: information that can subsequently be used to adapt policies, behaviors and enact responses to changing circumstances.

1.2.2 Applying Security Data for Improved Access Control

The notion of using security data for policy enforcement assumes a level of accuracy for the sensors that is often not present for security sensors. In particular, considering the class of intrusion detection systems, a number of issues with data inaccuracy have been noted in the research, both on a practical and theoretical level [5]. Generically, the problems are divided into two types: 1) false positives, which are incidents detected as intrusive that are in reality benign and 2) false negatives, which are incidents that are intrusive but are categorized as benign or simply missed altogether. Another issue is the sheer number of alerts that are generated by intrusion detection systems - this places

constraints on the performance of the techniques used for storage analysis to be able to keep up with the incoming stream of data.

1.3 Approach

In order to address the general issues regarding architecting context-aware systems, we develop a general framework for the collection, analysis and dissemination of security data. The acquisition approach will focus on the different architectures and mechanism that can be employed for integrating data from multiple sources. One such solution will be a tightly integrated system suited to small deployments but lacking in extensibility. The other acquisition architecture will provide greater extensibility by utilizing a service-oriented abstraction. The analysis approach will develop a set of critical analysis measures after examing the reasonable techniques for analyzing security data. The approach to application will be more domain-specific, surveying the available intrusion response controls and the techniques for activating them.

An implementation will be dicussed that focuses on addressing some of the concerns specific to the use of security data including performance and data inaccuracies. This system will also provide a platform for testing the enforcement of access control policies based on real-time analysis data. The implementation framework will include a web server as the primary access control mechanism. The analysis process will then produce entity-specific risk assessments based on data from vulnerability exploitation attempts. The data will then be applied to resolve context dependencies at the access control policy level and regulate permissions based on established risk thresholds.

1.4 Summary of Results

While maintaining high availability for non-affected entities, we are able to show an increased ratio of denied intrusive requests by triggering more restrictive permissioning in the face of escalating risk from external nodes and to system resources. We also provide performance analysis comparing the proposed framework with an ordinary webserver and

demonstrating the ability of the framework to handle high request loads in excess of one million transactions per day.

1.5 Significance and Impact

This approach demonstrates the feasibility of such adaptive security measures both in terms of effectiveness at limiting attacks and in maintaining high request throughput. The impact of the approach is primarily in demonstrating more careful, tailored response usage as a result of more detailed data analysis. Previous approaches that used data from assessment mechanisms in the performance of access control were primarily focused on integrating the performance of intrusion detection and access control in one mechanism - as a result, the data analysis was minimal and the resulting responses based on context data were applied at a system wide level. By generating context analysis data for specific sources and targets, preventative measures (such as forcing additional authentication) are applied more efficiently and only when necessary. Stronger response methods (such as locking out user accounts) can also be utilized because the scope is more granular.

Experimentation will focus on the utilization of a few key responses at the access control level: forcing additional authentication, restricting user permissions, locking user accounts and restricting access to threatened services. We will show that the resultant system is able to:

1. Use response methods more efficiently - decrease the number of requests for additional authentication that must be handled in situations of elevated threat, by elevating threat levels only for specific sources and targets

2. Limit intrusive behavior while maintaining resource availability - decrease the number of requests coming from threatening sources that are permitted (using permission restriction and account locking) while maintaining resource availability for legitimate requests.

3. Ensure greater integrity and confidentiality protection for select resources - limit risks to confidentiality and integrity by restricting access to selected services in cases where confidentiality and integrity are of greater concern than complete availability

1.6 Organization of this Report

The first section will focus on related work. In this section previous research in context information, systems integrations, the integration of security controls, the use of key assessments in access control and intrusion response will be addressed. The next three sections will each address one aspect of the general challenges discussed related to architecting context-aware systems: data acquisition, data analysis and data application. Following this will be a discussion of the system implentation: the analysis model, the architecture and the integration between the framework and the pre-existing access control system. Next a comprehensive set of testing results from experimentation with the system implementation will be presented. Concluding the sections, will be a chapter on the conclusions drawn from the research and a statement on future work.

CHAPTER 2
RELATED WORK

2.1 Context Information

2.1.1 Existing Definitions of Context

There are two definitions of context that will be considered when defining context for the purposes of this study: a linguistic definition and a definition from the pervasive computing area.

Linguistic. The American Heritage Dictionary defines context as "The part of a text or statement that surrounds a particular word or passage and determines its meaning."[19] The Merriuam Webster Dictionary defines context as, "The interrelated conditions in which something exists or occurs" [20].

Pervasive computing. Schilit and Theimer first mentioned the term *context-aware* more than a decade ago with the explanation that such systems, "[adapt] according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes in such things over time." Other definitions have taken a human user-centric view of context, defining it as: "any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application [21]. Broader definitions also exist such as, "all the knowledge that constrains problem solving at a given step without intervening in it explicitly" [22].

2.1.2 Redefining Context

There are a few problems with definitions such as those stated above. Firstly they are either are overly broad, lumping many types of information together as context: or they are overly specific, restricting context to types of information useful in the particular application being deemed *context-aware*.

Intuitively context is necessary because it clarifies meaning - but definitions that neglect this will adopt information as context that does not add to meaning. Definitions

that only consider context to be information used in a particular application will not be widely usable, and could possibly miss some information that might improve the application.

Another issue is the notion of context ownership. Most of these definitions view contextual information as belonging to an object or entity. But events - the occurrences that change the context of an object or entity also have their own context: time of occurrence, initiating entity, receiving entity, location, etc. Considering events separately from objects allows us to store temporal information directly, and to begin using the abstraction of an event history within a given domain. This would allow us to describe the semantics attached to other events in close spatial or temporal proximity to the event under examination.

Many applications in the security domain for instance typically consider patterns of behavior over time to be a primary type of context. For an application such as intrusion detection, the physical context of a system (with the exception of network topology) is actually irrelevant. In addition we would like to develop a way of describing context and context-ownership that facilitates context-sharing among entities in a given domain. This would be difficult if we consider a model where only objects possess context. If we were to develop a model based on the additional abstraction of an event (along with objects), then we could begin to describe the context of an event: its time, its actor, its subject. And context-sharing becomes as easy as providing event information to all objects in the domain where the event occurred.

Logical context definition. Context is the set of interrelated conditions surrounding an entity or event, such that when they are considered together give a full and complete 'usage' or 'applied' meaning to an entity or situation. These properties are not inherent to the entity or event and may be changed without affecting the semantics inherent to the item.

2.1.3 Context Representation

Strang et al. [23] cite the following six models for representing context: key value, markup scheme, graphical, object oriented, logic based and ontology based.

They also establish six types of requirements that a ubiquitous computing application would need from a context modeling approach. The six properties of a context model that are most appropriate are:

- distributed composition - the ability to compose context information in the absence of a central entity responsible for the
- partial validation - the capability of validating context data without all of the related data being present at the same node
- formality - the overall level of structure used to organize the context data
- applicability - the context model should fit the application it is being used in
- incompleteness and ambiguity – in the domain of a security system incompleteness can be manifested through the missing of events by an intrusion detection system
- quality of information – the concern about the quality of information delivered by a single sensor varying over time is less of a concern for well-connected intrusion detection, but the need for a context model that can accommodate varying levels of information quality between sensors is valid. Different intrusion detection systems provide different types and amounts of data on events in the system and the context model must be able to accommodate this.

Based on these requirements, the most appropriate context-modeling strategy is ontology based. This is also confirmed by Kemmerer et al. [24] who note that a common ontology is an important addition the efforts by the Internet Engineering Task Force (IETF) to establish an intrusion detection message format and protocol.

2.2 Systems Integration

Before actually addressing the issues needed to achieve an integrated security system, the question as to what constitutes integration must be addressed. In this respect, there

are two major philosophies, both of which have had a significant amount of attention in the areas of system integration and microeconomics.

The first idea is to preserve the existing systems, sometimes called legacy systems, and apply technologies to make them interoperable and achieve the desired overall system, albeit a heterogeneous one. We will refer to this as horizontal integration. A great deal of research on this topic has been done in the area of Information Systems called Systems Integration or Application Integration.

The second idea is to take the requirements fulfilled by each of the legacy systems and develop an entirely new system that fulfills the requirements of all of the old systems, but is homogeneous. We will call this vertical integration. Although the objects to be integrated in our case are somewhat different than those in the fields whose research will be cited, many of the same analyzes hold.

2.2.1 Horizontal Integration

There are three main characteristics that distinguish a horizontally integrated system [25]: 1) heterogeneity, 2) autonomy and 3) distribution. From the perspective of systems integration, all of these issues are risks which must be mitigated; in other words, they are things standing in the way of an integrated system. The mitigation process often does not change the fundamental characteristics of the constituent systems and so the same characteristics are usually present before and after integration. Systems integration views (horizontal) integration as a goal that must be facilitated, whereas economics theory views (vertical) integration as a phenomenon that occurs when certain factors are present. Thus, the determinants to be discussed later are usually described as strategies for overcoming these characteristics, but they are factors that lead to a horizontally integrated system nonetheless. Heterogeneity can manifest itself in two main areas: technical and conceptual [26]. Technical heterogeneity can come from differences in things such as: hardware platforms, operating systems, database management systems and programming languages. Conceptual heterogeneity can be produced by differing programming and data models

or differences in modeling real-world concepts. Autonomy usually occurs in the areas of design or communication and execution.

Architectures for horizontal integration. The first integration architecture is termed a component coalition. The architecture integrates independent components by providing a custom solution that will link the interfaces of the two components. These coalitions maintain the independence of the individual components in the following ways:

- each component has its own interface
- each component has independent control of its data and processing
- components may provide overlapping or conflict

The second type of integration architecture is a component federation. The main concept underlying component federations is the creation of a platform which can support a myriad of components as long as they conform to a set of standards. The federation provides infrastructure for inter-component communication and data sharing. Therefore in contrast with component coalitions, component federations are more general-purpose and more flexible.

Mechanisms for horizontal integration. There are two primary mechanism to facilitate the *persistence* aspect of data integration: coversion and a common data store. Under the conversion approach, components maintain separate data stores and data is translated to a format consumable by other components. With a common data store, however, there is a single source that accumulates data from all of the components. There are also two mechanisms commonly used to ensure *semantics* in data integration: a common schema and common data formats. The main method for achieving *control integration* is message passing. This message passing solution is actually the product of a mechanism to enable communication and a protocol to define the communication pattern.

2.2.2 Vertical Integration

One of the main characteristics that distinguishes vertical integration is the use of internal exchanges within a firm, instead of market or contractual exchanges [27]. Contractual exchanges are those in which the characteristics of the exchange between the two parties (typically price, quantity, etc) are regulated by a contractual relationship. Another characteristic of a vertical integrated process is centralized control over neighboring stages of production or distribution. An extension of this is the way in which decision making in a vertically integrated firm differs from decision making in a vertically disintegrated one.

Determinants of vertical integration. There are three main factors that produce or lead to vertical integration: technological economies, transaction economies, and market imperfections. Technological economies are situations where less of an intermediate input is necessary to produce the same downstream output, when the exchange is controlled by the same firm. This leads to vertical integration because by taking on a certain task of a production process, a given company can lessen its need for certain resources. Transaction economies are situations where costs associated with the exchange of certain inputs can be lessened by internalizing the process. It is very similar phenomenon to technological economies.

Advantages. The typically cited advantages to vertical integration that are applicable here are lower transaction costs and synchronization of supply and demand along a chain of products. The disadvantages are rigid organizational structure, and higher organizational costs of switching to other suppliers.

Manifestations of vertical integration in software systems. The theoretical points mentioned above have implications in security integration as well. We will use the characteristics of a vertically integrated business firm to establish what we mean by a vertical integration approach to security. Namely that:

1. The exchange of data between the modules responsible for different tasks is viewed as an internal operation, and utilizes a format that is standard across the entire application

2. The same programmatic entity is responsible for each phase of the security assurance process, or each security task.

2.2.3 Summary on Integration

A Vertical approach to integrated security would perform the functions of access control, intrusion detection and intrusion response in absence of interfaces and protocols between the three modules. Each of these tasks would instead be performed by modules with a shared or centralized control mechanism. A vertically integrated architecture would take advantage of low cost data exchanges between all of the modules and would also offer a higher degree of synchronization.

The approach described above as vertical integration roughly corresponds to the approach to security integration known as merged policy: the operations of access control, intrusion detection and intrusion response are all performed by a single policy evaluation mechanism, working with a single uniform policy. The absence of data exchanges based on interfaces or protocols also point to the fact that all of the operations in a merged policy solution are internal and do not require communication between different independent modules. Consequently many of the drawbacks cited for vertical integration are also apparent in the merged policy solution: primarily the rigid structure of the solution and the prohibitive cost of using a data provider outside of those packaged in the evaluation mechanism.

The drawbacks of a vertically integrated solution have serious implications for a security system. Intrusion detection began as a means to detect intrusive behavior that could not be explicitly prohibited in an access control-like specification. In addition, many current methods for intrusion detection using methods such as neural networks, or immunology models could not be satisfactorily represented in a rule-based specification. So there are theoretical as well as practical limits to a vertically integrated security system.

A Horizontal approach to integrated security would facilitate interoperability while preserving autonomy and consequently some degree of heterogeneity. Depending upon

the architecture, it might be necessary to devise and enforce 'contractual relationships' between the access control, intrusion detection and intrusion response modules so data exchange is performed in an agreed upon way. This could take the form of interfaces between each two modules in question. It would also be necessary to provide control and data integration to preserve the granularity of the system components and still provide an integrated solution.

A Horizontally integrated solution is more consistent with the characteristics and needs of a distributed system including: distribution, heterogeneity and autonomy for the involved systems. It could also enable a varied set of intrusion detection systems to interact without necessarily enforcing a particular detection method on each of the systems. Such a system would, however, have higher data transaction and processing costs.

For those reasons, therefore, the primary approach to integration will be a horizontal one. Both of the data persistence methods mentioned earlier in this chapter (conversion and a common data store) will be used for this solution. A common schema will be used to provide semantics, and a form of message passing will be used to provide some control integration.

2.3 Integration of Security Controls

We will define security integration loosely to be: the performance of a single security function or task utilizing the data or functionality from what are traditionally considered different security mechanisms. This notion of software components that perform multiple security tasks is not necessarily new, but the formality with which it is being dealt with is increasing. Franqueria [28] notes three basic strategies for 'narrowing the gap' between access control and intrusion detection: merged policy (a single component for AC and IDS using a uniform policy), correlation (both online and offline in logs) and additional information (using access control policies to model normal behavior). With the exception of the merged policy investigation, all of the research cited under correlation and

additional information are implementations of the respective technologies, not attempts to perform access control and intrusion detection in a coordinated way.

Ryutov et al. [29, 30, 31] take the merged policy approach and produce an implementation that seeks to perform access control and intrusion detection in a coordinated manner. They develop a multi-stage policy evaluation mechanism that operates on policies written in a language that has constructs for application level access control and intrusion detection. Unaddressed in this effort is a transparent method for a general access control mechanism to interface with data provided by different intrusion detection systems. In addition, although there has been some recent work on specification-based IDS [32, 33] most IDS systems still do not work on specifications, and the inability to specify to traits of attacks could be a potentially restrictive limitation.

The classification we will use for approaches to security integration will be based on the strategy used for integration, of which there are two: horizontal and vertical integration. We will define a vertically integrated security system as one where: a) the exchange of data between the modules responsible for different tasks is viewed as an internal operation, and utilizes a format that is standard across the entire application and b) the same programmatic entity is responsible for each phase of the security assurance process, or each security task. A horizontally integrated system then, would be one where: a) security tasks are performed by relatively autonomous programmatic entities and b) the exchange of information between those entities is based on standards and an architecture to mitigate the effects of distribution and heterogeneity.

Vertically integrated systems can be somewhat rigid and difficult to expand to include new components or functionality from outside systems.

In reality, all of the approaches to performing access control and intrusion that have been done thus far reflect a vertical approach to integration with the exception of [34] where the focus is using alert correlation to prevent local system resources from being used

to assist in a distributed or coordinated attack. There is still a need for an exploration of an open approach to integrating security components.

2.4 Use of Threat, Risk and Trust in Access Control

In order to utilize data from intrusion detection systems at an access control level, it is necessary to have a foundation upon which the derived context data can be related to traditional access control concepts. Solutions to this issue can be manifested at the policy level, or at the implementation level.

In [35] an extension to RBAC is developed to incorporate the notion of trust. They focus on trust based authorization and make provisions to adjust the trust given to a user dynamically based on the attributes of the user and environment as well as the past access behavior of the user.

The notion of risk is used in conjunction with threat in [36, 37]. The risk that a given request might pose to the system and the trust that should be afforded to the requesting entity are assessed simultaneously. Based on the risk of the request, a trust threshold is established which all requesting entities must meet or exceed in order for their requests to be granted.

An assessment of threat is used to make access control decisions in [38], by assigning a threat threshold which cannot be exceeded to each network resource and then maintaining a threat level for all outside nodes that is dynamically updated when they display suspicious behavior.

What is missing from the preceding efforts is an implementation that can enable access control systems to use the assessment data produced by intrusion detection systems to make decisions that are aware of system context.

2.5 Intrusion Response

In [39] one of the factors used to classify response systems is their method for selecting responses. They are divided into three classes: those that map attacks statically, those that do so dynamically based on some parameters and those that use a calculation

of the relative cost of the intrusion with the cost of the response. Static response selection matches a particular attack with a pre-determined response. Dynamic mapping systems select an appropriate response based on attack metrics. At design time, each attack is associated with a set of responses and then in real time one of the responses is chosen based on the characteristics of the attack. Cost-sensitive response selection determines the best response based on several cost and risk factors. These values may include monetary values, probabilistic measurement or other objective metrics. They may also include relative measurements of organizational security and risk factors.

CHAPTER 3
GENERAL APPROACH PART 1: CONTEXT ACQUISITION

3.1 Introduction and Design Goals

The first necessary stage in architecting context-aware behavior is to gather the data which will influence the behavior of the mechanism. Because the data in question is coming from largely autonomous assessment mechanisms with high degrees of heterogeneity, the acquisition of context data is partially an integration problem. Systems integration approaches typically discuss the architecture, data integration and control integration strategies used to reduce heterogeneity while preserving the autonomy of the involved systems. The other facet of the context acquisition method is the means for discovering relevant context information once heterogeneity has been reduced. Design goals for the approach for context acquisition include:

- Dynamic Context Discovery - One of the primary characteristics of the approach for context acquisition will be to provide support for what we will call dynamic context framing. Using the notion that the context of an event consists of other, related events we then establish a criteria for relatedness that is appropriate for each individual security mechanism and then frame the context of an event based on those two factors. This context acquisition strategy must allow security components to select and receive *only* that data that is relevant to the decision they are trying to make. Because security mechanisms deal with events, they should be able to select the other events that relate to the event under consideration without necessarily having to process and deal with every event that occurs in the domain. As noted in [40] this property is not so much a desired trait as a required one as the volume of events processed solely by intrusion detection systems can reach tens of thousands per day. This implies also that the strategy for context acquisition must be able to search for events based on characteristics of relevance. So the first required property of the context acquisition approach is that it must provide relevant data.

- Implementation Transparency - Another goal of our approach with regards to acquisition of context data is to allow security mechanisms to acquire data from other security controls while remaining agnostic of their implementation details: that a security component can acquire context data merely by knowing the features of the data it would like to receive. In this case that will entail the features of the event that is being evaluated and the domains from which the data should be gathered.
- Provider and Consumer Decoupling - Another necessary feature is that the provider and consumer should be decoupled in time and space. We would like to provide functionality where an event provider can register or publish event information and then consumers can access that data according to their own constraints around what constitutes relevant context data. This also implies that the accesses of the provider are to be asynchronous, while those of the consumers will be synchronous. Decoupling in space is also necessary to support distribution.
- Allowing Policy Level Description of Relevant Context - Before we can analyze context data, or even search for it, we must have a means to describe its features and characteristics. One primary way of achieving this is through policy-specifications that include the features of context data.

3.2 Survey of Context Acquisition Approaches

In this section we outline two approaches for architecting a system capable of providing on-demand context data and then discuss specific issues relating to how access control and intrusion detection can be brought closer together. Both of the approaches are within the classification of horizontal integration approaches as discussed previously. They differ primarily in whether or not data exchanges are carried out with the use of an external third party. The first architectural approach is a closed one, based around a component coalition. This approach is appropriate for environments where the participants in the architecture are few and will not need to be extended. It relies on point-to-point integration between the interfaces of each participating mechanism. The second approach

we will discuss is an open approach, based around a federation. This approach focuses on providing what amounts to a middle-ware that facilitates communication between the different mechanisms with a high level of transparency, but a corresponding increase in required development effort.

3.2.1 Closed/Coalition Approach

A component coalition integrates independent components by providing a custom solution that will link the interfaces of the two components. These coalitions maintain the independence of the individual components in the following ways:

- each component has its own interface
- each component has independent control of its data and processing

Because each component maintains control over its own data and processing this architecture allows for the desired transparency of intrusion detection method. In addition, this architecture allows the focus to remain on developing effective interfaces for each of the components.

The architecture of the coalition is pictured in Figure 3-1. The movement of data in this architecture is achieved through a pull mechanism on the part of the event consumers. This is necessary, because only select events are actually of interest to the consumers, and the actual attributes of those events are dynamically determined. Each component is pictured as both a provider and consumer, but any one component can serve as a provider or consumer, both or neither.

The diagram in Figure 3-2 describes the sub-parts of each security component. The Core Decision Mechanism (CDM) is the portion of the component that is responsible for fulfilling the primary tasks and responsibilities designated for that component. The flow of data from the Event Consumer (EC) to the CDM is based on a pull request from the CDM. This is based on the assumption that not every context event is relevant to the tasks that the CDM is trying to perform. Thus, as-needed, the CDM can request context data using the context discovery policy that the EC contains. The flow of data from the

CDM to the Event Provider (EP) is a push mechanism for much the same reason. It is assumed that not every event processed by the CDM will be useful to publish as context data, and so the CDM can publish select events at its discretion.

Data integration: data store and common schema. In this system, it will be necessary to provide both data semantics and data persistence. There are two primary strategies for achieving data persistence. The first is conversion where each component maintains its own data store and data is translated into formats that are consumable by other components. The second mechanism for data persistence is a common data store. This data store is merely a single source that accumulates information from all of the components. Both of these strategies will be used to provide the necessary transparency. A common data store is necessary to collect data from all of the intrusion detection systems in a single format. The strategy for clustering and linking alerts depends on a standard for the representation of intrusion detection alerts. These clustering and linking algorithms use conversion operate on intrusion detection data while still returning a value consistent with the access control schema.

The mechanism used for data semantics is a common schema. This common schema is generated differently, however, than the merged policy schema. In the merged policy approach two or more schemas are merged to produce a new schema and the originals are discarded. The approach being used here, however, is to augment one schema (e.g. the access control schema) with essential elements from another schema (intrusion detection), but preserve that other schema for representing data in that domain and convert between the two as necessary. The augmented schema serves as a common point of reference for the two separate domains.

Control integration: message passing. The mechanism for control integration in the coalition based approach is message passing. The alerts from IDS systems to the central data store. Message passing will also be the means through which access requests and responses are sent to and from the decision point.

3.2.2 Open/Federation Approach

Architecture: context management services. The desired functionality from an architectural point of view is to provide services that can aggregate context information from multiple providers, and then subsequently disseminate that context information back to consumers on-demand. There are a number of distributed system architectures which could be used in this scenario, however, Publisher/Subscriber (pub-sub) systems are best suited to the problem of a context data-sharing framework for a number of reasons. First is the fact that all pub-sub systems support decoupling of the producer and consumer in time and space, which is an essential design requirement for this system. In addition there is existing work on content based, and even ontology based pub-sub systems [41] that can be built upon for allowing the selection of appropriate context data which will be consistent with the chosen approach for modeling context data.

Service Oriented Architectures also provide additional formalisms on top of a basic publish-subscribe architecture to facilitate registration and lookup of service (or in this case context) providers that will also be necessary. The basic model used will consist of a network of context providers and context aggregators. The aggregators are hierarchically structured and subsequently feed context consumers. Any security component in the network can serve as either a context provider, consumer, or both. The security components will be primary context providers and secondary context consumers. The meditative services will primary context consumers and secondary context providers.

In Figure 3-3, security components are again pictured containing event consumers. In this instance, however, the event provider is a common service. Instead of requesting events from the different providers based on knowing which provider has which events, the events can be acquired from one service, merely by knowing the features of the desired events.

In Figure 3-4, the anatomy of a security component under the open architecture is shown in greater detail. The core decision mechanism now pushes events to the common event provider service which is external to the component.

Data integration: low-level context modeling. This approach is based on a different concept of context than is commonly used. In pervasive computing research, the primary concern is having applications respond to changes in the physical environment generated by users and other physical objects. In security, however, the primary concern is events in a virtual environment where most of the objects are passive data elements. Security controls such as access control and intrusion detection function as event classifiers. As a result, the focus shifts from decomposing an event into the state changes that it produces in physical objects to maintaining the event itself as the primary object of concern which must examined. Consequently, a more appropriate definition of context would be the events related to a given event that provide additional information about the circumstances under which that event occurred. In this way, the context of an event consists of other, related events.

In [23] six methods for modeling context data are cited and the ontology based modeling technique is cited as the one most capable of providing the features used to evaluate all of the modeling methods. The features most relevant to the discussion at hand are: distributed composition, partial validation, handling incompleteness and ambiguity, sufficient formality and applicability to existing environments. This survey, in addition to the frequent use of ontologies in service oriented architectures indicate that an ontology based context model would be the most suitable to achieve the aims of this project.

As for actual construction of the ontology, it is noted in [42] there are three primary methods for using ontologies to facilitate integration. Those methods are the following: establishment of a single global ontology, use of multiple ontologies (each for a specific sub domain), or a hybrid approach that allows for multiple specific ontologies, but bases them all on a shared vocabulary to facilitate interoperability.

The fact that there are some existing standards for representing security event data [43, 44] and the need to make the context model extendable advocate in favor of a hybrid approach. The major task then, for this context model is two-fold: 1) the content of the shared vocabulary, and 2) the transformation of existing event representations into a form that is compatible with the shared vocabulary. The shared vocabulary will consist of two primary elements: 1) a set of attributes common to two or more security domains that can be built upon to establish specific domain ontologies and 2) a set of predicates that express certain relationships, both semantic and syntactic, between two or more events.

Control integration: ontology-based event correlation. The preceding two research focuses (context modeling and context management services) have been necessary to provide a framework and infrastructure support for the first pillar of the approach, namely: context acquisition. The actual process of context acquisition will rely on correlating events using the attributes in the shared vocabulary and the predicates for expressing relatedness between events that will also form part of the context model.

3.3 A Coalition-Based System for Context Acquisition

3.3.1 Information Model

3.3.1.1 Access control

Defining access control. Sandhu and Samarati [45, 46] define access control as a family of strategies for one party to precisely control what other parties will be allowed to do with resources that it controls. They restrict it, however, to limiting and controlling the actions that authorized users of a system are allowed to perform. They also note that true information security is the product of access control in conjunction with authentication and auditing. They highlight what are perhaps the three most prominent access control policies:

- Discretionary Access Control (DAC): information access is governed by rules that state for each user and for each data item which access modes the user is allowed on that object. DAC is very flexible which makes it adaptable to a variety of

environments, but it does not provide true guarantees on the flow of information in a system.

- Mandatory Access Control (MAC): also know as lattice-based. Security levels are associated with each user and data object in a system. Users can only read down (read items of a lower security level), and may only write up (write to data objects whose security level is greater than their own). Prevents information from higher security areas flowing to lower security ones.
- Role Based Access Control (RBAC): a role is a set of actions and responsibilities associated with a certain work activity. Access authorizations are then specified for roles, and individual users adopt roles as needed with certain restrictions.

Tasks and responsibilities in integrated security. In an integrated security architecture, access control policies should take advantage of state information from other parts of the architecture so that policies can enforce access limits with a high level of granularity and specificity. In addition, these dependencies should be allowed to change dynamically and autonomously. Access Control also takes on the additional role as the point of inflection and the place where changes are reflected.

Data provided by access control. The access control mechanism has access to the policies governing each resource and therefore can provide information on how each resource is being controlled. In addition, in most systems an access control mechanism will intervene on every resource request that passes through the system and thus access control can provide detailed information on a request-by-request basis. Such information would include the following: abnormalities in access requests, operation requests and failures, resource usage, login/logout information and exception conditions.

Information needed by access control. To achieve the goal of access control being based on the overall context of the system, it is necessary for the access control system to resolve dependencies in access policies. These dependencies could include information about any attacks that have taken place, or changes in the policy to help the

system recover from and prevent future attacks. An outline of the information needs for access control is the following:

- Vulnerability of each available system resource
- State of compromise of each available system resource
- Compromised/Misused user accounts
- Attack description: user, operation, resource, means
- Policy Modifications/Updates
- Aggregate system data for policy evaluation

3.3.1.2 Intrusion detection

Intrusion detection traditional definition. A widely accepted definition of an intrusion is, “any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [47]. Sandhu and Samarati mention intrusion detection as a method for ensuring audit controls and divide intrusion detection systems based on reactivity into passive (used to analyze audit data and report anomalous behavior) and active detection (analyze audit data in real time and may respond to protect systems). Other authors classify intrusion detection strategies based on how they detect an intrusion: either detecting misuse, which relies on recognizing well-known attacks, or detecting anomalies, which merely tries to establish a deviation from normal user behavior. Misuse detection systems typically rely on matching events with known attack scenarios or signatures. Anomaly detection systems establish models for normal behavior through techniques from statistics, artificial intelligence or other fields.

Tasks and responsibilities in integrated security. The notion of an intrusion detection system that uses audit records is inherently dependent upon at least one other security mechanism: namely access control. Introducing the notion that information will flow back ’upstream’ from intrusion detection to access control, however gives rise to new opportunities for the intrusion detection system.

Information provided by intrusion detection. In general, an intrusion detection system should provide the following information regarding a perceived intrusion: certainty of analysis, attack impact, and attack characteristics.

Information needed by intrusion detection. The information that the intrusion detection system uses to perform its tasks mainly consists of information on specific access requests, such as the following: the user performing the action, the action being performed, the resource being used and any exception conditions generated.

3.3.1.3 Intrusion response

Intrusion response traditional definition. The main function of an intrusion response system is to take steps to both prevent future attacks and mitigate the damage from current attacks based on characteristics of the attack and the type of the system resource. The Fisch Taxonomy [48] provides the following classification of the types of intrusion response: active damage control , passive damage control, damage assessment and damage recovery.

Tasks and responsibilities in integrated security. In addition to traditional response techniques, the integration of an intrusion response system with access control provides the opportunity to make access control policy modifications.

Information provided by intrusion response. An intrusion response system will have various response scenarios for different attack types. It will be able to provide information about how to mitigate attack vulnerabilities, and the measures necessary to counterattack different attack types. Including policy changes for the following: intrusion mitigation or recovery and intrusion prevention.

Information needed by intrusion response. The Carver Intrusion Response Taxonomy [49] classifies intrusion responses in six dimensions. Each of these dimensions is, in reality, an input to the intrusion response mechanism that determines the type of response. The taxonomy consists of the following elements: timing, type of attack, type of attacker, attack implications, strength of suspicion and environmental constraints. The

attack implications dimension refers to how critical the resource is to the system as a whole. The environmental constraints are any limits on the type of response that can be taken whether they be legal, technical or otherwise. The strength of suspicion is a measure of certainty the IDS system has in the intrusiveness of the event under question. If the goal is to integrate intrusion response directly with access control, the response should be based on the policy used to control access to the resource: the underlying model, the policy target and the operations permitted by the policy.

3.3.1.4 Model overview

This information model (see Figure 3-5) is based on the main abstraction of a Security Event. The three main subclasses of events are Access Request, Intrusion Attempt and Intrusion Response which will each be discussed separately below.

This model relates the conditions for an access request to the data from an intrusion analysis by placing the latter as attributes of a subclass. Intrusion attempts are linked to intrusion responses by including one or more of them as a property of each response. There are four main classes: *Security Event*, *Access Request*, *Intrusion Attempt* and *Intrusion Response*.

3.3.1.5 Security event

- Attributes:

1. Source: the initiator of the event. Possible sources are: Node, User, Process or Service

   (a) Target: the intended object of the event. Possible targets are: Node, User, Process, Service or File

- Subclasses:

   1. Access Request: an attempt by some source to perform and operation on a target.

   2. Intrusion Response: the response to an intrusion attempt.

3.3.1.6 Access request

- Attributes:

  1. Evaluation Result: the result of the evaluation of the set of preconditions applicable to this request. Possible results are: Permit or Deny

  2. Requirements: set of conditions that must be satisfied for the request to be granted. Preconditions can be specified for the Environment, Subject, or Resource

  3. Consequences: specification of actions that should be taken on request completion.

- Subclass:

  1. Intrusion Attempt: An access request that is deemed intrusive.

3.3.1.7 Intrusion attempt

- Attributes:

  1. Means: what was used to perpetrate the attack. Could be in one of three subcategories: Bypassing Control, Passive Misuse or Active Misuse

  2. Assessment: the systems judgment of the event on: its Impact in the system, the Confidence of the analysis and the Classification of the attempt

  3. Results: the net effect of the intrusion attempt. This could fall into one of three categories: Denial of Service, Exposure or Erroneous Output

3.3.1.8 Intrusion response

- Attributes:

  1. Constraints: any system factors that limit the response that can be taken against the intrusion attempt

  2. Intrusion Attempt: one or more intrusion attempts that are being responded to

- Subclasses:

  1. Response During the Attack: could be either Active or Passive

  2. Response After the Attack: could with the aim of Assessment or Recovery

3.3.2 Implementation

We have implemented a system based on the proposed information model. This implementation consists of two primary systems: an access control system and an intrusion detection system. In this initial implementation we make use of the closed/coalition approach with reactive data analysis to produce threat information.

Our access control system is based around the eXtensible Access Control Modeling Language (XACML) [44]. XACML specifies a language for access control policies as well as access requests and responses, all in XML. The language supports policies that are composed of the following elements:

- A target: the set of resources, subjects, actions and environments to which the policy will apply
- A rule-combining algorithm: a specification of how different rules should be composed
- A set of rules: each rule is a combination of a target, effect and a condition
- obligations: operations that are performed by the parent of the current policy container when a specified authorization decision is returned

The effect portion of the rule indicates whether the rule is a negative or positive access right. The condition element of a rule offers the possibility of further narrowing the applicability of a rule, by specifying some things that must be true for the rule to come into effect.

The motivation behind this language is to provide a common policy language for enterprises to ease the difficulties that come with the current heterogeneity in policy representations for different security domains. Due to its descriptiveness and expansive range some of the elements of the XACML data model have been included in the proposed information model.

The XACML specification also includes abstractions for an access control architecture. The primary elements of this architecture that we will discuss are:

- Policy Decision Point (PDP) - evaluates policies and returns a decision on the request.
- Policy Enforcement Point (PEP) - responsible for making requests to the PDP based on access requests and actually enforcing the decisions returned by the PDP

The information flow in this system is detailed in Figure 3-6. It includes the following steps:

1. An initial access request is made for a resource under the control of the web server. In this way the web server fulfills the role of a Policy Enforcement Point (PEP) discussed in the XACML architecture description.
2. The web server initiates a request to the Access Control System (the Policy Decision Point) indicating the resource being requested and the source of the request
3. The access control system finds the policy that governs that resource and extracts any rules indicating the request response should be based on threat information, and what are the dimensions of the threat profile (source, target or both)
4. Based on the dimensions of the threat profile and the features of the actual request, the access control system establishes criteria for a set of related events
5. All of the related events are selected from the IDS data store
6. The calculation method for the threat profile is applied across the set of selected events and the value is checked against the threshold from the policy
7. A request response is returned to the web server which then regulates access to the resource

In our implementation system, the PEP is a basic web server, modified to pass access requests first through an XACML PDP written with the Sun Microsystems XACML API. The PDP functions as a module on the web server, but could be implemented as a standalone service.

All of the intrusion detection data is generated through an instance of Snort, configured to report alerts in IDMEF [50] format. Alerts are generated by the IDS and sent to the service that receives the alerts and stores them in an XML database. For each

access request, the database is then queried for alerts based on the policy that governs the resource being accessed.

3.3.3 Summary of the Coalition Approach to Context Acquisition

We have modeled and implemented a system that uses a coalition approach or one-to-one mapping to integrate an access control and intrusion detection. The benefits of this approach are its relative simplicity and lack of reliance on infrastructure support. Because there is less overhead in selecting relevant context data, this approach is also more efficient. Overall the coalition approach described would only be appropriate for access control systems embedded into networked applications that have strict time requirements and do not need a full range of assessment data to select from. The main strength of the federated approach is the ability to use the same interface and access context data from a variety of assessment mechanisms: if this functionality is not required, then the coalition approach could be sufficient. The implementation discussed served to elucidate some of the critical limitations of pursuing the coalition approach to context acquisition.

While the coalition approach is easy to implement and provides efficiency because of its limited scope, it still fails to fully provide all of the design objectives that were identified in Section 3.1. The federated approach, in contrast, provides for things like provider and consumer decoupling, implementation transparency and dynamic context discovery in a more complete way due to the use of secondary context aggregation services. For that reason, the federated approach to context data acquisition will next be examined.

3.4 A Federated System for Context Acquisition

3.4.1 Approach

The approach for addressing the design of federated systems will be different than the approach used for coalition systems. Because coalition systems rely on one-to-one interactions between the different components, it is necessary to eliminate the heterogenity or provide a common schema that allows interoperability between each pair of systems. In the federated approach, the focus shifts to maintaing the autonomy and existing schemas

but mitigating those factors with secondary mechanisms. We will employ a strategy of event correlation as the primary means for aggregating contextual information across multiple domains.

3.4.2 The Use of Event Correlation in Federated System

In Section 3.2.2, the use of ontology-based event correlation for control integration was discussed. Event correlation, however, is a broad field encompassing many different approaches. In order to select the appropriate approaches for correlation we will first survey what has been done in this regard and then evaluate those methods in light of the requirements that must be satisfied to facilitate event correlation across multiple domains. In addition, the construction of the ontology must be dealt with in some depth. Various approaches will be explored for constructing an ontology aimed at integrating various data sources. Based on the design requirements previously discussed, the best approach will be selected and an ontology for correlation will be proposed. Finally, the design and implementation of a context aggregation service incorporating ontology-based correlation will be discussed.

3.4.3 Available Correlation Methods: A Taxonomy of Event Correlation Approaches for Security Data

We next survey relevant research in the area of alert correlation, offer two taxonomies of event correlation approaches used specifically with IDS data. The first taxonomy is based on the goal of the correlation process and the second classifies approaches based on the type of relationship that is established between the alerts in question.

3.4.3.1 Taxonomy of alert correlation methods based on outcome

To summarize the preceding survey, we offer the first of two taxonomies. There are three primary goals for event correlation seen in the existing literature: 1) alert reduction 2) alert association and 3) alert analysis.

Correlation methods that achieve alert reduction are typically referred to as fusion or merging. In reality, most of the merging techniques do not merge alerts in the sense of

replacing two distinct alerts with a new one that somehow addresses them both. Instead, of the approaches that perform merging, such as [51, 52, 53] utilize the notion of a meta-alert which represents the information present in several distinct alert through lists for discrete values such as IP addresses and ports, or ranges for continuous values such as time. The effect of creating a meta-alert achieves alert reduction because, ostensibly, the administrator can inspect a meta-alert instead of the multitude of alerts that it is constituted by. In this way, the reduction of alerts is only from the perspective of the administrator.

The next division of correlation methods contains techniques that detect or assert an association between two or more alerts. This class encompasses the following approaches: aggregation [9, 52], clustering [51, 53, 54, 55, 56], multi-step attack detection [8, 52, 57], session reconstruction [52] and detection by chronicles [58]. All of these techniques have been grouped together because in the end they all express a relationship between the alerts that are the subject of the approach. In the case of clustering, the relationship is a similarity based on perceived distance. For aggregation, the association is a set of one or more shared attributes (in that sense it is a stricter form of clustering). Multi-step attack detection associates alerts that are proper subsets of the same set. A high-level attack sequence is viewed as an ordered set of distinct actions that achieve a certain observable effect on the system. The individual alerts are therefore associated to each other through membership in a specific high level attack-sequence. The chronicles approach is similar to the multi-step detection approach with the addition of temporal constraints between various states.

The correlation techniques in the alert analysis class are impact analysis and prioritization [52]. Both of these approaches make a determination about an individual alert using only its features and data about the overall context of the system. In the case of impact analysis the context information is an asset database and whether or not the attack

succeeded. For prioritization the context information is a weighting assigned to the class of the attack as well as the asset database also used for impact analysis.

3.4.3.2 Taxonomy of alert correlation methods based on means

The taxonomy for alert correlation methods based on means includes the following techniques: 1) attribute congruence 2) attribute similarity 3) membership in common superset and 4) ontological relationship. The aim of this taxonomy is to distinguish between correlation approaches based on how they are conducted.

In the first category of attribute congruence, there are two subcategories: syntactic and semantic. Aggregation is the only correlation approach that relies on complete syntactic attribute congruence to associate one or more alerts. This is as a result of establishing criteria for membership in an output set that certain attribute values a_1, a_2 and a_3 must be equal to values a_1, a_2, a_3 respectively (in the case where all three values are used). The most thorough examination of aggregation is found in [9] but it is also discussed under the name of in [52], where it is divided into two distinct tasks. Attack thread reconstruction is aggregation of alerts with the same source and target, while attack focus recognition is aggregation of alerts with either the same source, or the same target.

The session reconstruction approach in [52] relies on what will be referred to as semantic attribute congruence. Session reconstruction links two alerts, issued at different deployment levels, that refer to different targets syntactically (a port number and service name), but are in reality describing the same system entity.

Under the category of attribute similarity we place the clustering [51, 53, 54, 55, 56]. Implicitly, merging is also placed in this category, because it is almost always preceded by clustering. A cluster of alerts is the product of using a set of expert-designed similarity measures to determine which alerts are most like one another. Within the common membership class are multi-step detection [8, 52, 57] and the chronicles approach [58],

both of which link alerts based on the fact that they are associated with the same high-level attack description.

Under ontological methods, the first is the pre/post-condition approach. This approach is most thoroughly discussed in [8, 57] and relies on a specification of pre-conditions and post-conditions for each attack. When alerts arrive indicating an attack it is compared with other available alerts to find matches between its conditions and the conditions of other alerts. The second approach is the cause and effect or triggering approach used in [56]. Although the pre/post condition approach is also a part of this method (specifically the one used in [8]), it is somewhat distinct because the complete functionality relies on a specification of events and the alerts that they trigger.

The full taxonomy is presented in Figure 3-7.

- Attacks: relationships between distinct attacks (we will assume that certain attacks can be identified by the presence of a few specific properties such as classification, action, etc.)
  - Ontological Relationship: those relationships that identify a specific logical relationships between two attacks
    - Pre/Post Condition:
    - Cause/Effect: one attack is identified as the trigger or cause for another
  - Shared Membership
    - Attack Classification: both attacks are in the same class. A class may be identified by specific values certain fields
    - High Level Attack Scenario: both attacks are members of a high-level attack scenario made up of multiple events
- Alert Attributes: relationships that can be established between individual fields in an alert
  - Congruence

        * Semantic: the two attributes are related through a third element that indicates that they refer to the same entity
        * Syntactic: the value for the specified attribute is the same in both alerts
    - Similarity
        * Positive (Proximity): based on a pre-designed distance measure, the attributes for the two alerts exceed a threshold for the maximum separation
        * Negative (Separation): based on a pre-designed distance measure, the attributes for the two alerts exceed a threshold for minimum separation
        * Covariance: specific sets of attributes vary in the same way from one alert to the other

3.4.4 Selecting an Effective Ontology Approach

In [42] three methods are discussed for developing ontologies whose purpose is content explication in an integrated system: a global shared ontology, multiple isolated ontologies and a hybrid approach. In the first approach, one global ontology is devised to provide a shared semantic vocabulary across all of the information sources. This approach is most suited to situations where all of the information sources share a similar view of the domain – if event one information source has a slightly different view it can become difficult to produce an effective global ontology. The second approach of constructing a separate ontology for each information source has the advantage that it supports evolution of the information source, and the addition and removal of information source. Under this approach, however, in order to compare ontologies, it is necessary to define an inter-ontology mapping. Inter-ontology mappings, however can be difficult to define in practice, not to mention the fact that as the number of information sources expands the number of mappings that are necessary grows exponentially. The hybrid approach allows for the semantics of each information source to be described by its own database, but with the requirement that the individual ontologies are built from a global shared vocabulary. The shared vocabulary contains the basic terms that are combined in local ontologies to

produce more complex semantics. The hybrid approach supports addition and evolution of ontologies, but has the drawback that existing ontologies must be rebuilt from scratch.

Comparison of ontology approaches. The global ontology option is not suited to providing inter-domain event correlation, because different views of the domain do exist, even if only considering access control and intrusion detection. While the task of producing a global ontology for access control and intrusion detection might not be that infeasible, there are a few additional goals present in this case which make this option inappropriate. One such goal is to provide a semantically explicit description of the elements in current data models to facilitate adoption and interoperability. Another is to preserve a degree of modularity between the systems. The approach of producing multiple, isolated ontologies also does not meet the design requirements in this case, because the primary goal is to compare events based on these ontologies. And the need to provide pair wise mappings between every set of ontologies would makes such a system difficult to produce. The hybrid approach best meets the requirements in this situation. It can allow a particular security mechanism to discover events from a variety of potential data sources, only using the information contained in the base vocabulary. It can also allow the ontologies for the individual domains to evolve individually.

3.4.5 A Hybrid Event Ontology

3.4.5.1 The basis for a common ontology

In order to produce a base vocabulary we will use the IDMEF and XACML schemas, both of which provide representations for events, although in different domains. The goal of this process is to extract enough common elements from these two data models to form the basis of a shared vocabulary for cross-domain event correlation.

A survey of the XACML event schemas. The XACML schema provides two event representations: access control requests from a source, and the response from the policy enforcement mechanism. The XACML request context schema contains the following elements: 1) Subject – information about the subject of the request 2) Resource

– the resource or resources for which access is being requested 3) Action – attributes about the action being requested 4) Environment – attributes of the environment in which the request is occurring. The XACML response context schema includes the following elements: 1) Decision 2) Status 3) Obligations.

The IDMEF event schema. A message in the Alert Class contains the following relevant classes: 1) Classification - The name of the alert 2) Source - The source of the event described in the alert 3) Target - The target of the event described in the alert 4) Assessment - Information about the impact of the event, actions taken by the analyzer in response to it, and the analyzer's confidence in its evaluation 5) Additional Data - Information included by the analyzer that does not fit into the data model.

3.4.5.2 The proposed ontology

The ontology being proposed (see Figure 3-8) consists of three main parts: a base vocabulary, an access control domain ontology and an intrusion detection assessment ontology.

The base security event vocabulary. The common elements to these two event schemas are the access source, target and the action being performed. Combining all of the preceding discussion regarding existing data models and the taxonomy of system actions, we therefore offer a base vocabulary for inter-domain event correlation of assessment data in access control systems based on the following primary elements:

- *SystemEntity* – the class containing all valid system entities. The Subclasses of SystemEntity are Node, Process, User, Service and File. The first four subclasses are valid domains for the EventSource property, and the latter four subclasses are valid domains for the EventTarget property.
- *SecurityEvent* – a generic type of event possessing the properties of an event source, event target and event action. The subclasses of SecurityEvent needed for detailed data description are *Access Request*, *Access Response* and *Assessment*.

- *Action* – the class containing all of the distinct system actions. Subclasses needed for detailed data description are ClientAction and ServerAction.

Access control domain ontology. In order to provide a more detailed description of information at the access control level utilizing the base vocabulary, we develop two subclasses of the Security Event Class (*AccessRequest* and *AccessResponse*) and two subclasses of the SystemAction class (*ClientAction* and *ServerAction*). The class AccessRequest has the properties *hasSource, hasTarget* and *requestAction.* The first two properties range over SystemEntity and are inherited from the parent SecurityEvent classes. *requestAction* ranges over the class of ClientAction. The Access Response has the *respondsTo* property which references an Access Request, a *responseString* dataType property that contains the actual request response (permit or deny) and a *hasResponseAction* has a range of the set of ServerAction actions. In the case of the AccessResponse, the hasSource and hasTarget properties are still present, but the relationship between client and server has been flipped: the source of the AccessResponse is the server that was initially contacted with the access request.

There are two subclasses of the Action class generated for access control: *ClientAction* and *ServerAction.* ClientAction are the desired manipulations of system resources that are specified in Access Requests. They are subdivided into Malicious and Non-Malicious. Non-Malicious actions include *CommandExecution*, *DataAccess* and *DataAlteration.* Subclasses of ServerAction actions are *RequestDecision* and *EvaluationObligation.* RequestDecisions are the basic permit/deny decision issued for each request: EvaluationObligation refers to the obligations mentioned in the XACML schema [59], which constitute actions performed by the policy decision point as a result of evaluating an access request.

Intrusion assessment domain ontology. Assessment is the class of analyses regarding events or entities. Each assessment has an *assessmentSource* property whose range is an AssessmentSensor. Each *Assessment Sensor* has a confidenceRating that affects the way its analyses are viewed; in this case the confidence rating is given by the

access control system utilizing events from that sensor. The primary subclass included in the event ontology is the *IntrusionAssessment* class. Other assessment types will be discussed under the topic of context analysis.

Each IntrusionAssessment has the following dataType properties: *impactSeverity*, *impactType* and *attackCompletion* (indicating whether the attack completed successfully or not). Assessments also have the following object properties: *describedBy, triggeredBy*, *hasIntrusiveAction, hasSource* and *hasTarget*. The *describedBy* property has the range of a VulnerabilityDescription and denotes that the vulnerability whose exploitation was detected is described in the referenced description. Each VulnerabilityDescription has a *referenceID*, an *originDB* and a *referenceURL*. The *triggeredBy* property has a range of the set of AccessRequests and denotes the access request to which the assessment applies. The properties hasSource and hasTarget both refer to SystemEntites that are the source and target of the intrusive event, respectively and are inherited from the parent SecurityEvent class. The *hasIntrusiveAction* property has a range of the class of *Malicious* client actions, which gives a general description of the kind of attack being perpetrated. The following subclasses of malicious client action are included in the ontology and are based on the taxonomy from [60]:

- *Probing* – all activities related to gathering data about a system. Subdivided into probing of users, services and nodes.
- *Denial of Service* – hindering legitimate access to the system. Subdivided into *Temporary*, *Administrative* and *Permanent*. A temporary denial of service is one that will be automatically recovered from. An administrative denial of service is one that will require administrator intervention for recovery. Permanent denial of service attacks are those whose effects are indefinite.
- *Interception/ Reading Data* – subdivided into interception of files or of network traffic.

- *Alteration/ Creation of Data* – subdivided into modifying system data or modifying intrusion traces such as log files

3.5 Summary

Two contrasting approaches were presented for facilitating the process of acquiring context data and achieving the integration necessary to overcome component autonomy, heterogeneity and distribution. A coalition architecture was presented where each security component must establish an interface to provide data to other components serving as consumers. Under the coalition architecture, each mechanism is also responsible for explicitly performing the one-to-one interaction with each component that it wants to obtain information from. A federation architecture was also presented that uses a common context aggregation service to facilitate the dissemination of context data. The federation also has the added benefit that the architecture is overall more extensible in that each component of the architecture only needs to be updated to pull new information from the common data store rather than interacting with a new component with a separate interface. Two taxonomies of current methods for intrusion detection alert correlation are provided: one based on the means used to correlate the alerts and another based on the outcome of the correlation. After examining approaches for integrating data coming from heterogenous domains, a hybrid ontology is proposed for the purpose of allowing related events in different domains to be aggregated. The ontology uses a base vocabulary, synthesized from common elements in existing schemas for security events and allows extension with domain specific classes and attributes. This ontology will be translated into a relational database model that serves as the schema for the event database discussed in Section 6.5. Also, based on the conclusion that the federated approach more completely fulfills the needed design goals, the implementation will follow a federated model as the architecture for the system.

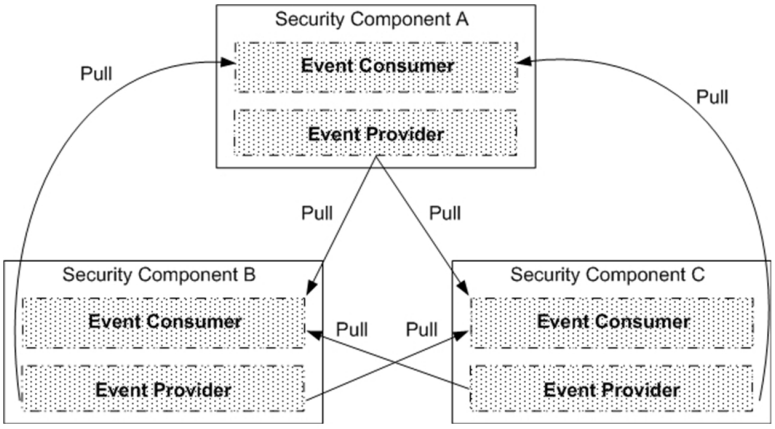


Figure 3-1. Diagram of a security coalition. Each security component has to interact with all of the other components in order to access their data. This architecture is limited in extensibility because each time a new member is added to the coalition, all of the other members must be adapted to use its interface.

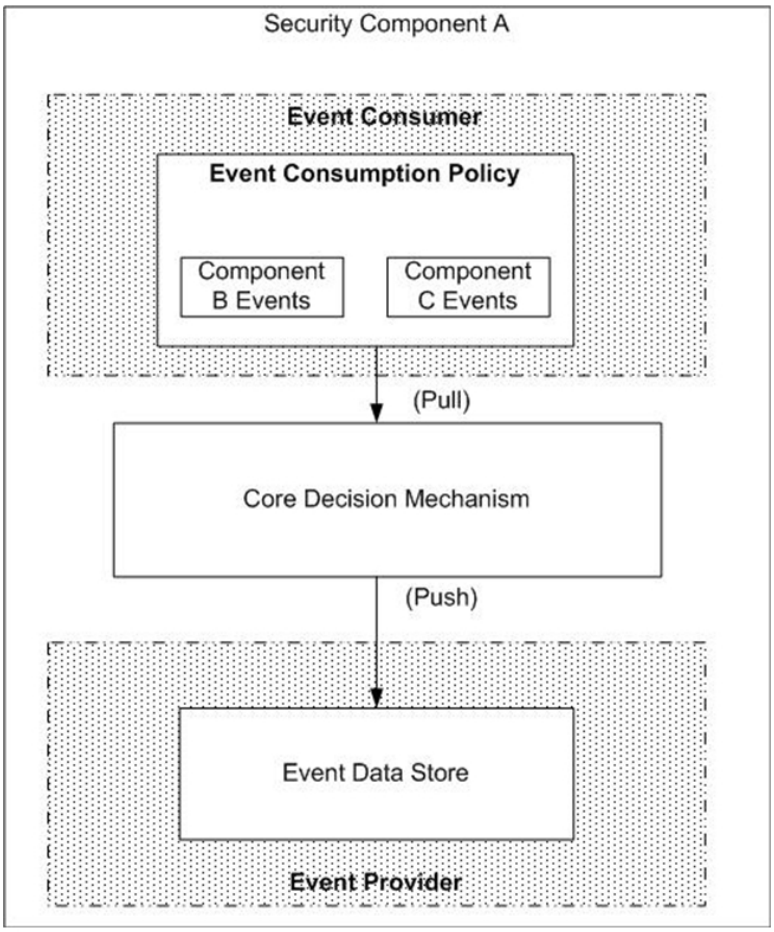


Figure 3-2. The anatomy of a security component in an open architecture. The core decision mechanism is responsible for placing new events into the mechanisms event data store which subsequently provides the data to other consumers. The core decision mechanism also pulls data from the component's event consumer module in order to enforce policy based on external event information. The event consumer module includes a policy describing the different types of events that should be drawn from the event provider (this interaction is depicted in Figure 3-3).

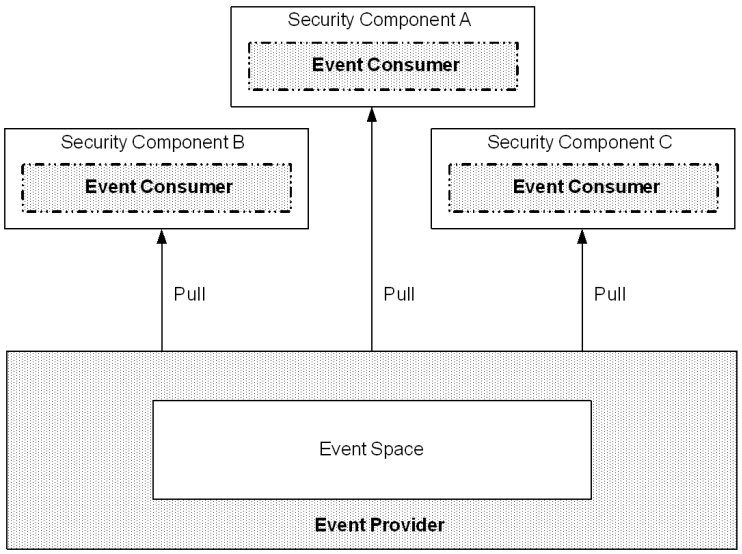


Figure 3-3. Security components with a common event provider. Rather than having to interact with each other member of the system, the components can now access data through a common event provider.

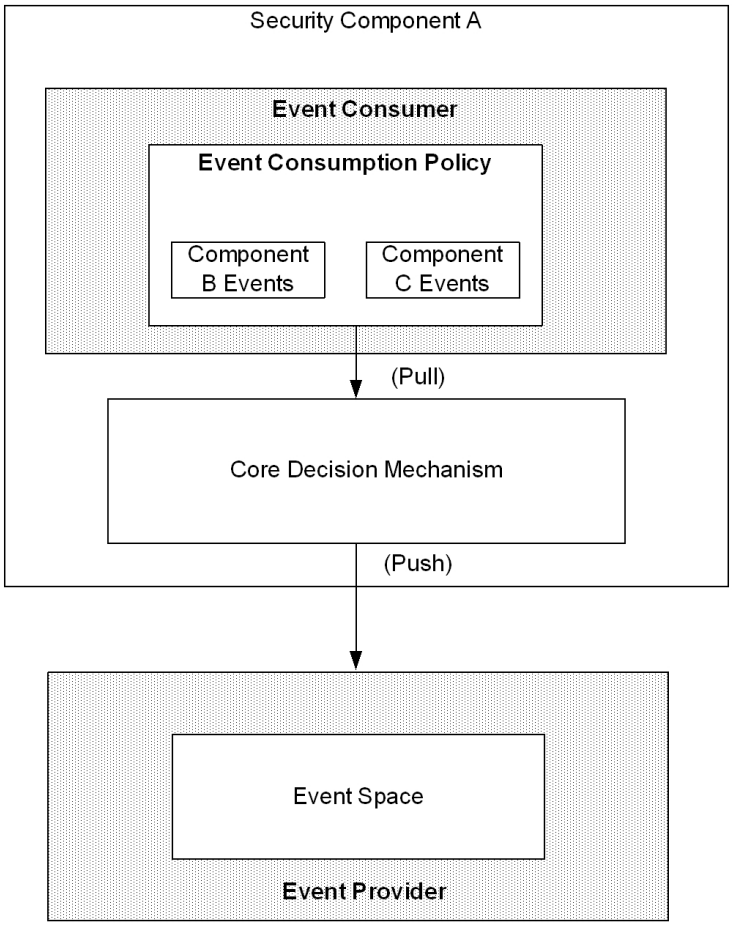


Figure 3-4. The anatomy of a security component in an open architecture. The core decision mechanism is responsible for placing new events into the common event provider that is now an external service instead of the previous data store that was contained in the mechanism itself. The core decision mechanism also pulls data from the component's event consumer module in order to enforce policy based on external event information. The event consumer module includes a policy describing the different types of events that should be drawn from the event provider (this interaction is depicted in Figure 3-3).

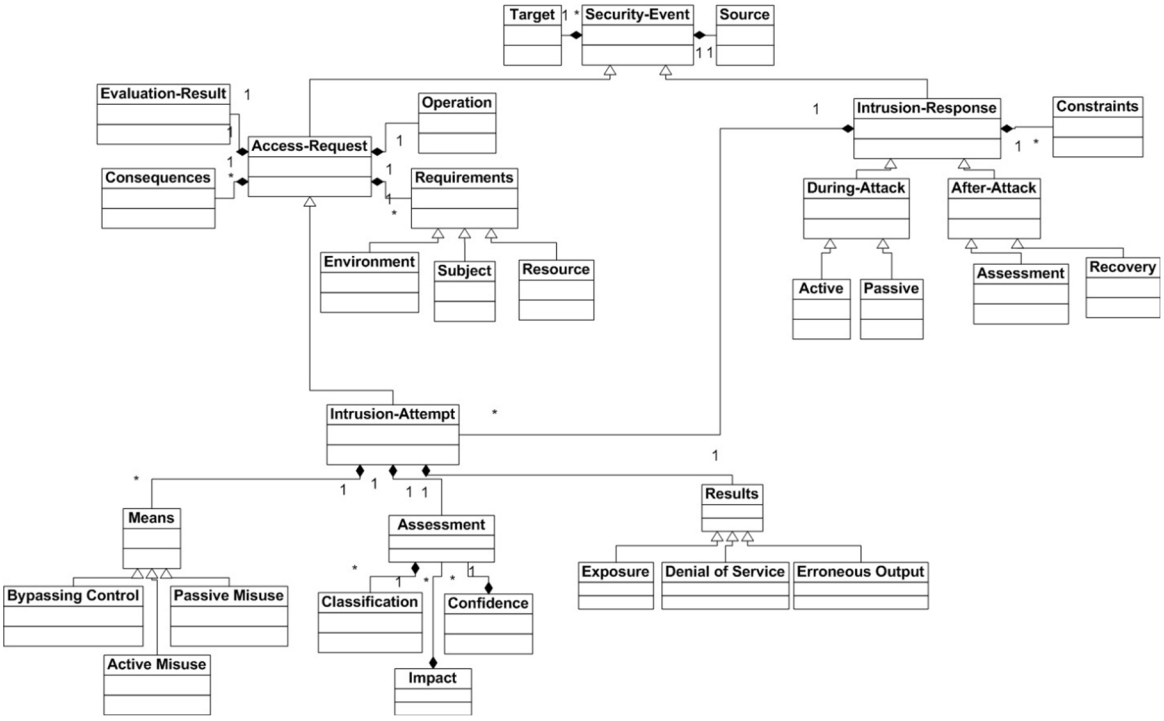


Figure 3-5. Security event information model

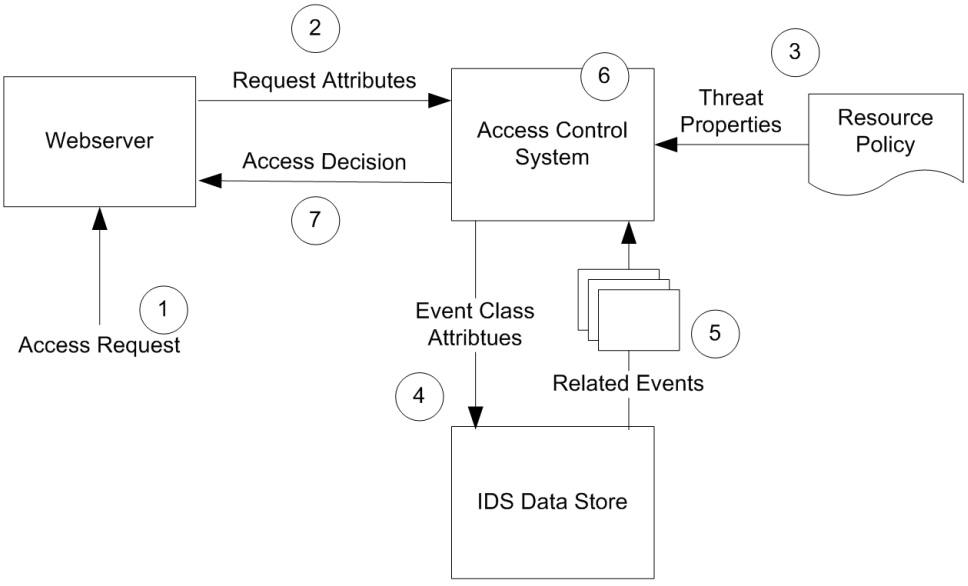


Figure 3-6. The flow of data between an IDS and a web server under the coalition-based implementation.

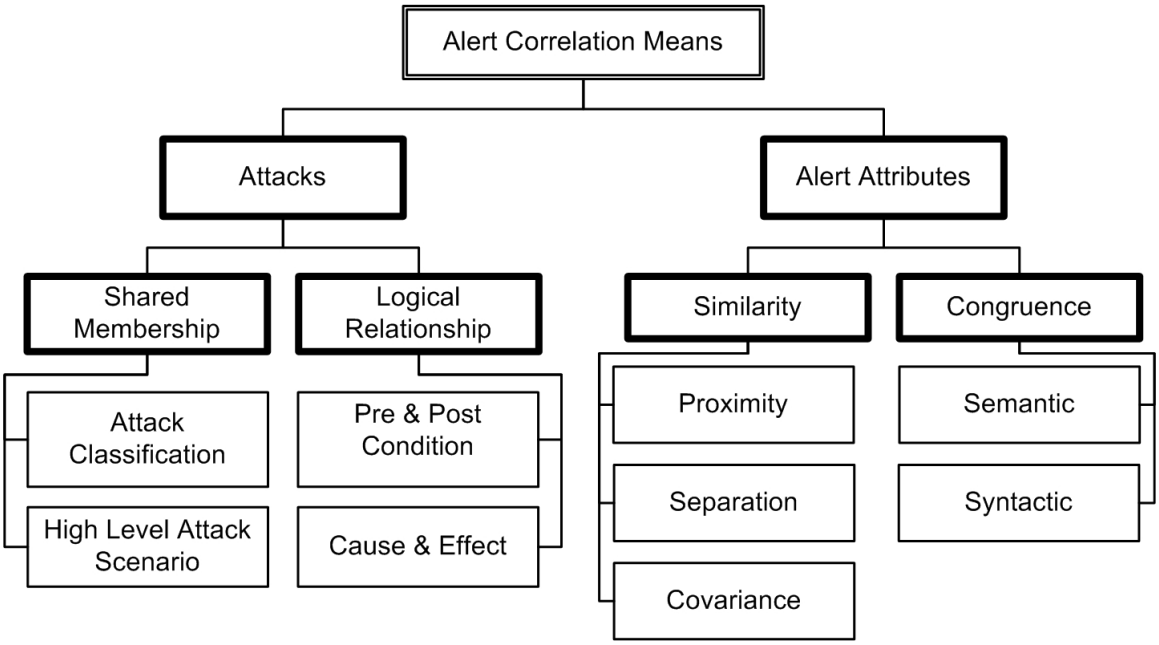


Figure 3-7. Taxonomy of the means used to achieve alert correlation.

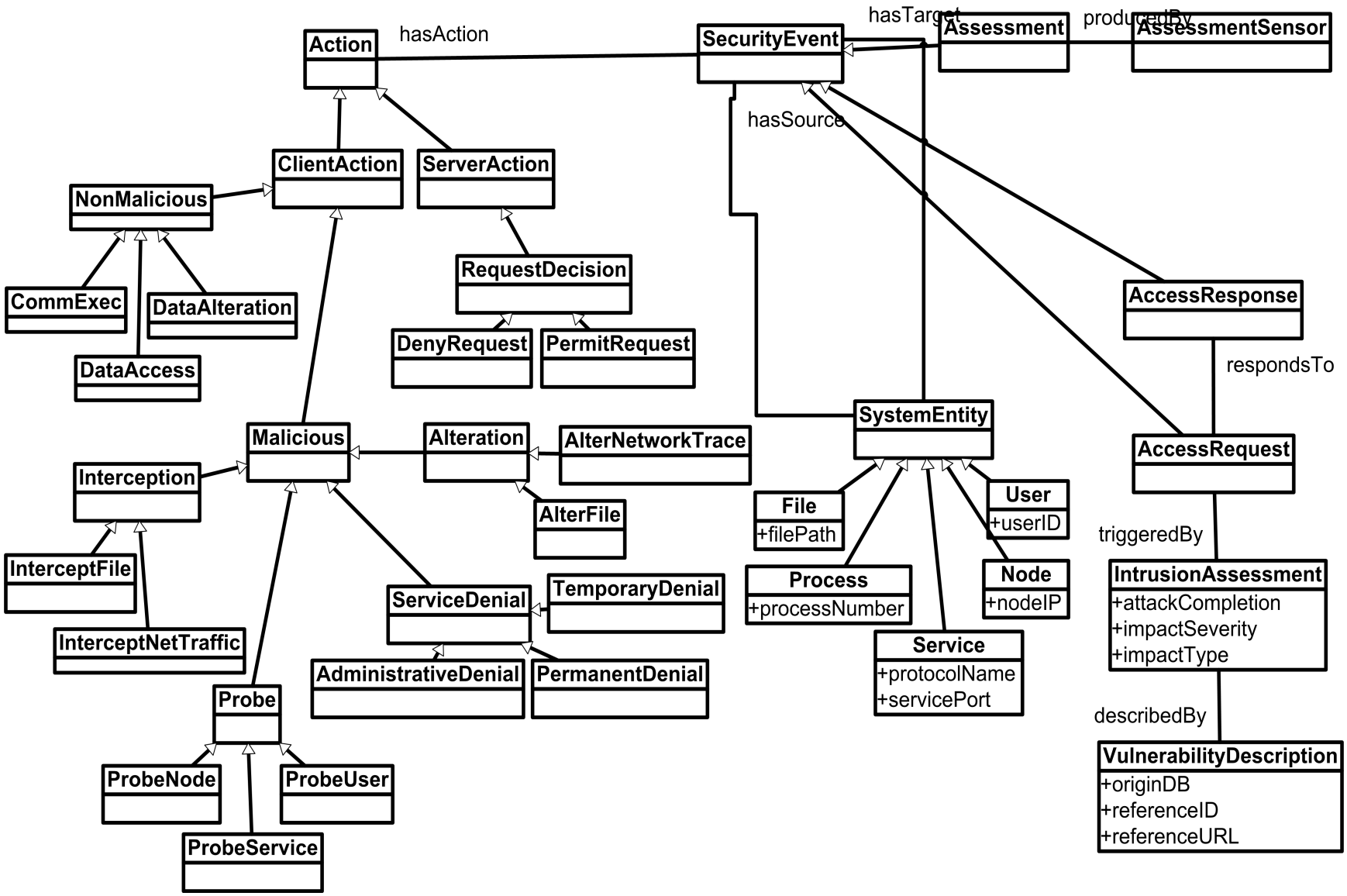


Figure 3-8. Ontology for inter-domain event correlation

CHAPTER 4
GENERAL APPROACH PART 2: CONTEXT ANALYSIS

4.1 Introduction and Design Goals

Context analysis is the process of taking event data provided by a secondary source and deriving information from that data which provides more useful indications about the state of the system. We examine the task of context data analysis on two levels: the first providing an overview of the major security measures and indicators and the relationships between them and the second by providing a detailed approach for the analysis of a specific type of context data. This chapter will propose a framework which structures several critical security measures and their determining factors. Chapter 6 will discuss the specific analysis and subsequent usage of a particular security property.

The objectives for context analysis have been divided into two main types: objectives involving how data is analyzed, and objectives involving what the product of those analyses should be. Context analysis covers all of the tasks in the system between when the data is acquired by a security component and when that is used in a decision-making process. Design goals for the approach to context analysis include:

- Actionable Data - The primary aim of the context analysis process is to produce information that represents complex system events in a relatively straightforward way that can enable autonomous responses. This requires synthesizing multiple pieces of data into higher level assessment information. The aim is to prevent the access control system from necessarily incorporating all of the factors that an intrusion detection system considers in making its decision, by providing procedures that map those properties to high-level concepts. Such high-level concepts can then be incorporated into access control policies to allow the use of real-time assessment information.
- Extensibility - Another goal is that the analysis process be based on a structured model that allows additional analysis procedures and assessment properties to be

added easily. This requires a very detailed definition of each type of assessment and how they relate to one another.

An ontology is used as the medium to define the assessment properties because it offers a more precise definition of the terms than would be possible with only words. In addition, because ontologies are described with first order predicate logic, a reasoning engine could be developed based on the specification to perform fusion of sensor data to produced the desired properties.

4.2 A High-Level Ontology of Security Assessment Information

An overview of the proposed ontology is depicted in Figure 4-1, with Figure 4-2 providing further detail on the factors used to arrive at the various assessments. At the core of the ontology there is an *Access Request* that has three aspects: it is *initiatedBy* a *Subject*, it is *directedTo* an *Object* (the resource being acted upon), and it *executes* an *Action* on the object. A *Request Evaluation* (either *Permit* or *Deny*) is based on one or more *Assessments*. Each assessment has a *Source*, a quantitative *Value*, a percentage of *Certainty* and one or more *Constraints*.

Assessments can be categorized in multiple ways: based on what is assessed and based on the type of data that is used to produce the assessment. Under the first categorization, assessments are divided into two main types: *Entity Assessments* and *Event Assessments*. Entity Assessments apply to entities such as the subject and object. Event Assessments apply to the access request itself. Under the second categorization (by the type of data used to produce the assessment) there are *Core Assessments* and *Composite Assessments*. Core Assessments are produced based on data from an event description such as an intrusion detection alert. Composite Assessments utilize one or more Core Assessments, and provide information that contrasts properties of the event under consideration with properties of an entity.

Subjects are characterized primarily by a *Trust Assessment*. Objects are characterized by *Dependability* and *Importance* assessments. Events are assigned a *Risk*. A *Threat*

assessment contrasts the trust granted to the subject with the risk of the request itself. The *Impact* assessment contrasts the risk of the request with the dependability and importance of the object. Each of these assessments will be discussed in more detail.

4.2.1 Core Assessments

4.2.1.1 Risk

In general risk denotes a probable loss to some asset or valuable property of an entity. Specifically, in this context, risk is used to quantify the probable impact of an event on the three primary security properties: confidentiality, availability and integrity. Some existing definitions of risk include the following: "a measure of the expected loss in the absence of any mitigation actions of countermeasures" [61], "a characterization of the danger of a vulnerability or condition" [62] and "the relative impact that an exploited vulnerability would have to a user's environment" [63]. The final definition is the closest to the meaning being invoked here and will also provide the factors used to determine risk.

The factors used for risk in this ontology are derived from the Common Vulnerability Specification Standard (CVSS). The CVSS gives criteria for standardizing the severity ratings given to system and software vulnerabilities. The specification includes base metrics which are solely based on the vulnerability characteristics, temporal metrics and environmental metrics. The core of the specification rates vulnerabilities on the required environment, which includes *Access Vector*, *Access Complexity* and *Authentication* and the projected impact on *Confidentiality*, *Availability* and *Integrity* - all of these are enumerated as risk factors in the ontology.

4.2.1.2 Trust

There are two main areas of concern when discussing trust: what is trust based on, and under what circumstances is it valid. To address these issues, the ontology includes *Trust Factors* and *Trust Domains*. Trust Factors are those subsidiary values which, when considered together determine the actual trust assessment value. The two trust factors included here are *Capability* and *Intent*.

Capability was mentioned in [64] and previously [65] to refer to a combination of demonstrable access and authority. This kind of definition is most relevant in the context of peer-to-peer interaction where parties are being rated according to their ability to fulfill a specific contractual relationship. In contrast, we are focused on defining trust in a manner that is restricted to an external object interacting with controlled resources in a non-malicious way. The concern is essentially to evaluate whether or not the trusted party will behave as expected (benevolently) and, if not, to what degree they have the capacity to do harm. As a result of this simplification, some of the complexities provided by other trust definitions are not relevant to the current discussion. It is assumed everyone has the same capability to perform benevolently (which is simply non-misuse), but that some parties have a greater capacity for malevolent or malicious behavior than others.

The other trust factor considered is *Intent*. Although it is a difficult concept to measure and quantify, it is the only means through which we can produce a notion of mistrust necessary for situations that require distinguishing between malicious and non-malicious users. There are two states for the property of Intent: *Benign* and *Malicious*.

Trust values also have a set of constraints that are typically called *Trust Domains*. Trust Domains are the situations or contexts in which a particular trust assessment applies or is valid. We include five domains as constraints for a trust assessment: *General* or *Universal*, *Action-Specific*, *Target-Specific* and *Context-Specific*. A *General* trust is one that is valid across an entire application or application domain and not constrained by any secondary factors. *Action-Specific* trust is granted to a subject based on the action being performed. Similarly *Target-Specific* and *Object-Specific* trust are only valid for a particular resource or object, respectively. *Context-Specific* trust is a more abstract notion that allows trust activation based on certain dynamic context properties.

4.2.1.3 Dependability and importance

Both dependability and importance of the object play a role in the way the request is viewed. CVSS includes the security requirements of the targeted asset (confidentiality,

integrity and availability) under environmental metrics that affect the severity of a vulnerability exploitation. In [66] several properties are mentioned that characterize the dependability of systems and services. An object's dependability is encapsulated by the following properties: *Availability*, *Reliability*, *Safety*, *Confidentiality* and *Integrity*. Availability is the readiness for usage, reliability is the continuity of service and safety is the non-occurrence of dire consequences on the environment. Confidentiality is the non-occurrence of unauthorized information disclosure, integrity is the non-occurrence of improper alterations of information and maintainability is the ability to undergo repairs and evolutions. These properties encompass those used in the CVSS and add a few additional properties that can be critical to contextualizing an event.

In [52] a relative measure of the objects importance is used to measure the impact of an attack. Also in [38], the importance of a network node is used to determine a suitable threshold for the risk of incoming requests. Both of these properties, Dependability and Importance have been included in the ontology as assessments of the object of the request that are needed to produce high-level assessments.

4.2.2 Composite Assessments

4.2.2.1 Threat

The *Threat* assessment is the first of the composite assessments to be discussed. Some of the definitions for threat used in the research are as follows: "the adversary's goals or what an adversary might try to do to a system", "an indication of a potential undesirable event" [61]. and "the likelihood or frequency of a harmful event occurring" [63].

In [30, 67] a global system-wide threat level is used to integrate information from outside intrusion detection systems into an advanced security policy that can specify allowed activities, detect abuse and respond to intrusions. Teo et al. [38] propose a system to manage network level system access that uses source-centered threat to regulate access control decisions. Although not concerned with a higher-level threat analysis, Valeur et al. [52] aggregate IDS alerts in multiple ways to characterize the traffic coming from a single

source or into a single target (attack focus recognition). They also consider sets of alerts between a single source-target pair (attack thread reconstruction).

In its strictest sense, threat assessment should rely on both a trust assessment for the source of the request and a risk assessment of the request itself. A threat is therefore a specific, quantifiable security risk coming from a subject (that is also assigned a corresponding trust value). This results in a notion of threat that is primarily based on two factors: the nature of the request (its risk), and the source of the request (the degree of trust given to them).

4.2.2.2 Impact

The second composite assessment measure is *Impact*. The Impact combines the risk of the request with the importance of the object and its dependability. Thus, in the case of security, an impact assessment is a judgment about the potential security damage inflicted by an access request considering the general fault-tolerance of the object and its importance to the system in which it exists.

4.3 Summary

This ontology helps define the goals of the analysis process in terms of concrete attributes that should be derived. They include measurements used in various systems and standards, but are placed in a unified structure that elucidates the differences between the various terms more precisely than mere definitions. In addition, the factors used to determine each assessment property are also listed which provides clearer insight as to how a system can arrive at the assessment and what lower level data is required as an input to the process. In an ideal system, this ontology would serve as the basis for a reasoning engine that could produce the necessary outputs given the required incoming data. The implementation discussed in Chapter 6 (see Section 6.3) will focus on a single security property (namely risk) and provide further details on how an analysis server can produce assessments of that type given input data.

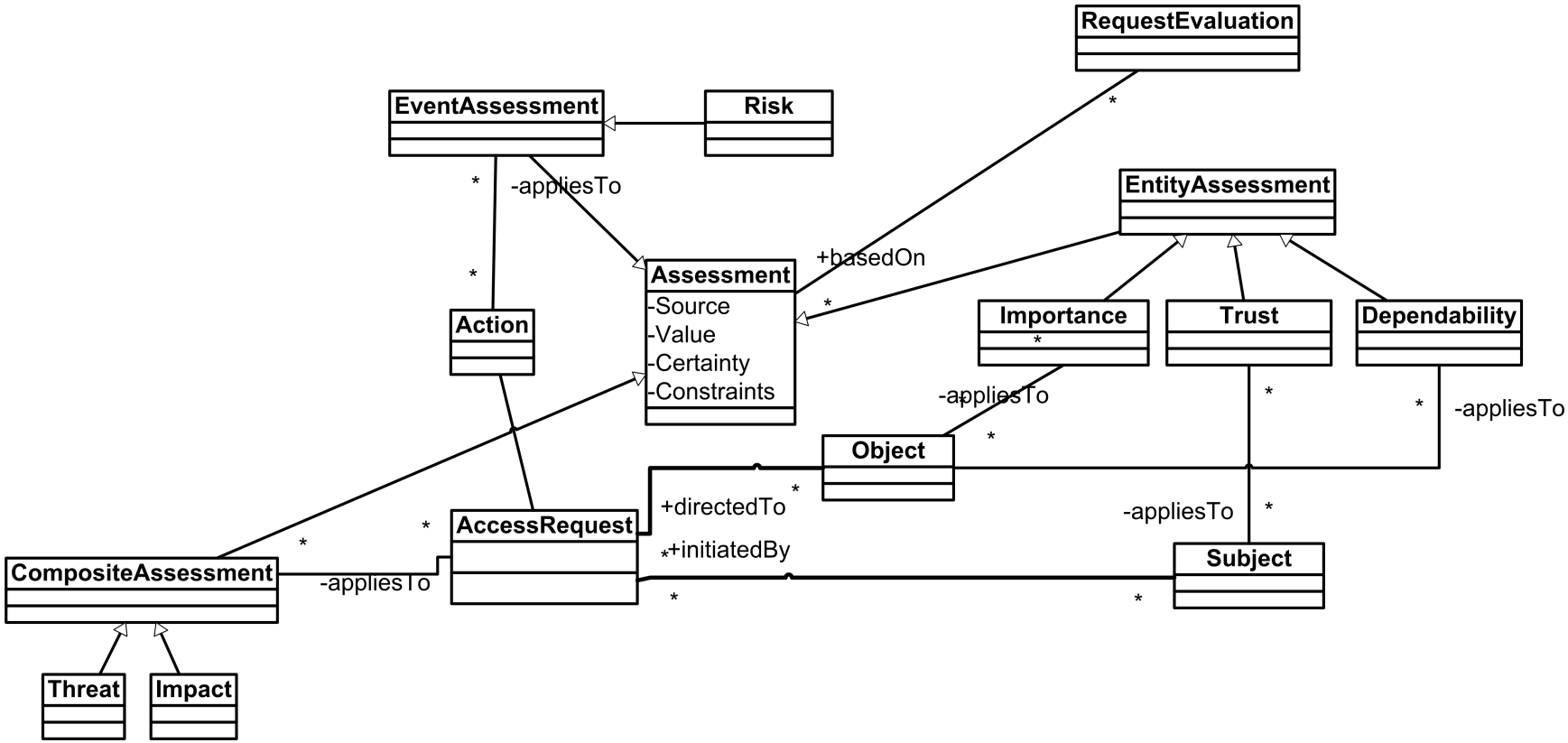


Figure 4-1. Core assessment classes. Importance, trust and dependability assessments for entities. Threat and impact assessments for access requests. Value and risk assessments for the action of an access request.

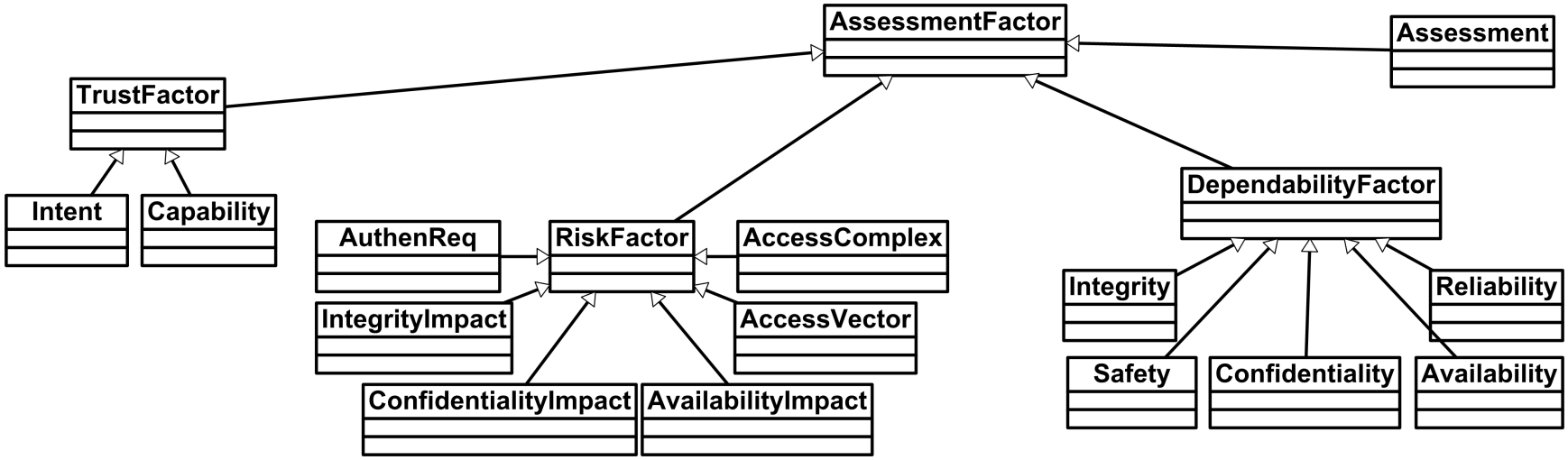


Figure 4-2. Assessment factors for three assessment types: trust, risk and dependability. The class Assessment is also a subclass of Assessment Factor because of the composite assessments that are derived from other assessments. The assessment factors for threat are the risk of the request and the trust granted to the subject. The assessment factors for the impact are the dependability and importance of the object and the risk of the request.

CHAPTER 5
GENERAL APPROACH PART 3: CONTEXT APPLICATION

5.1 Introduction

The last phase of the process of architecting context-aware behavior is the application of context data. If we define access control as a family of strategies for one party to precisely control what other parties will be allowed to do with resources that it controls [46, 45] then it becomes apparent that access control is performed at virtually every layer of a system, including at the network, operating system and application levels. To show concrete impact of the approach it is necessary to focus the application of context data on improving some aspect of the performance of access control. Some of the design goals for context application are the following:

- Responses native to the access control paradigm - The primary elements of manipulation for access control systems are permissions: the modes of interactions allowed for various subjects with system resources. There are many different types of responses to intrusions, but as the focus here is on context-aware behavior for access control, we will focus on strategies that manipulate the permissioning process at the access control level.
- Application level context application - All of the strategies used for the application of context data will be designed to be applied at the application level, meaning: any piece of software relying on an underlying operating system for the management of hardware resources. This will enable us to continue the work we have previously done in XACML, a widely adopted framework for application-level access control. This will also increase the possibility that the system for context application can be further extended by other researchers.

5.2 Context-Based Policy Evaluation

The first approach to utilizing context data that will be discussed is context-based policy evaluation. This process relies on the introduction of context-dependencies into the

access control policy that are resolved at policy evaluation time by using information from the context analysis process.

5.2.1 Access Control Schema Extension

Using the abstraction of an event as the unifying factor for security integration, it is still necessary to find a concrete representation for the attributes of those events. In most current systems, access control has the role of enforcing global security policy. All requests for resources must be checked against an access control policy before they are granted. But most present methods for access control make little or no use of intrusion detection data. The merged policy option does solve this problem by developing a schema that includes concepts from all three relevant domains, but in the process of migrating IDS concepts to the access control domain, it also migrates the intrusion detection method to a policy specification.

Our approach will be to provide a common policy by developing new attributes that can be included in an traditional access control policy. These attributes will describe the following:

- properties of intrusion detection alerts that should be taken into account when returning a value for the attribute
- the desired procedure for calculating the value returned for the attribute

In essence, the coupling between IDS attributes in the access control policy and the details and information used by the actual IDS systems will be loosened. An additional layer of abstraction will be added to make the binding to those access control attributes IDS implementation independent.

This approach also demands that the author of the mapping function understand the output of the IDS system(s). The use of a standard schema for IDS information will enable the implementer of the access control policy evaluation mechanism to write a single function for each new attribute added to an access control policy (provided that

the IDS systems under use will either use that standard schema natively, or provide a transformation from the native format to the standard one).

This policy extension will provide the means for semantic integration. One of the methods for data persistence (conversion) will be used here to return the necessary value for the access control policy from the set of relevant IDS alerts. This operation will require a limited form of conversion because the schema for IDS alerts is different than the schema for the access control policy.

Based solely on the aggregation relationships specified in [9] and using the concept of threat outlined previously, the following attributes were added to the XACML schema of the access control mechanism described in Section 3.3.2:

1. source-target-class-threat - provides the total severity of the threat from this source to this resource in this class of attack
2. source-target-all-class-threat - provides the total severity of the threat from this source to this resource for all classes of attack
3. source-class-all-targets-threat - provides the total severity of the threat from this source to all resources with this class of attack
4. all-sources-target-class-threat - provides the total severity of the threat from all sources to this resource in this class of attack
5. source-all-targets-all-threats-threat - provides the total severity of the threat from this source to all resources for all classes of attack
6. all-sources-target-all-classes-threat - provides the total severity of the threat from all sources to this resource for all classes of attack
7. all-sources-all-targets-class-threat - provides the total severity of the threat from all sources to all resources for this class of attack

Each of these rules contains at most two attributes: the threat threshold value and an attack class for cases 1, 3 ,4 and 7. The source and target will be specied in the access control request and therefore do not need to be specied in the policy - the values for source and target will be inherited from the request values. Examples of access control rules incorporating these properties are provided in Figures 5-1, 5-2, 5-3, 5-4 and 5-5.

The process for access control schema extension under the federated implementation is dicussed in Section 6.5. Unlike the XACML-based implementation, because the access control process in the Apache webserver was not designed to incorporate contextual information into the schema, the process of schema extension is less structured. The subsequent discussion will, therefore rely on the XACML schema abstractions for discussing the incorporate of context data in the application process.

5.2.2 Application Scenarios

We provide some examples of situations where our approach can be used to provide very granular IDS-aware access control.

Threat escalation by a single source. In this first scenario we have a single source performing multiple intrusive requests against various system resources. The aim in this case is to restrict access to system resources from this source after the threat profile for that source passed a given threshold. This application is similar to the task of packet filtering which is performed by some Intrusion Prevention Systems (IPS) but with a few major differences. The first is that this technique can be applied dynamically based on the past history of the source whereas packet filtering must be configured manually to filter based on source. The second is that, because this restriction is performed at the application level, there is a wider range of possible responses available. Access could be denied entirely, similar to packet filtering, or only specific access rights could be revoked, allowing a more measured response.

In a concrete example we consider a web server with three available resources: R_1, R_2, R_3. Each of the resources is governed by a policy that denies requests coming from sources with a threat profile above 25. The basic structure of the rule is shown in Figure 5-1. Each resource is assigned an impact value ranging between 1 and 3, and confidence value ranges between 1 and 5 based on the number of systems referencing the vulnerability used.

Two different hosts: S_1 and S_2 make requests to the server in parallel. Host S1 attempts three web cgi exploits against the server, with its overall threat profile increasing each time. By the time it requests R1 on its fourth overall request to the server, its threat profile exceeds the threshold allowed in the policy that controls R1, R2 and R3 and is therefore denied access to all of them. Simultaneously, because S_2 has maintained a threat profile of 0, it can continue to access all of the resources available. An example of this scenario is summarized in Table 5-1.

Threat escalation against a single target. In this second scenario, we consider a web server with two available resources: R_1, R_2. Access to each of the resources is regulated by a policy that denies write and update requests when the threat profile for the resource is over 30. The basic structure of the threat rule is shown in Figure 5-2. Three different hosts: S_1, S_2 and S_3 make requests to the server in parallel. In this case, however after two independent requests from S_1 and S_2 for R_1 , the threat profile of the resource is at 20. As a result, a third request for R_1 from S_3 is denied. All of the hosts, however, can still access R_2 because none of their individual threat profiles exceeds the threshold set in the policy that controls R_2. An example of this scenario is summarized in Table 5-2.

5.3 Context-Based Threat Response

The next strategy for context application extends the notion of context-based policy evaluation by identifying traditional intrusion response strategies and applying them based on context data, while still maintaining context dependencies in the access control policies. We will first survey the available intrusion response methods, noting the factors that determine the circumstances under which they are used effectively. We will then utilize the available context data produced during the analysis process as a criteria for employing appropriate mitigation, prevention and response methods. Another difference between this application technique and the previous one, is that each of these response methods will be dependent on multiple properties to be applied whereas the previous context dependencies only introduced a single property into the access control policy.

Carver [68] offers a number of responses along with the situations they can be used appropriately. We present a summary of the most relevant methods in Table 5-3, along with their respective implementations at an access control level. Implementations for four of the six aforementioned response methods are ongoing: restricting user activity (both specific actions and all actions), blocking access to threatened services, and forcing additional authentication.

- *Forcing additional authentication* - this approach will be implemented by adding a rule to the access control policy for the desired resources that specifies additional authentication as the obligation if certain conditions are met. The request is denied pending further authentication.
- *Complete User Activity Restriction* - accomplished by keeping a dynamically updated black-list of sources that have exceeded allowable thresholds for intrusive behavior and simultaneously have a low trust amount. A new attribute will be provided that will require a custom attribute evaluation module (similar to the ones used to perform context-based policy evaluation in Section 5.2). This module will return a Boolean value indicating whether the requesting user is unconditionally blocked or not. This evaluation module will query a blacklisting service that runs within the policy decision point itself. This blacklisting service will examine available behavior within a pre-determined time window and will blacklist users temporarily when their behavior exceeds the allowable threshold. The threshold will also be dependent on the level of trust accorded to the user - a greater amount of trust gives a correspondingly higher threshold. An example of a policy to block or lockout a user is shown in Figure 5-4.
- *Blocking Access to Threatened Services* - also achieved through a new policy attribute evaluated with a custom evaluation module that returns a boolean value indicating if requests to that target are being denied due to an over whelming amount of suspicious traffic. If this attribute is used in a policy, then this module

will query a service monitoring incoming requests to the services under control of the policy enforcement point. Based on the required availability of the target recorded in an asset database a threshold will be set for the amount of intrusive behavior that can be ignored before the access to the resource should be blocked. An example of a policy designed to block access to a threatened resource is shown in Figure 5-3.

- *Restricting User Activities* - a new set of policy attributes will be introduced to designate the restricting of specific permissions. A module will be provided for each attribute which will check if the source of the requests meets the criteria to be able to perform that specific action - if not that specific request will be blocked. Each resource will have thresholds for source trust, the severity of the threat and the certainty of the assessment that will be inputs to the function that determines if the specific permission will be allowed for that request. An example of a policy to restrict user activity is shown in Figure 5-5.

5.4 Summary

Two general uses for context with regards to access control are explored here: the evaluation of policies with contextual dependencies and the triggering of responses based on context data. The context-based evaluation of policies requires that the policy schema is extended with new attributes and that dependencies are added to the policy that force context information to play a role in the decision issued by the enforcement mechanism. Context-triggered response, however, focuses on identifying specific scenarios which can be indicated by context data and then matching those scenarios with a high-level countermeasure that will help mitigate the risk in that scenario. As a result of the fact that access control is primarily a policy enforcement mechanism there is an element of commonality between the two approaches because the implementation of the response techniques will likely take place at the policy level. Context-triggered response builds on the inclusion of context data into the policy evaluation process and provides a response based on comparing multiple types of individual contextual properties.

Both of the methodologies outlined here (context-based policy evaluation and context-based threat response) are incorporated into the system implementation detailed in Chapter 6. Context-based policy evaluation is achieved by directing request evaluation to custom access handlers that interact with an analysis server and then return a decision after checking the risk value against thresholds set in the policy specification. Context-based threat response is incorporated in that each access control handler enables a different response more suitable to one situation or another based on examing specific context information.

```
<Rule RuleId="Source-Threat" Effect="Deny">
  <Condition
  FunctionId="integer-greater-than-or-equal">
    <Apply FunctionId="integer-one-and-only">
      <SubjectAttributeDesignator
      DataType="integer" AttributeId="total-source-threat"/>
    </Apply>
    <AttributeValue DataType="integer">20
    </AttributeValue>
  </Condition>
</Rule>
```

Figure 5-1. XACML rule including source-centered threat. This rule demonstrates the extension of the XACML schema with a new property *total-source-threat*. This property is designated as an attribute of the subject of the request. An integer function is used to compare the value returned for this property with the designated value of 20. If the total-source-threat property is greater than or equal to this value, then the rule has the effect of causing the request to be denied.

Table 5-1. Escalation of threat in subsequent requests by two different sources. When the threat is assigned to individual sources seperately, the system is able to distinguish between malicious and non-malicious subjects.

Source	Request	Threat	Total Source Threat
S_1	1st	8	8
S_1	2nd	10	18
S_1	3rd	8	26
S_2	1st	0	0

Table 5-2. Escalation of threat in subsequent requests by three different hosts on a common target. When the effect of requests from different subjects to the same object are considered in aggregate, the system is able to contextualize individual requests into an overall pattern of interaction with the object.

Source	Order of Request	Threat	Total Target Threat
S_1	1st	10	10
S_2	2nd	10	20
S_3	3rd	0	20

```
<Rule RuleId="Target-Threat" Effect="Deny">
  <Condition
  FunctionId="integer-greater-than-or-equal">
    <Apply FunctionId="integer-one-and-only">
      <ResourceAttributeDesignator
      DataType="integer"
AttributeId="total-target-threat"/>
    </Apply>
    <AttributeValue DataType="integer">
    30</AttributeValue>
  </Condition>
</Rule>
```

Figure 5-2. XACML rule including target-centered threat. This rule demonstrates the extension of the XACML schema with a new property *total-target-threat.* This property is designated as an attribute of the resource being accessed. An integer function is used to compare the value returned for this property with the designated value of 30. If the total-target-threat property is greater than or equal to this value, then the rule has the effect of causing the request to be denied.

```
<Rule RuleId="Resource-Disabled" Effect="Deny">
  <Condition
  FunctionId="boolean-equal-to">
    <Apply FunctionId="boolean-one-and-only">
      <ResourceAttributeDesignator
      DataType="boolean"
AttributeId="resource-lock-status"/>
    </Apply>
    <AttributeValue DataType="boolean">
    true</AttributeValue>
  </Condition>
</Rule>
```

Figure 5-3. XACML rule including an attribute indicating that a resource is locked. This rule demonstrates the extension of the XACML schema with a new property *resource-lock-status.* This property is designated as an attribute of the resource being accessed. A boolean function is used to compare the value returned for this property with the designated value of 'true'. If the resource-lock-status property is true, then the rule has the effect of causing all requests to this resource to be denied.

```
<Rule RuleId="User-Account-Lock" Effect="Deny">
  <Condition
  FunctionId="boolean-equal-to">
    <Apply FunctionId="boolean-one-and-only">
      <SubjectAttributeDesignator
      DataType="boolean"
AttributeId="user-account-lock-status"/>
    </Apply>
    <AttributeValue DataType="boolean">
    true</AttributeValue>
  </Condition>
</Rule>
```

Figure 5-4. XACML rule including an attribute indicating that a user account is locked. This rule demonstrates the extension of the XACML schema with a new property *resource-lock-status.* This property is designated as an attribute of the subject initiating the request. A boolean function is used to compare the value returned for this property with the designated value of 'true'. If the user-account-lock-status property is true, then the rule has the effect of causing all requests from this source to be denied.

```
<Rule RuleId="Write-Prohibit" Effect="Deny">
  <Condition
  FunctionId="boolean-equal-to">
    <Apply FunctionId="boolean-one-and-only">
      <SubjectAttributeDesignator
      DataType="boolean"
AttributeId="user-write-prohibit"/>
    </Apply>
    <AttributeValue DataType="boolean">
    true</AttributeValue>
  </Condition>
</Rule>
```

Figure 5-5. XACML rule including a property to restrict a specific permission . This rule demonstrates the extension of the XACML schema with a new property *user-write-prohibit.* This property is designated as an attribute of the subject initiating the request. The Policy Decision Point will be extended with a new module that provides the logic to provide a current value for this property. An boolean function is used to compare the value returned for this property with the designated value of 'true'. If the user-write-prohibit property is true, then the rule has the effect of causing the request to be denied.

Table 5-3. Selected intrusion response strategies. Each general response strategy is listed along with its appropriate use case, its implementation at the access control level and the contextual properties that constrain its application.

Response strategy	Use case description	Implementation for access control level response	Application constraints
Locking user accounts	Compromise of user account in question	Global policy that denies requests from a particular source	High certainty of attack and high risk evaluation for the source with low to moderate user trust
Disabling the attacked ports or services	Making the port or service inaccessible	Global policy that denies requests to a particular target	High Certainty, High target-centered threat, Low Target Availability requirement
Force additional authentication	May slow or stop intrusions (especially automated ones), while authorized users will continue	Policy that requires multiple authentication tokens for a request to be granted	Used with low-certainty, Source, Target or Pair-Centered Threats of Moderate to High level
Restrict user activity	Suspicious users may be restricted to a special user shell that allows some functionality while restricting certain commands	Policy that separates allowed actions into levels based on the perceived threat class of the user	1) low to mid certainty of attack and low to mid user trust 2) high certainty of attack and high trust (which would indicate the account has been compromised) 3) low to mid threat severity

CHAPTER 6
ADAPTIVE RISK-AWARE ACCESS CONTROL FOR WEB SERVERS

6.1 Introduction

6.1.1 Connection Between the Implementation and Previous Chapters

The previous chapters have sought to answer some of the fundamental questions regarding how context aware security systems can be designed. The design process has been broken down into a few main elements to ensure that at each step helps fulfill the larger design goals of the system.

This chapter will focus on addressing some of the more specific questions regarding the use of intrusion detection data in an access control process, such as: dealing with data inaccuracies and ensuring performance in the face of large amounts of incoming data and high request frequency. The solutions presented in the implementation will rely on the groundwork established in the previous chapters.

A federated approach is used that follows the principles discussed in Section 3.2.2. The correlation ontology of Section 3.4.5 is adapted to a relational database model that serves as the schema for the event database described in Section 6.5. An independent analysis server performs the functions of aggregating context information and then disseminating it. The implementation focuses on one of the context properties defined in the Chapter 4 ontology of assessment properties - that of risk. A model is defined for deriving risk information from assessments produced by an intrusion detection system on attempts to exploit software vulnerabilities. This model designates how risk is assigned to both subjects and objects of access requests. Details are also provided on how the analysis data is applied and used in the access control process. Context-based policy evaluation (previously discussed in Section 5.2) is achieved by providing custom access control handlers that interact with the analysis server to receive risk information. These access control handlers also implement the response techniques discussed in Section 5.3.

6.1.2 Implementation Overview

The proposed solution for achieving attack-resistant access control is the use of real-time assessment data in access control policy evaluation and enforcement. Specifically, evidences of vulnerability exploitation are collected and analyzed into a higher level risk assessment for the sources and targets of access control requests. This risk assessment is subsequently used as an additional parameter or contextual property in access control policies so that permit and deny decisions for an incoming request are based on an assessment of the risk posed by the requesting source and/or the risk posed to the targeted resource. This approach has been termed the **A**daptive **A**ssessment-**B**ased **A**ccess **C**ontrol **S**ystem (ABACUS for short).

Two closely-related strategies for implementing this approach are discussed. The first is an approach relying on on-demand analysis, abstracting the three parts of the context management process (acquisition, analysis and application) into different server mechanisms and performing the analysis by aggregating requests and deriving a risk assessment as new requests come in. A set of results are then provided for the testing done with an implementation of this approach, along with conclusions on the constraints and limitations of this approach.

The second approach differs in that the analysis function is triggered by new security events in the system and consequently, the analysis does not take place as a function of an incoming request. Risk assessments are continually maintained for all of the entities in the system. As new assessment data becomes available, those risk assessments are updated for the entities in that event. A set of results are then provided for the testing done with this approach along with conclusions on its constraints and limitations. Finally, a summary and relative comparison of the two approaches is offered.

6.2 Intrusion Response and Attack Resistance

6.2.1 Strategy Selection

The strategies put forth in the literature for responding to intrusions and attempted system attacks are very numerous and varied. Therefore, it is necessary to select only those that most closely match the requirements for achieving the desired goal: namely, attack resistant access control. The first restriction is that the responses applied should serve to manipulate some element in the access control domain. Access control is primarily concerned with a set of subjects, a set of objects and the specific operations that each subject can perform on each object. So our response technique must manipulate these permissions, either at the subject side (by designating which actions a subject can perform) or at the object side (designating what can be done with the object). The second requirement is that the strategy or response can be triggered using risk data.

A number of different intrusion responses are detailed in [49, 48]. Using the criteria just discussed, however, the following three strategies were selected as appropriate for this application: 1) forcing (additional) authentication, 2) restricting subject permissions 3) restricting object permissions.

Forcing authentication could take a number of forms. The first would be forcing anonymous authentication. This is a strategy that has become somewhat common in the Internet today, that implements an authentication check not based on a shared secret between the user and the host system (such as a password) but based on the subjects ability to perform an operation that distinguishes him from a class of undesirable users (frequently automated attack scripts). Another form of authentication would be a traditional password check that establishes the actual identity of the user. In either form, however the aim is to ensure that the user requesting access is not a member of the set of users who should be denied access to the resource.

The response of restricting subject permissions also takes more than one form. The first restricts the subject from performing a specific operation or restricted set of

operations across the entire set of system resources. This somewhat assumes that the set of systems resources have a set of common actions or operations. The second form is an extension of the first, that adds all of the available operations to the restricted set, effectively locking the subject out of performing any action on any of the system resources.

Similarly to the previous two operations, the response of restricting permissions on a target has two forms. The first restricts all subjects from performing a specific action or restricted set of actions on the object in question. The next form blocks all subjects from performing any operation on the object. These techniques satisfy the previously mentioned requirements and provide a framework of responsive behaviors that can be used to curtail or limit intrusive actions in the system.

6.2.2 Response Triggering

The next aspect to detail is when the response techniques will be employed, or based on what conditions will they be activated and how will those conditions be described. Our approach to response selection is roughly within the third category of the intrusion response taxonomy mentioned in - cost-sensitive response selection. The author of the access control policy is responsible for deciding which security risk factors (ie. global system risk, risk from the requesting source or risk to the target) will be used during the process of evaluating whether or not a request will be permitted. These individual measures are therefore the inputs into the response selection process. Each risk factor is then matched with a threshold that determines when the action associated with the factors should be performed.

6.3 Notion of Risk and a Preliminary Risk Assessment Model

Risk was previously defined, along with its critical determining factors, in the assessment ontology from Section 4.2.

6.3.1 Analysis Model

We consider the following basic scenario: a new access control request is generated r_1 . This request has a source s_1 and a target t_1. We take an approach to assessing

the risk of the request that relies primarily on assessing the source and target of the request. Therefore, the first step is to aggregate the set of requests generated by the source $R(s_1) = r_a$, r_b, $r_c ... r_n$and the target $R(t_1) = r_f$, r_g, $r_h ... r_m$ and thereby assign risk to those entities. The first section will deal with how we arrive at a risk assessment for each request in the aggregate sets for the source and target. The next section will then deal with how those values are combined to arrive at a single assessment for the entity.

Estimating risk for past events. Risk is associated with a probable intrusion attempt, evidenced by an attempt to exploit a system vulnerability. The risk posed by a request, therefore, is proportional to the severity of the vulnerabilities it is suspected to be seeking to exploit.

The CVSS standard provides a widely accepted, quantitative measurement scale for the severity of vulnerabilities, and therefore we will leverage that standard for the rating of vulnerabilities. The overall method for providing a single vulnerability estimate based on multiple vulnerabilities spread out over time is derived from the method used in [69]. The method has been adapted, however, to take as input a set of vulnerabilities associated with a request, instead of the set of vulnerabilities that apply to a particular service. The function $R(r_j)$ given below provides an estimation of the total risk for a request r_j by taking the exponential average of all of the vulnerability descriptions associated with that request. The exponential average was chosen, as noted in [69], to provide an risk estimate for the request that is at least as large as the highest severity vulnerability associated with the request.

$$R(r_j) = \sum_{v_k \epsilon V(r_j)} e^{(SS(v_k) * Decay(r_j))} \tag{6–1}$$

$$Decay(r_j) = e^{-\beta * (currenttime - requesttime(r_j))} \tag{6–2}$$

In many cases, an alert is triggered by an intrusion detection system and because of the nature of the request it could correspond to multiple vulnerabilities. The set

$V(r_j)$is the set of all vulnerability exploitation signatures triggered by the request r_j. $SS(v_k)$ is the magnitude of the vulnerability v_k. $Decay(r_j)$ serves as a weighting for each vulnerability based on the age of the request. It determines the amount of the original magnitude that remains as a function of time - this allows more recent information to play a more prominent role in a risk evaluation. β is the decay period after which the magnitude of the request begins decreasing over time.

Estimating the risk posed by a source or to a target. The risk posed by the source of a request is then the weighted sum of the risk values for all of the requests that can be attributed to that source. We define the function $SR(s_i)$ to be the risk assessment assigned to a source s_i. This is given by the following formula:

$$SR(s_i) = \gamma * ln(1 + \sum_{x \epsilon \{H,M,L\}} w_x * \sum_{r_j \epsilon HV_x(s_i)} R(r_j)) \tag{6–3}$$

All of the requests initiated by a source s_i and associated with an attempt to exploit a system vulnerability are contained in the set $HV(s_i)$. This set is then divided into three subsets ($HV_L(s_i)$, $HV_M(s_i)$ and $HV_H(s_i)$) based on the magnitude of the risk for the request. The set $HV_x(s_i)$ denotes all of the requests with vulnerability exploitation assessments of a certain severity level (Low, Medium or High) for which s_i is the source. Each set is summed into the total risk evaluation with a weighting of w_x. The final weighting γ amplifies the output so that differences between different sources can be viewed more effectively. The exponential average is used again, to give an overall estimate that is at least as large as the weighted sum of the highest severity vulnerabilities in each of the three classes. The logarithm of the weighted sum is taken to keep the output within a range that is manageable and which can be

Similarly, the risk associated with a target t_i is given by by the function $TR(t_i)$:

$$TR(t_i) = \gamma * ln(1 + \sum_{x \epsilon \{H,M,L\}} w_x * \sum_{r_j \epsilon HV_x(t_i)} R(r_j)) \tag{6–4}$$

In this case, however, the set $HV_x(t_i)$ denotes all of the requests with vulnerability exploitation assessments of a certain severity level (low, medium or high) for which t_i is the target.

Contributions of this assessment model. Extending the work done on security metrics that assess the state of the system at a given point in time we propose a) the use of security metrics to monitor the system state in real time and b) to focus the use of system metrics on assessing the principal entities in access control requests, namely: request sources and targets. The focus is therefore using vulnerability exploitation information to develop risk assessments for entities in a system.

In particular, we have used a method based on the one in [69] to combine the magnitude of multiple vulnerabilities spread out over time, but focused on when the request was generated instead of when the vulnerability was discovered as a means for assessing the security of a particular service.

6.4 Triggering Restricted Permissioning With Risk Data

Although the cost determination equations for response selection are highly system dependent, the risk progression in Figure 6-1 is provided as an example and has been tested using the model discussed previously. For this specific progression, the attacker executes exploitation attempts of mid-severity every 60 seconds. The risk progression would change if any of the variables such as the risk rating of the individual requests, the inter arrival time between requests, or the weighting of the low, medium and high level risk events were adjusted.

Using the example risk progression, the following sample conditions are provided for performing each of the previously mentioned intrusion responses:

1. if Source_Risk >= 36.11 OR System_Risk >= 53.8 THEN Force_Authentication
2. if Target_Risk >= 41.11 THEN Restrict_Permission_X_On_Object
3. if Source_Risk >= 41.11 THEN Restrict_Permission_X_For_Subject

The first condition forces authentication for the subject if the risk generated by the subject exceeds 36.11 (roughly three exploitation attempts of mid-severity) or if the overall system level threat exceeds 53.8 (fifteen exploitation attempts). The second condition denies the subject from performing action X on the object if the target risk has risen at or above 41.11 (meaning it has received 5 or more exploitation attempts). The last condition denies the subject from performing action X on any objects if the source risk is at or above 41.11 (meaning that 5 or more exploitation attempts have been attributed to that subject).

6.5 Abacus Framework Architecture

The architecture abstracts the risk analysis functions into an external risk analysis service which the access control system is then adapted to interact with. The access control system used to demonstrate this architecture is the Apache web server. This second approach corresponds roughly to a federated or service oriented approach. The three risk types discussed previously: source, target and system are each implemented in an analysis module which can provide a risk assessment for the appropriate entity (or in the case of the system level risk for all of the entities). All of the analysis data is then made available by an analysis service that receives and services requests for risk analysis information. The web server is also extended to perform the three intrusion responses discussed previously as the means to attack resistance: forcing additional authentication, restricting user permissions and restricting access to a target. Based on the resource and the actions available on that resource, a threshold is determined for the source and target associated risk above which, requests are denied.

Event database. The event database is backed by a relational database implementation (in this case Muscle). Some of the structure of this database was derived from the IDMEF schema [50]. Other parts of the structure were produced as part of a larger ontology for security assessment parameters, which is soon to be published. The event database contains the following tables:

- CVSS Vulnerabilities - this table stores information regarding current vulnerabilities from the National Vulnerability Database (NVD), which has adopted the CVSS scoring system. Each vulnerability is listed along with its CVSS base score, exploit sub-score, impact sub-score, overall score and vector. The name of the vulnerability, the product it affects and the versions of that product that are vulnerable are also stored in this table.
- Network Access Requests - Entries in this table are generated on the receipt of an IDS alert by the alert processing engine. The IP address and port of the source node are listed along with the IP address and port of the target node. The time of the request, action being performed and target entity are also included in this table.
- Files - listing all file entities referenced in requests; includes the file's path and a reference to the node on which the file is hosted
- Nodes - listing of all node entities referenced in requests; includes the nodes IP address
- Port - listing of all ports referenced in requests; includes the node that the port was on, and the protocol to which it was bound
- User - listing of all users referenced in requests; includes their user id
- Intrusion Assessments - this table links individual requests to an intrusion assessment. Each assessment provides a classification for the event, its severity (which may be provided by the IDS) and whether or not the attack completed successfully. It also includes the id for the analyzer which produced the assessment and any additional data that the IDS provides, such as the packet payload, etc.
- Vulnerability Descriptions - a vulnerability description provides information on a concrete software vulnerability. Each vulnerability description is provided by a vulnerability database (for the purposes of this study we only use CVE vulnerabilities because they have an objective scoring system). Each vulnerability description,

therefore only links to one element in the table of CVSS vulnerabilities and, consequently, only has one base score. The table also stores the reference name and URL, along with a link to the intrusion assessment which references the vulnerability.

- Request Risk Cache - this table stores a calculated risk value for each request id by querying for the CVSS score for all of the vulnerability descriptions that are linked to an intrusion assessment (and which provide a CVE ID). As mentioned in the section describing the model, the exponential average of all of the CVSS scores for the vulnerability descriptions used in a particular intrusion assessment are taken, and this value is stored in the request risk cache. When a particular risk handler queries the risk cache to produce a risk evaluation for a particular entity, the risk estimate is multiplied by the decay factor to produce a dynamic risk estimate for that particular request.

Dynamic risk modules. The three dynamic risk modules implement the functions described under the risk model. The functions for each handler are the same, with the exception of the first step. In the case of the source risk handler, all of the requests originating from that source are aggregated. For the target risk handler, all of the requests directed at that target are aggregated. Lastly, for the system risk handler, all of system requests are aggregated. Following this, for those existing requests that may be cached in the request risk cache, the estimate value is pulled and the decay function is calculated. If no value is cached, then the risk handler calculates a risk estimate by joining the request, intrusion assessment, vulnerability description and CVSS vulnerability tables to find all of the CVSS scores for all of the vulnerability descriptions referenced as a part of the intrusion assessment for the request. Based on this, a static risk estimation is produced and cached for future access.

Alert processing module. The alert processing module is responsible for extracting the information for each of the tables mentioned previously from the alerts it receives. In addition it can perform the functions of filtering out alerts that do not reference concrete

vulnerabilities, or alerts for which the vulnerability does not match the current system configuration.

The architecture is shown in Figure 6-2. The analysis server performs risk analysis operations, providing risk assessments for various entities (sources and targets) based on requests from the access control system. The intrusion detection system listens on the link for incoming requests and reports alerts for any requests that seem intrusive (in this case specifically, those requests that appear to be an attempt to exploit a known software vulnerability). The raw alerts from the IDS are passed through a processing module that may filter the alerts using concrete vulnerability or configuration verification as mentioned earlier. Finally, the data from the new events is stored in an event database.

The access control system used with the second approach was the Apache web server. In order to make as few modifications as possible to its existing access control policy evaluation mechanism, the ability to make and specify custom access control handlers for certain resources was utilized. Rather than returning a value for a specific attribute and querying against the event database within the access control handlers, the querying and analysis functions were abstracted into an external analysis server that provides risk analysis as a service. Requesting access control systems (such as the Apache web server implementation) submit requests to the analysis server specifying the type of desired risk analysis (source, target or system) and the attributes of the entity which the analysis should center around (in the case of the source and target analyses). Based on the risk assessment returned and the risk threshold that is assigned to that particular resource or action a permit or deny decision is returned.

Source restriction implementation. An excerpt from the httpd.conf file for the webserver that establishes the access control handler for restricting source permissions is shown in Figure 6-3. This directive establishes the module "SourcePermissionRestrict" as an access control handler. This module implements the attack response of restricting source permission. In this particular example five different levels of granularity are

established. Action "A1" is the least tolerant of risk: a threshold of 26 is set for the source risk, above which, requests will be denied. The other actions are progressively more risk-tolerant. The final threshold "SourceLockoutThreshold" establishes that a source will be blocked from all actions on all objects when its source risk level exceeds 41. The corresponding pseudocode for the handler is shown in Figure 6-4.

The processing steps for the source restriction and target restriction handlers are relatively the same, summarized in the following steps:

1. The properties of the request (subject and object of the request and the action being performed) are extracted from the URL and the request properties.
2. A request to the risk analysis server is generated specifying a) which type of analysis data is required and b) the identifier for the subject or object of the request
3. Once the risk value is returned, it is compared with the threshold(s) specified in the configuration file to determine if the request should be denied.
4. If none of the thresholds are violated, the request is permitted.

Force authentication implementation. The policy configuration for the access control module to force authentication is shown in Figure 6-5. The authentication module was actually written as a content handler, because the Authentication handlers are somewhat restricted and would not allow for the type of random challenge authentication that was desired in this case. The example shown establishes three independent thresholds, any of which could be used to trigger authentication for the requesting source. The corresponding pseudocode for the authentication module is shown in Figure 6-6. The system threshold is higher to limit the number of authentication requests that are necessary when the risk for a particular source or target is not yet at a suspicious level. It also, however, offers protection for as-yet untouched resources when the majority of intrusive traffic is concentrated elsewhere in the system. The analysis server receives requests and then loads the appropriate risk analysis module, dynamically generating queries to the event database to select the appropriate events. The risk module then generates the risk measure which is returned to the service requester.

6.6 Updates and Modifications to the Initial Model and Architecture

6.6.1 Performance Issues With the Initial Architecture

The problem with the initial architecture was the fact that the events were being aggregated on the demand of the client, and each time a request was made all of the related events were being re-examined and having risk values re-calculated based on a new decay factor that accounted for the accurate time difference between the request and the time the event actually occurred. As the number of events that were stored in the event database and needed to be analyzed as part of the aggregate set increased, the time to process each request was also increasing.

6.6.2 Solution 1: Caching

The scheme for caching of risk data relied on the creation of a background thread to pre-fetch risk data. The goal was to eliminate the increased response time problem described previously. Under the caching scheme, the analysis server creates a background thread that continually extracts all of the sources and targets from the event database. It then evokes the risk handlers on each of the entities extracted, producing source, target and system level risk where appropriate and storing the data back into the event database. When the analysis server receives a request for a risk evaluation on a system entity, instead of invoking the handler for that individual request, it performs a lookup on the cache tables in the event database.

Although the caching approach addressed the issue of the response time between the part of the analysis server that services requests and the data consumer, it also introduced another issue: the timeliness of the data that is returned to the consumer. The increasing time taken to execute a caching run that re-calculates the risk values for system entities is approximately the same as the increasing response time from the server. In testing, after a significant number of requests, the time taken to perform a full refresh of the risk cache for all of the entities became prohibitively long, such that requests from consumers for risk information would receive outdated, inaccurate information. In essence, the caching

approach did not address the complexity of the algorithm for risk calculation approach which was the primary reason for the increasing response time - it merely detached the calculation of an updated risk assessment from the fulfillment of a client data request. This analysis led to the development of a second implementation approach to address these shortcomings.

6.6.3 Solution 2: Redesigning the Analysis Algorithm and Refactoring the Architecture

The second implementation approach provides more efficient analysis of events - each event is seen and analyzed only once, when it first arrives. Risk assessments for affected entities are updated when a new alert arrives, and the function of the analysis server is to pull the stored assessment (not calculate it as before) and return the result to the requesting client.

6.6.4 Revised Risk Assessment Model

The risk magnitude assigned to a vulnerability exploitation attempt is still the exponential average of all of the magnitudes of all of the vulnerabilities referenced in the alert. The algorithm for calculating risk based on multiple events is no longer iterative, but recursive. The decay function has been removed so that the weight of each request does not need to be re-assessed. Instead, the ε value serves to weight the previous risk assessment for the entity with respect to the risk assessment for the newest event. This serves the same function of decreasing the influence of older data in favor of newer data, but does so as triggered by new events, and not merely a uniform time dependency. This also accommodates better assessing risk to entities with vastly different request frequencies.

$$R(r_j) = ln(\sum_{v_k \epsilon V(r_j)} e^{(w_x * SS(v_k))}) \tag{6–5}$$

$$TR(t_i, r_{t+1}) = ln(\varepsilon * e^{TR(t_i, r_t)} + R(r_{t+1})) \tag{6–6}$$

$$SR(s_i, r_{t+1}) = ln(\varepsilon * e^{SR(s_i, r_t)} + R(r_{t+1})) \quad (6\text{–}7)$$

The set $V(r_j)$, with members v_k is the set of all vulnerability exploitation signatures triggered by the request r_j. $SS(v_k)$ is the magnitude of the vulnerability v_k. $TR(t_i, r_{t+1})$ is the risk assessed to the target t_i as a result of intrusive request r_{t+1} . $SR(s_i, r_{t+1})$ is the risk assessed to the source s_i as a result of intrusive request r_{t+1} .

6.6.5 Restructured Architecture

The primary change to the architecture is the distribution of analysis functions between the alert server (responsible for context acquisition) and the analysis server. In the previous approach, the alert server only received alerts, extracted the necessary information, and stored them in the event database. The analysis server was responsible for receiving client requests for risk data, performing the required analysis operations and the sending a response to the client. The change in the risk model however, demands that the updating of risk information occurs as new events are processed. This requires that the primary analysis function (updating risk values for entities) occurs as the events are processed (and consequently must be performed by the alert server).

Another change, used to facilitate the preservation of the incoming analysis data, was to queue incoming alerts in the database until they could be processed by one of the available processing threads. This allowed the number of active alert processing threads to be decreased so that the total processing time for each alert would be less.

6.7 Summary

A system implementation is detailed based on the general framework previously discussed and satisfying the key design goals outlined under the acquisition, analysis and application of context information. A federated approach is used that follows the principles discussed in Section 3.2.2 and adapting the correlation ontology of Section 3.4.5 to a relational database model that serves as the schema for the event database described in Section 6.5. An independent analysis server performs the functions of aggregating

context information and then disseminating it. The implementation focuses on one of the context properties defined in the Chapter 4 ontology of assessment properties - that of risk. A model is defined for deriving risk information from assessments produced by an intrusion detection system on attempts to exploit software vulnerabilities. This model designates how risk is assigned to both subjects and objects of access requests. Details are also provided on how the analysis data is applied and used in the access control process. Context-based policy evaluation (previously discussed in Section 5.2) is achieved by providing custom access control handlers that interact with the analysis server to receive risk information. These access control handlers also implement the response techniques discussed in Section 5.3.

Figure 6-1. Sample risk progression for an intruder executing intrusive requests of moderate severity (5)

Request Number	Risk Estimation	Number of Previous Requests
1	0	0
2	25.65	1
3	32.19	2
4	36.11	3
5	38.92	4
6	41.11	5
7	42.91	6
8	44.43	7
9	45.75	8
10	46.92	9
11	47.96	10
12	49.77	11
13	50.57	12
14	51.99	13
15	53.23	14
16	53.8	15
17	54.34	16
18	54.85	17
19	55.34	18
20	55.8	19

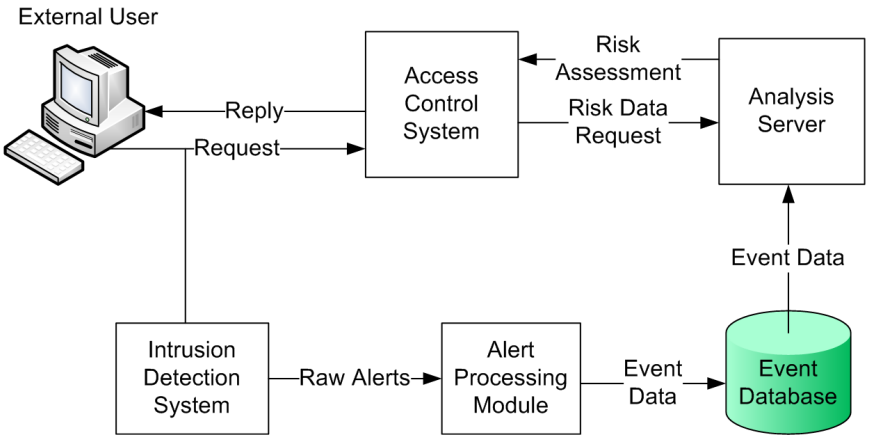


Figure 6-2. Architecture for the ABACUS framework.

```
<Location /s>
PerlAccessHandler SourcePermissionRestrict
PerlSetVar Action_A1_RiskThreshold 26
PerlSetVar Action_A2_RiskThreshold 32
PerlSetVar Action_A3_RiskThreshold 36
PerlSetVar Action_A4_RiskThreshold 39
PerlSetVar SourceLockoutThreshold 41
</Location>
```

Figure 6-3. Apache configuration directive that establishes a *SourcePermissionRestrict* access handler to evaluate all requests to resources in the directory '/s'. The directive also establishes four risk thresholds, each for a different action. These thresholds are subsequently used by the access handler to compare against the current risk evaluation for the source of the request, with the request being denied if the source's risk exceeds the threshold. The final variable *SourceLockoutThreshold* establishes that one the risk attached to the source exceeds 41, all requests from that source will be denied.

```
read threshold_values;
read request_properties;
request source_risk from analysis server;
set response = OK;
if(source_risk > lockout_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A1 AND source_risk > A1_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A2 AND source_risk > A2_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A3 AND source_risk > A3_threshold)
{ response = DENY_REQUEST; }
else if(request_action == A4 AND source_risk > A4_threshold)
{ response = DENY_REQUEST; }
return response;
```

Figure 6-4. Psuedocode for the access control model that performs restriction of source permissions based on a risk assessment obtained from an analysis server. It retrieves a risk assessment for the source from the analysis server and then compares it with the appropriate threshold for the action being performed.

```
<Location /sc3>
SetHandler perl-script PerlHandler AuthChain
PerlSetVar SystemRiskThreshold 55
PerlSetVar SourceRiskThreshold 33
PerlSetVar TargetRiskThreshold 45
PerlSetVar AuthExpiration 300000
</Location>
```

Figure 6-5. Apache configuration directive for a custom authentication handler. Three different thresholds, or properties are established which could be used to trigger the use of authentication. A value is also set for *AuthExpiration* which ensures that, one authenticated, users are only re-authenticated every 300 second (five minutes) at most.

```
read threshold_values;
read request_properties;
request source_risk from analysis_server;
request target_risk from analysis_server;
request system_risk from analysis_server;
if(source_risk >source_threshold OR target_risk > target_threshold OR system_risk >
system_threshold)
{
send authentication_request;
if(credentials_incorrect)
{ return AUTHENTICATION_REQUIRED; }
else
{ return AUTHENTICATION_GRANTED; }
}
else { return NO_AUTHENTICATION_REQUIRED; }
```

Figure 6-6. Pseudo code for authentication module. Authentication is required if any of the established risk thresholds are exceeded.

CHAPTER 7
RESULTS

7.1 Testing Setup

Hardware setup. All of the testing to be discussed was performed using two identical Linux virtual images each running Ubuntu Linux 8.04 with a 1.86 Ghz processor and 1 GB of RAM. One machine served as the server and the other as the client or traffic generation node. The server machine contained the web server, IDS, alert server, analysis server and event database mentioned in the architecture. For the purpose of testing and implementation, the web server used was Apache version 2.2.10. Snort version 2.8.1 was used as the intrusion detection system.The event database was supported by the MySQL DBMS version 5.0.51. Both the analysis and alert processing servers were written in Java. The access control modules for Apache were written in Perl using mod_perl 2.0.4.

7.2 Validation of Analysis Model

The first set of results pertain to the evaluation of the risk analysis model. The goal of this testing is to demonstrate the following:

1. that the essential assumption of the model - that of escalating risk - is valid for scenarios that involve successive, related intrusion attempts
2. that this assumption can be validated experimentally using real data sets
3. that various techniques exist, and can be used effectively, to deal with some of the problems regarding data quality including: false positives and false negatives

The data for the tests were from the the first of the two scenario-specific data sets provided by the Lincoln Laboratory [70]. The data set records a distributed denial of service attack and was divided into the following five phases: 1) an IPsweep of the target network from a remote site 2) a probe of live IP's to look for the sadmind daemon running on Solaris hosts 3) breakins via the sadmind vulnerability, both successful and unsuccessful on those hosts 4) installation of the trojan mstream DDoS software on three hosts in the target network and 5) launching the denial of service attack. Initial test results showed the intruder (as described in the provided labeling data) with the highest risk rating after

a majority of the attack had concluded. Unsatisfactorily, however, due to false positives early in the tests some other nodes were initially given higher risk ratings during the first phases of the attack. In addition, the overall number of nodes that were assessed as potential intruders was high. Two different alert filtering techniques were applied, in an effort to improve the data accuracy and reduce false positives. The first was to use a technique proposed in [71] to filter out alerts that do not correspond to the exploitation of a 'concrete vulnerability'. A concrete vulnerability is defined in this case as one which is listed in the CVE [72], a standardized database for software vulnerabilities. In order to compile a working database to check vulnerability signatures, the latest CVE entries were downloaded and stored in a relational database. The results for the second round of testing using the concrete vulnerability filtering are shown in Figure 7-1 on page 118.

The latter part of the risk progression is relatively flat because the intrusion detection system being used failed to detect some of the later events involved in the attack sequence. And while the risk model does not make provisions for detecting attacks which are missed by intrusion assessment mechanism, the use of historical data to assess the threat posed by the source at least ensures that the same risk level based on earlier behavior is maintained. In this way, the model is tolerant of missed detections.

The risk assessments in the second set of test results were still somewhat inaccurate; a number of nodes on the local network were rated as suspicious and up until approximately the 9th sampling iteration the actual intruder does not have the highest risk rating. A second alert filtering technique was used to further increase the accuracy of the assessment: configuration verification. This is similar to the approach of verify alerts using network knowledge as discussed in [73, 74]. In this case, a database was constructed with all of the known, labeled nodes in the data set, the operating system running on the node and its version of the operating system. Each time an alert was generated this database was consulted to see if the vulnerability being reported actually matched the configuration of the targeted machine. If there was no match, the alert was discarded. Using these two

filtering techniques in conjunction the risk assessment reflected the single-intruder nature of the data set, as shown in Figure 7-1.

After applying the filtering techniques, the results for target risk estimation were improved. Results for target risk estimation are shown in Figure 7-2 part A and Figure 7-2 part B. In the final risk estimation graph for targeted nodes (Figure 7-2 part B), only the nodes actually attacked are rated, and those nodes for which successful attacks are launched are rated with the highest risk values.

7.3 Web Server Attack Resistance Results

This second set of testing results is designed to demonstrate results of testing the second of the two architectures (the risk analysis server integrated with Apache) with real time incoming requests. In order to effectively illustrate the effect of the three chosen response techniques, three different scenarios were generated with a web server traffic simulator and requests were sent to two different web servers: one using the three analysis modules described previously, and another only using the notion of the global system threat to trigger response techniques. Whereas validation of the risk model could be performed with a captured data set being replayed over the network, the use of the response strategies will require active connections to the access control system and hence demands live traffic.

The traffic simulator creates an array of requesting nodes S where s_i is a member of S, each with an intrusiveness rating i_r, an inter-request period p and a total request life l. The web server is arranged as an array of target resources T (where t_i is a member of T). Each t_i has a set of valid actions a_1, a_2,.... a_n and invalid or intrusive actions i_1, i_2, ... i_k. Every p seconds (or a random number of seconds between 0 and p), request source s_i selects a member of T and then based on its intrusiveness rating, selects either a normal or intrusive action to perform on the resource. Sources with a higher i_r have a greater probability of selecting an intrusive action for each request. In practice, these intrusiveness or maliciousness ratings range from 0% to 90%.

For the risk analysis model, vulnerability weightings were the following: high severity ($w_H = 3$) , medium severity ($w_M = 2$) and low severity ($w_L = 1$). The risk multiplier (γ) was set to 10, to provide a more noticeable difference between various assessments.

Scenario 1: single intruder, vulnerability probing. In this first scenario, a single intruder executes intrusive requests on several system resources - a method indicative of probing for which vulnerabilities have been patched or which configuration holes have been closed. The rest of the sources generating system requests are normal users - executing little or no requests that could be categorized as intrusive. The requests were generated over the course of a three hour simulation. The request trace for the intruder demonstrates that requests for different actions are denied based on his overall risk profile and eventually the intruder is locked out from all system requests. Meanwhile, requests from the other users are still permitted. A summary of the results for a simulation of this scenario are presented in Table 7-1. Figure 7-4 charts the growth of the risk assessed to the intruder. In this scenario all of the intrusive requests were from the single intruder. Server 1 began to deny requests from the intruder after their source risk passed the threshold of 45. The normal requests blocked by server 1 were also from the intruder. Once the system risk for server 2 passes the threshold, it begins to deny requests from all sources. The configuration directives used in the two servers during this scenario are shown in Figure 7-3.

Scenario 2: multiple intruders, single target, many-to-one attack. In the second scenario, multiple intruders target the same resource with two different attacks. This could correspond to the publication of a new vulnerability for an existing service. In the interim period some non-intrusive requests are allowed on the resource, but when the target risk reaches the threshold, all requests to the target are denied. A summary of the results for a simulation of this scenario are presented in Table 7-2. The growth of the risk assessed to the target is charted in Figure 7-6.

The testing for scenario two demonstrates that using target risk when a particular resource is being targeted can increase the number of intrusive requests that are blocked while maintaining availability for the other system resources. During this simulation, both the system risk and the target risk for the targeted resource peaked at 83. This was due to the fact that all of the intrusive requests in the entire system were directed at the same resource. While the system risk threshold could have been raised to decrease the percentage of normal requests that were denied, it would have also increased the number of intrusive requests that were blocked. The configuration directives used in the two servers during this scenario are shown in Figure 7-5.

Scenario 3: multiple attackers on various resources. In the third scenario, multiple intruders attack multiple system resources. This could correspond to a system with high traffic levels that sees exploitation attempts on multiple resources from multiple sources in a given period of time. Using both source and target risk levels, requests at various points in the overall request trace are responded to by a request for authentication. Eventually when the system risk level passes the threshold, all initial requests are responded to by requests for authentication. A summary of the results for a simulation of this scenario are presented in Table 7-3. The simulation was run for approximately 2.5 hours with nodes generating requests at all levels of maliciousness (and thus there is no clear intruder).

Due to the use of source, target and system risk information, the policy for server one was stricter. Despite this, the proportion of non-intrusive requests that were responded to by a request for authentication was only four percent higher than for server two. This number of non-intrusive requests also includes requests from nodes with high maliciousness ratings such as 90%, which would otherwise be deemed intruders but were classified at non-intrusive because the particular request being classified was not intrusive. The configuration directives used during this scenario are shown in Figure 7-7. The policy used for the server within the ABACUS framework was more stringent than the server only

using system risk. The former used three different risk properties to trigger authentication, but only experienced a 3.9% increase in the number of non-intrusive requests.

The results for this scenario essentially represent the behavior of an authentication mechanism that has an immediate expiration of authentication credentials (after authentication) and thus re-authenticates for each request. In a simulation involving human users (who would be capable of successfully completing authentication), many of the subsequent authentication requests would be eliminated by a longer period before the expiration of the authentication. In such situations where a high percentage of the requests are being authenticated, the server could potentially cut overall response time by bypassing the request for analysis data from the analysis server (which typically dominates the length of the response) and just authenticating each request immediately. This would move the number of non-intrusive and intrusive requests authenticated to 100% when the system risk reaches a sufficient level. However, if the process of authentication requires more time than the request for analysis data it might still be slightly more efficient to eliminate some of the the authentication requests; in the end, this is highly dependent on the relative time required for each process.

7.4 Performance Analysis

7.4.1 Performance Testing Methodology

In order to compare the performance of the final version of the Abacus framework against the earlier version and also against a normal Apache web server, each server was stress-tested. This part of the testing relied on a regression testing and benchmarking utility called Siege[75]. The basic aim of this testing was to examine the behavior of each server subject to increasing load.The following parameters were used in the testing process:

- Number of clients - with the use of a wrapper for Siege called Bombard, the user is able to specify an initial number of clients an increment of how many clients the load

should be increased by for each iteration and a total number of iterations (which also limits the maximum number of clients)

- A set of URLs - the same URLs from the scenario testing were used (both normal and intrusive). They were placed in a configuration file and read into memory by the utility when it starts. The clients then randomly request one of the URLs in the file for each request.
- Delay between requests - before each request, the client waits a random number of seconds between 0 and d, where d is the maximum delay between requests specified by the user

By testing in this way, we hope to draw conclusions on the following: the degree of improvement provided by the third iteration of the Abacus framework over the first, the point at which each of the server types become overwhelmed given the hardware constraints as well as the specific reasons that account for the performance differences.

7.4.2 Performance of Initial Abacus Framework

In Figure 7-8 the time to serve requests on Server 1 is shown as the number of requests increases. For the collection of this data, the simulator was set to generate 3 hours of traffic from 10 different nodes, only one of them executing intrusive requests (scenario one as described above). In part a of the figure the time to serve is shown for all of the requests. In part b, only the time to serve requests from the intruder is shown - this graph has the same linearly increasing pattern that is apparent when looking at the peaks of the graph in part a. In part c of Figure 7-8 the time taken to serve requests from the non-intruder nodes is graphed. The time to serve these requests remained relatively constant throughout the entire simulation, oscillating between zero and one seconds. The reason that the increase only occur ed for the intruder node is that when the web server requests risk data on that node, there is a constantly increasing amount of event data to analyze. For the other nodes, there is no such increase of data to analyze and, as a result, requests are served in the same amount of time for the duration of the simulation. This

is undesirable, however, and could potentially create a scalability issue in scenarios where there are more nodes with intrusive behavior. In order to ameliorate these performance issues, a caching scheme was devised to facilitate faster generation of risk data.

7.4.3 Performance of Abacus Framework with Recursive Analysis Model

After adapting the analysis model as described in 6.6 to be recursive instead of the previous iterative formulation and modifying the alert server to make the updates for the risk values as new data came in, the performance of the framework was improved significantly. Whereas the first version was not stable with ten concurrent users, the modifications allowed the framework to handle up to 100 concurrent users stably for an indefinite period of time. This demonstrates that the modifications made to the model enable the framework to process incoming requests without noticing an increase in response time for increased amounts of data, which was the problem in the previous version. The graphs in Figures 7-9 and 7-10 summarize the performance of the different aspects of the framework. The web server and analysis server experienced some local spikes based on the short delay between subsequent requests from the stress testing application, but overall maintained a consistent response time. The alert server (see graphs in Figure 7-10) experienced an initial spike in processing time due to provisioning new threads to process the high volume of incoming alerts. Performance stabilized quickly and remained stable throughout the remainder of the simulation.

7.4.4 Performance Comparison for ABACUS Framework and Ordinary Apache Web server

Results. Figures 7-12 and 7-13 summarize the server response time, concurrency and transaction rate as seen from the client for three different server types: a normal Apache web server, with no integration of risk information, an Apache server integrated with the first version of the analysis framework as discussed above (Abacus Server version one), and an Apache server integrated with the final version of the analysis framework (Abacus Server version two). The range of delay between requests differs between the two figures:

in Figure 7-12 all testing threads were configured to delay between 0 and 1 second before initiating another request. In Figure 7-13 the range was between 0 and 10 seconds.

Figure 7-12 indicates that the Abacus Framework version two was able to serve 100 simulated clients with a response time of 5.16 compared to 1.73 seconds for the unmodified Apache web server. At this load the Abacus Framework was maintaining the request frequency without a noticeable increase in processing time during the duration of the test, as demonstrated by Figure 7-9. Figure 7-13 (where the delay for the stress testing simulation was between 0 and 10 seconds for subsequent requests) places the maximum number of clients at 210 with a response time of 6.35 before the response time spiked at the next increment of clients. What is consistent in both figures, however, is the transaction rate or the average number of connections processed per second. With the delay between zero and one, the maximum transaction rate was 18.76 and with the delay between zero and ten, the maximum was 18. A few individual simulations at higher numbers of connecting clients were run and the results are shown in Figure 7-11. These figures, particularly parts B and C demonstrate that the increasing response time generates more timeouts at higher concurrency, because there is also a higher rate of incoming connections. Part C in particular, where 200 simulated clients were used, shows the effect on response time from a large number of request timeouts.

Discussion. It would be difficult to say that the first version of the framework (before the caching approach) could realistically support any number of users for an extended period of time. As shown in Figure 7-8 part A, the time to serve requests for the first version of the server was increasing even when the number of simulated clients was held constant at ten. The performance determinant for the first version of the framework was the number of requests: increasing the number of simulated clients just caused the number of requests to increase more rapidly. The time to service each request was linear in the number of requests that the server had received up to that point. The caching

approach allowed for a constant time to serve each request, but at the expense of data accuracy, where the algorithmic complexity was still the same.

It is likely that the resources of the server machine were exhausted when the tests moved to higher numbers of simulated clients (110 for the higher frequency tests) than the server transaction rate could handle and new connections were still coming in resulting in queuing. The Apache web server limits the number of forked client processes to 256 by default (this limit is compiled into the software). It appears based on the data that at the failure points, when the response time spikes, the server resources were exhausted (before the limit of 256 client processes) by the increasing number of forked client processes being created by the Apache server. During testing, this led to incidences where the server machine locked up and required restarting (when testing with both the ABACUS framework and with the unmodified Apache server). This is corroborated by the fact that the unmodified Apache web server failed in a similar way (albeit at a higher transaction rate).

A rate limiting mechanism was built into the Abacus Framework v3 whereby once a certain number of requests are queued, the server begins to deny incoming requests until more worker processes become available (to avoid forking too many processes to serve requests). The Apache server access logs during these tests demonstrate that some of the requests for analysis data from the web server access control modules were being denied to due increasing load; at the same time, however, the Apache web server was still accepting and queuing new client connections. In summary, the testing failure of the Abacus Framework was due to the difficulty in controlling server resources: in particular of effectively limiting the incoming client connections in the face of increased concurrency and therefore increased response time per request. A more robust set of testing conditions would likely yield better results.

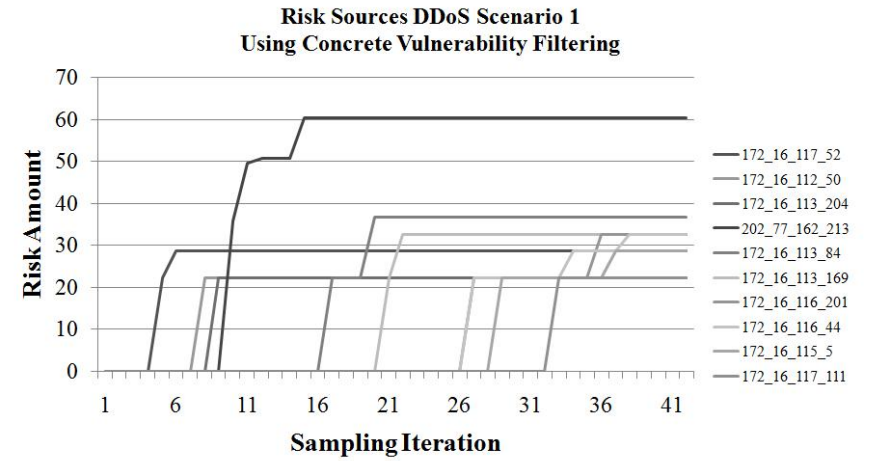
With that said, the peak transaction rate for the Abacus Framework v3 was still 15.53 transactions per second at a response rate of 5.73 seconds. This roughly equates to 931.8

transactions a minute, 55,908 transactions an hour and 1,341,792 transactions per day. By way of comparison, according to Compete.com [76] web statistics, Facebook.com received 874,806,456 page visits in December 2008 with an average of 62.1 pages accessed per visit for a total of 54,325,480,917.6 page access in December, or 1,810,849,363.92 page accesses each day. The UFL.EDU domain (and all of its subdomains) received 2,385,137 visits in the month of December 2008, with an average of 17.5 pages per visit or 41,739,897.5 page views in that month. This equates to 1,391,329.9 page views per day. Table 7-4 summarizes statistics for three top web sites, while Table 7-5 provides an estimate for the peak performance of the ABACUS framework under current testing conditions. Based on this data, we can can conclude that the proposed approach could be implemented in a large, high traffic website - particularly with dedicated server hardware with increased performance.

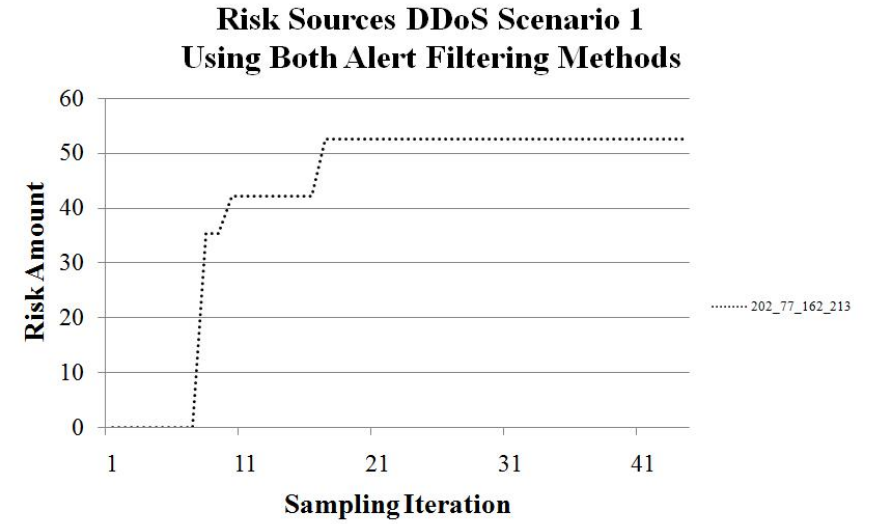
The data also demonstrates that failure of a similar nature occurs for the Apache web server in isolation. Because there was a slower growth in response time per request, the Apache server in isolation was able to handle a greater number of client connections before failure, but when the failure happened, it manifested with much the same behavior as was displayed when testing the final version of the Abacus Framework.

Figure 7-14 represents the factor of increase in the response time for the web server in the ABACUS framework compared with the response time of the normal Apache web server. After an initial spike in request time due to provisioning of server resources, the response time increase factor stabilizes at approximately three, meaning that during the majority of the testing period the average request to the ABACUS framework took three times as long to process as a request to a normal Apache web server. The figure also shows a sharp increase at 110 simulated clients which is where the simulation first recorded a significant number of timeouts for the ABACUS framework, but where the normal Apache web server remained stable. This increase factor is likely a result of the following additional steps taken before a response is generated in the framework:

the access control handler generating a request for risk information, the analysis server performing the necessary queries to the event database and formulating a response back to the web server. In addition, while the analysis server and web server were running, some of the system resources were consistently being consumed by the server responsible for receiving and filtering IDS information (which was not running along with the stand alone web server).



A



B

Figure 7-1. Simulation results from the validation of the analysis model showing risk estimates for the sources detected as intrusive. A) using only concrete vulnerability filtering B) using concrete vulnerability filtering and configuration verification.

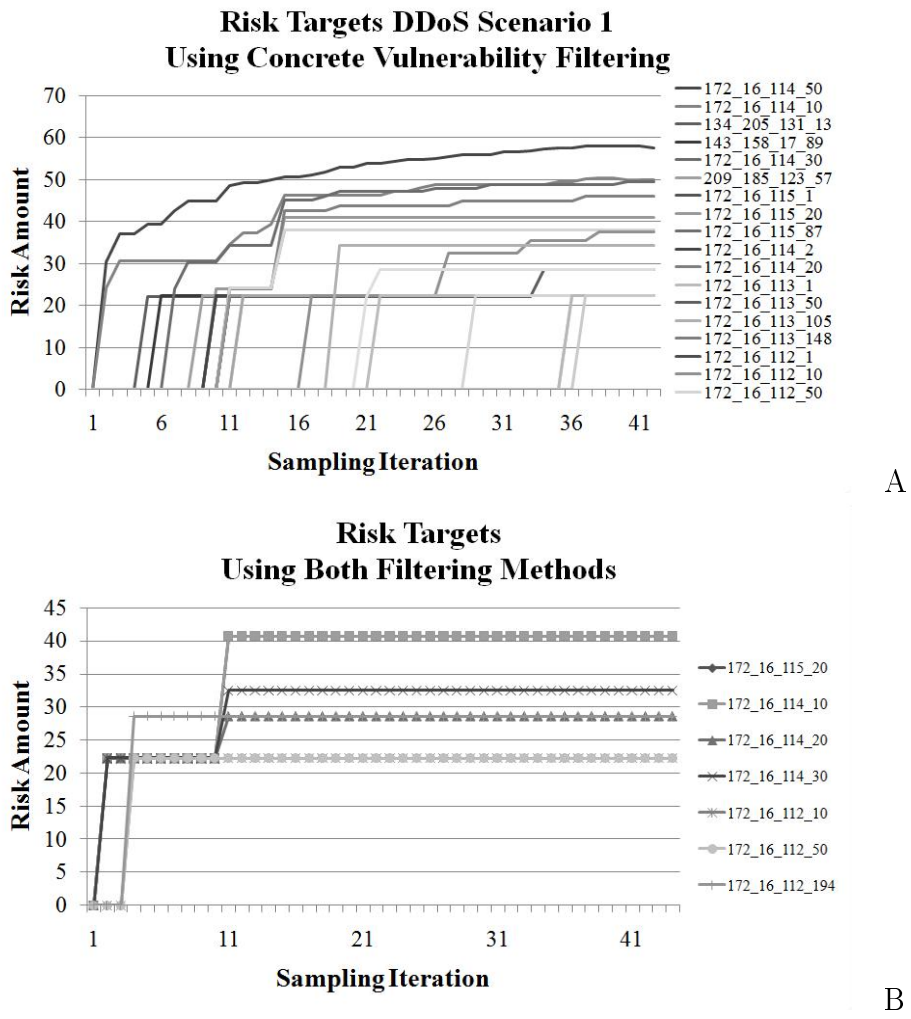


Figure 7-2. Simulation results from the validation of the analysis model showing risk estimates for targets being attacked by intrusive requests. A) using only concrete vulnerability filtering B) using concrete vulnerability filtering and configuration verification.

Table 7-1. A summary of the simulation results for scenario one simulating an attack from a single source on multiple system resources.

Property measured	Server 1 (source risk)	Server 2 (system risk)
Total requests	2472	2472
Total intrusive requests	230	230
Intrusive requests denied	229	179
Percentage denied	99.5%	77.8%
Total normal requests	2242	2242
Normal requests denied	16	1751
Percentage denied	.7%	78.1%

```
<Location /sc2>
SetHandler perl-script AccessHandler SystemPermissionRestrict
PerlSetVar SystemRiskThreshold 65
</Location>
```

A

```
<Location /sc2>
SetHandler perl-script AccessHandler SourcePermissionRestrict
PerlSetVar TargetRiskThreshold 45
</Location>
```

B

Figure 7-3. Access control policies for the two servers during scenario one while simulating an attack from a single source on multiple system resources. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses source risk data and sets a threshold of 45 for the source risk, beyond which, requests from that source will be denied.

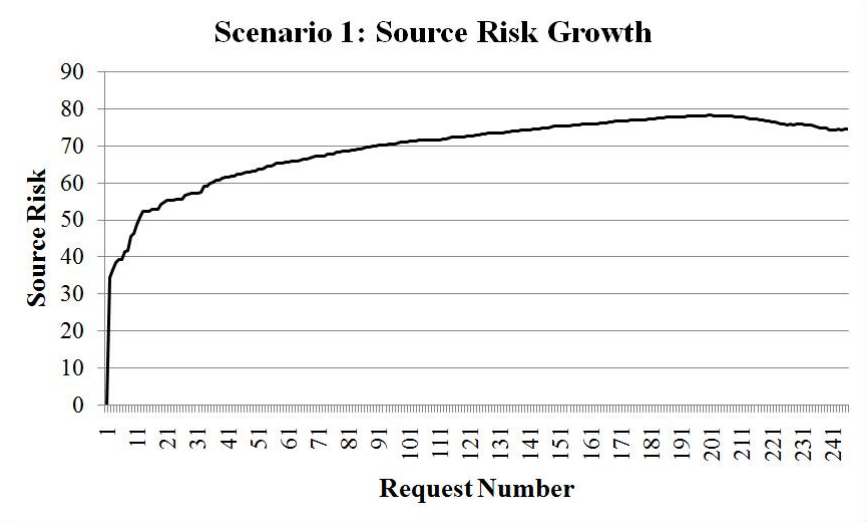


Figure 7-4. The growth of the risk from the intruder in scenario one.

Statistic	System 1 (target risk)	System 2 (system risk)
Total requests	1023	1023
Total intrusive requests	320	320
Intrusive requests blocked	319	274
Percentage denied	93.5%	85.6%
Total normal requests	703	703
Normal requests denied	65	588
Percentage denied	9.2%	83.6%

Table 7-2. A summary of the simulation results from scenario two while simulating an attack from multiple sources on a single system resource.

```
<Location /sc2>
SetHandler perl-script AccessHandler SystemPermissionRestrict
PerlSetVar SystemRiskThreshold 65
</Location>
```

A

```
<Location /sc2>
SetHandler perl-script AccessHandler TargetPermissionRestrict
PerlSetVar TargetRiskThreshold 45
</Location>
```

B

Figure 7-5. Access control policies for the two servers during scenario two while simulating an attack from multiple sources on a single system resource. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses target risk data and sets a threshold of 45 for the target risk, beyond which, requests to that target will be denied.

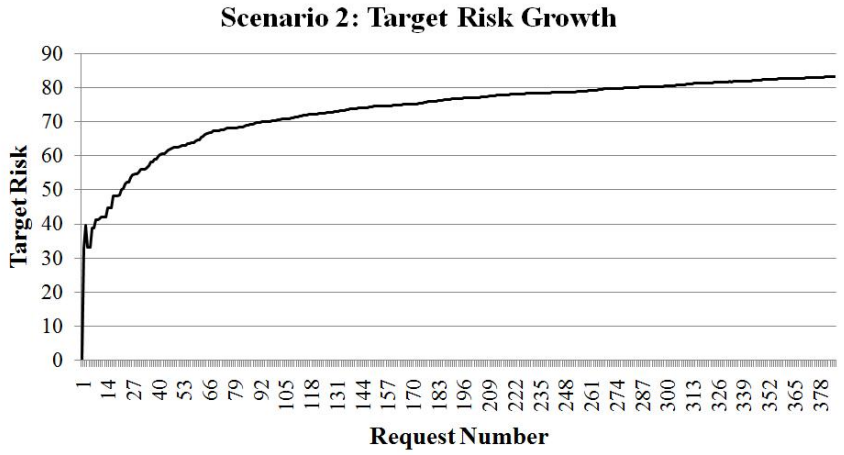


Figure 7-6. The growth of risk for the targeted resource in scenario two.

Table 7-3. A summary of the simulation results from scenario three while simulating an attack from multiple sources on multiple system resources.

Statistic	Server 1	Server 2
Total requests received	875	875
Total intrusive requests	437	437
Intrusive requests authenticated	409	252
Percentage authenticated	93.5%	57.7%
Total non-intrusive requests	438	438
Non-intrusive requests authenticated	385	368
Percentage authenticated	87.9%	84%

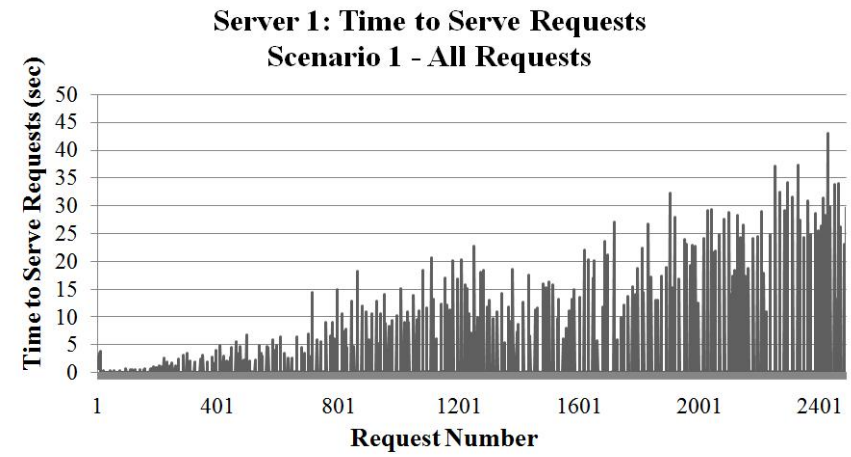
```
<Location /sc3>
SetHandler perl-script AccessHandler SystemPermissionRestrict
PerlSetVar SystemRiskThreshold 65
</Location>
```

A

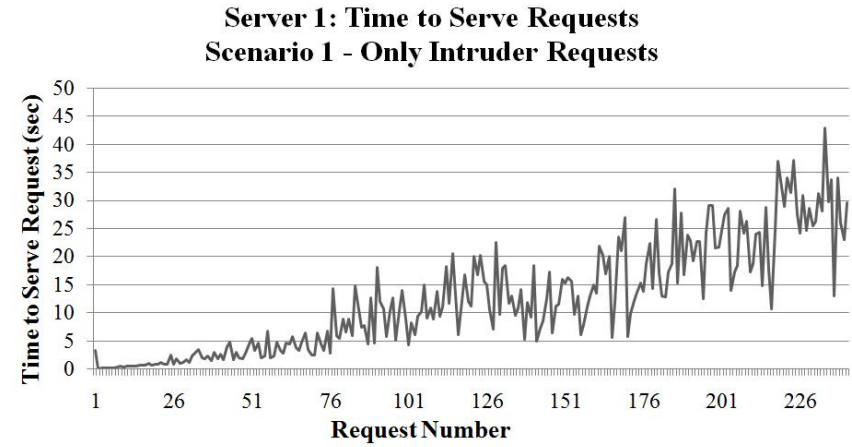
```
<Location /sc3>
SetHandler perl-script PerlHandler AuthChain
PerlSetVar SystemRiskThreshold 65
PerlSetVar SourceRiskThreshold 33
PerlSetVar TargetRiskThreshold 45
PerlSetVar AuthExpiration 300000
</Location>
```

B

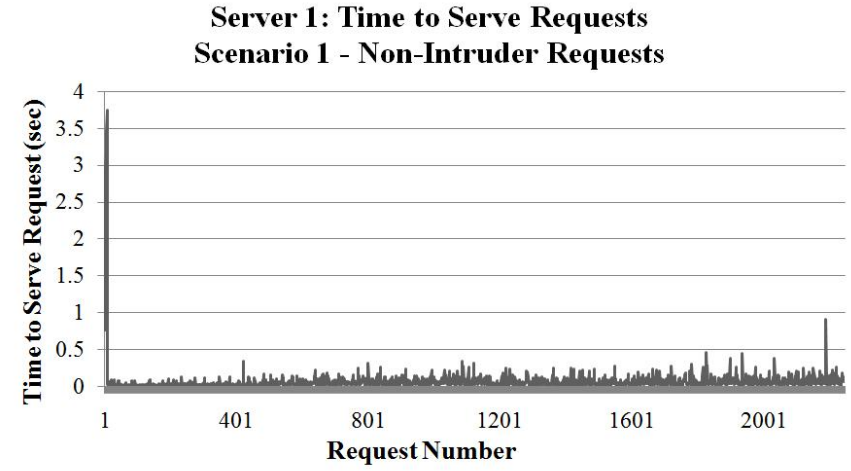
Figure 7-7. Access control policies for the two servers during scenario three while simulating an attack from multiple sources on multiple system resources. A) server two B) server one. The first policy (A) establishes an access handler that uses system level risk data and sets a threshold of 65 for the system risk, beyond which, requests will be denied. The second policy (B) establishes an access handler that uses three different risk properties to trigger the requirement of authentication. The system risk threshold is 65, the source risk threshold is 33 and the target risk threshold is 45. A time limit for the expiration of a valid authentication is set at 300 seconds using the *AuthExpiration* property.



A

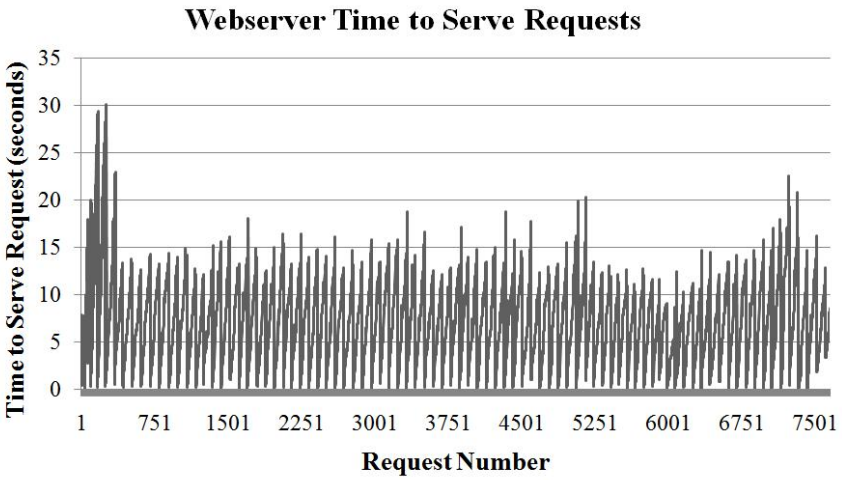


B

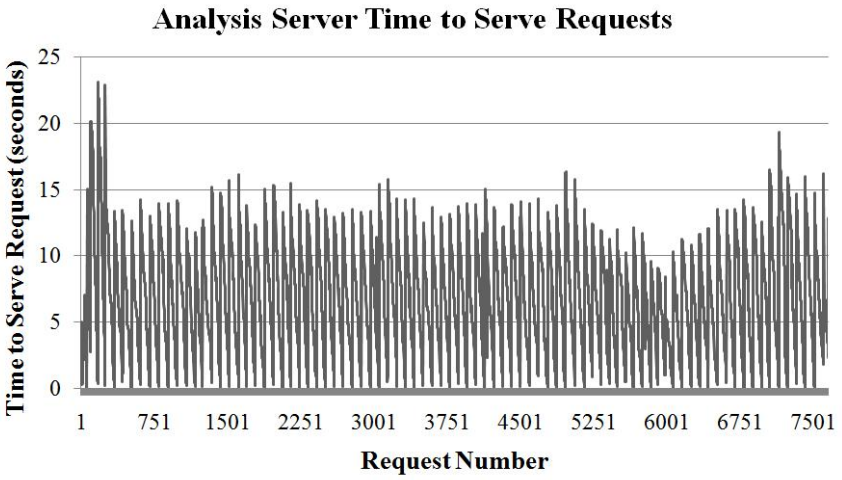


C

Figure 7-8. Statistics for ABACUS framework version 1 during a simulation with ten concurrent users, one of which was an intruder. Graphs show time to serve requests for different breakdowns of the set of requesting users. A) requests from all users B) requests from the intruder C) requests from non-intrusive users. These graphs establish that the time to process requests was increasing throughout the simulation and that this was due to the increased time in took to process requests from the intruder that required more data to be aggregated and analyzed in order to produce a risk assessment.

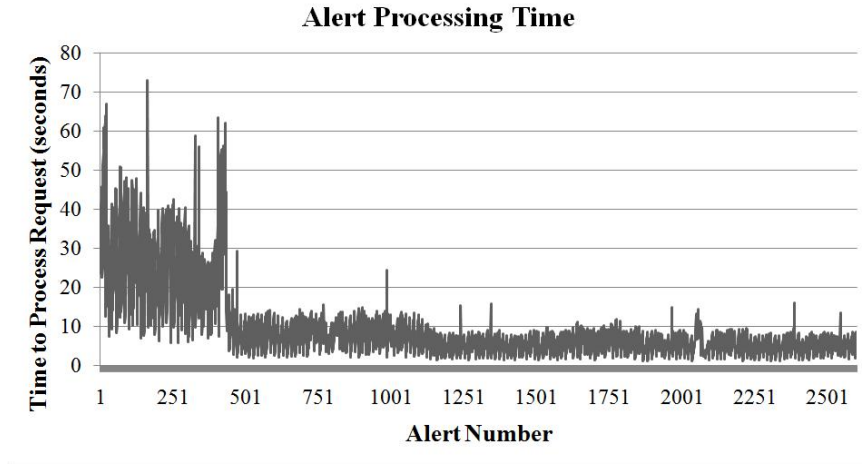


A

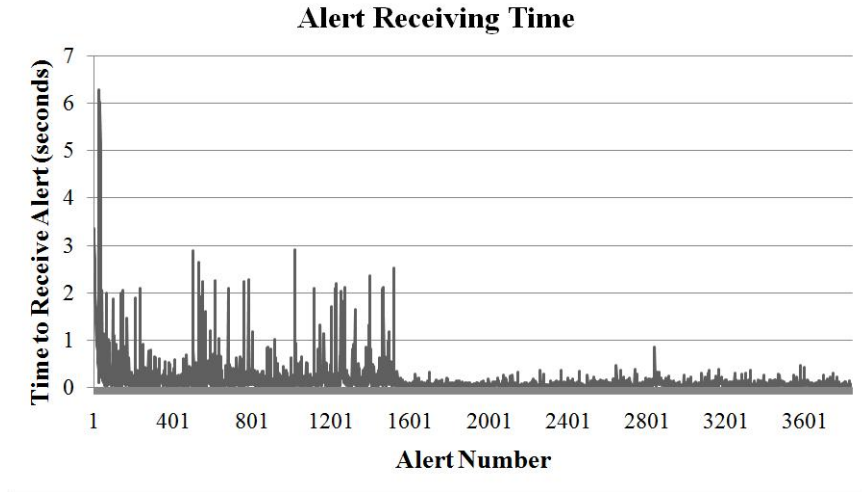


B

Figure 7-9. Statistics for ABACUS framework version two. A) time to serve requests for the web server B) time to serve requests for the analysis server. The graphs correspond to a simulation with 100 concurrent users for the entire duration of the test 10 minute stress test.

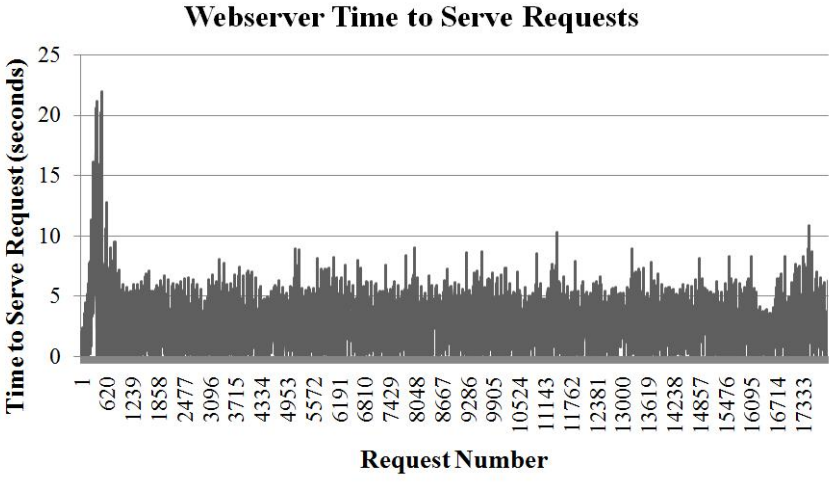


A

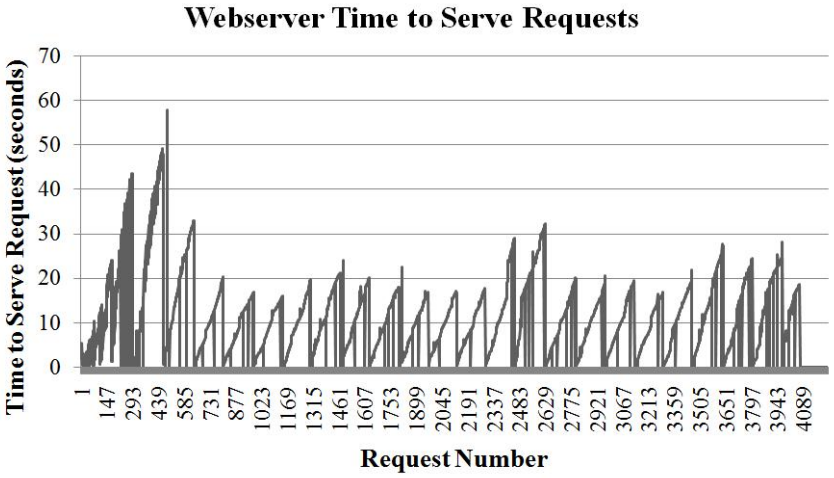


B

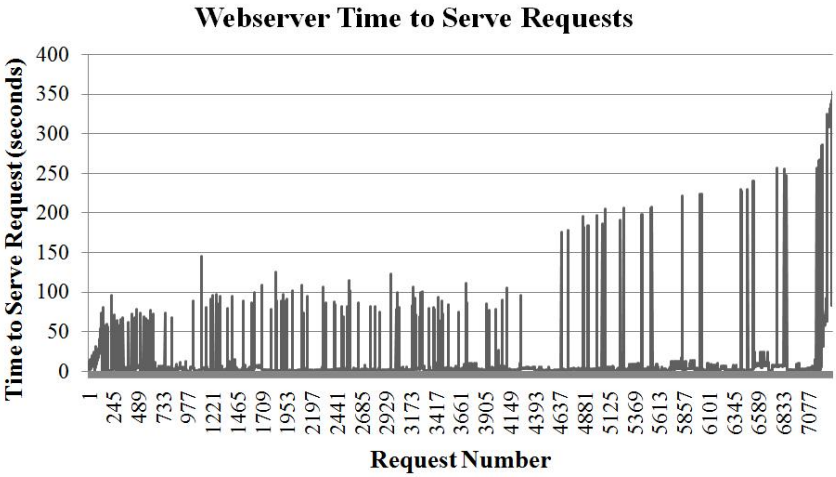
Figure 7-10. Statistics for ABACUS framework version two. A) alert processing time B) alert receiving time. The graphs correspond to a simulation with 100 concurrent users for the entire duration of the test 10 minute stress test.



A

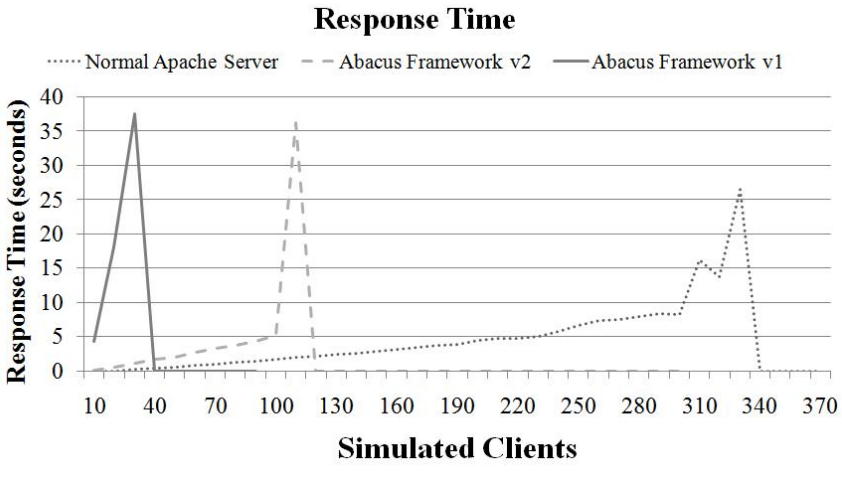


B

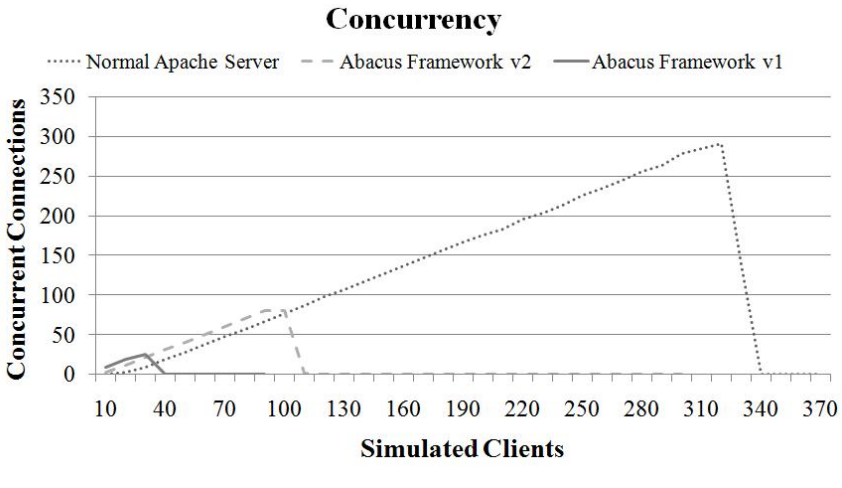


C

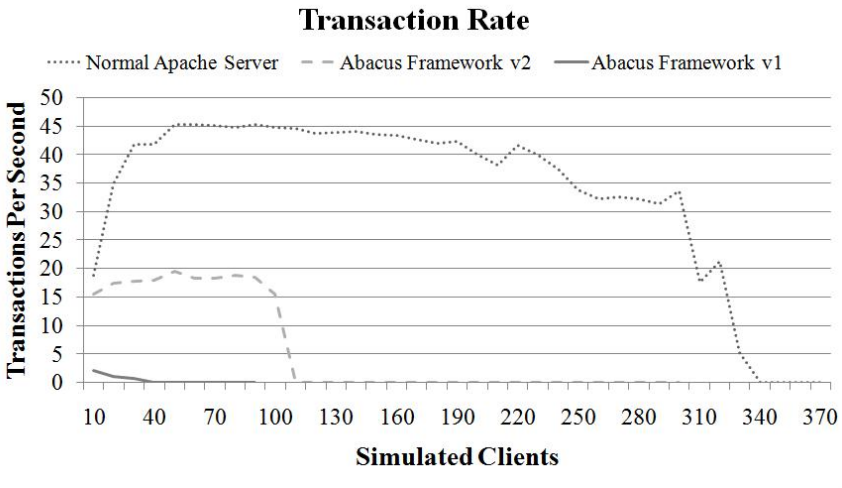
Figure 7-11. Additional stress test statistics for ABACUS framework version two. A) using 110 concurrent clients B) using 175 concurrent clients C) using 200 concurrent clients.



A

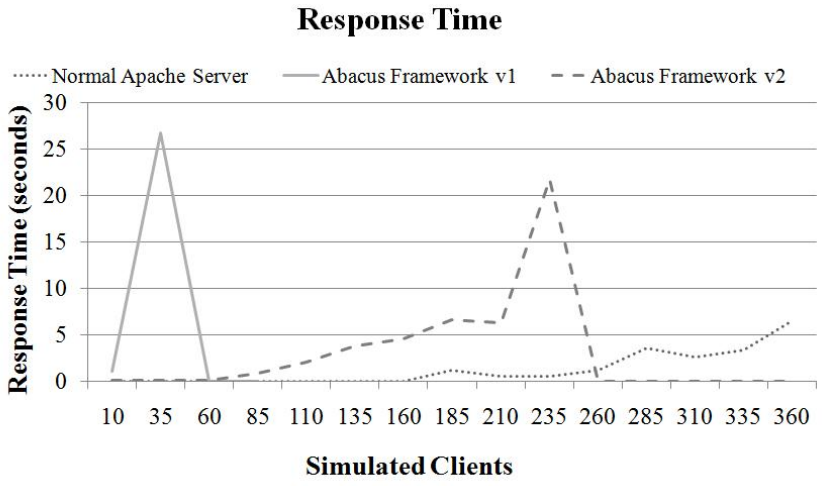


B

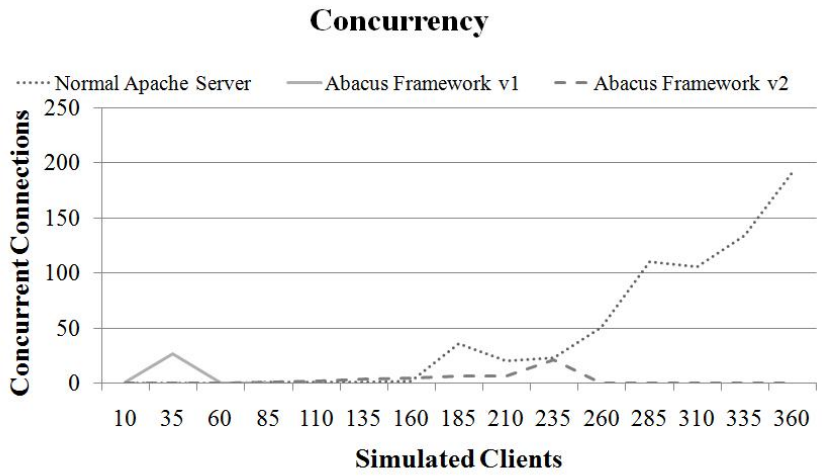


C

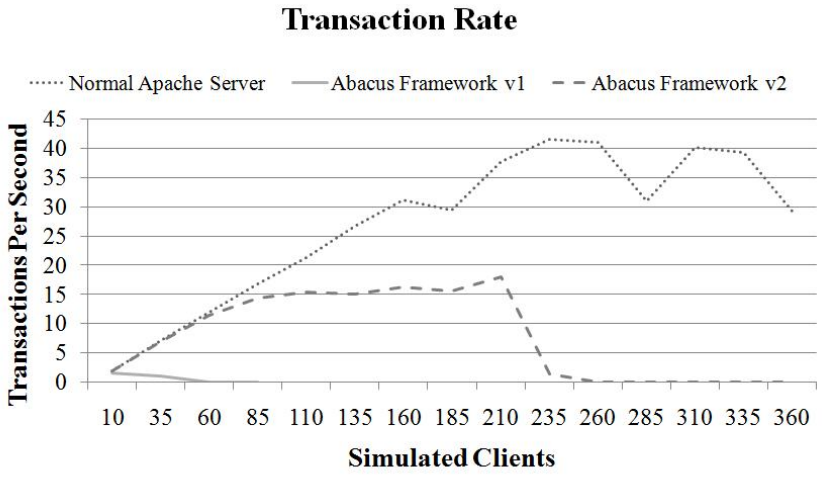
Figure 7-12. Web server comparison using a randomized delay from 0 and 1 second between requests. A) response time B) concurrency C) transaction rate



A



B



C

Figure 7-13. Web server comparison using a randomized delay from 0 and 10 seconds between requests. A) response time B) concurrency C) transaction rate.

Table 7-4. Traffic statistics for three top websites in December 2008.

Domain	Visits/Month	Pages/Visit	Page Views/Month	Page Views/Day
Yahoo.com	2,211,018,102	19.4	42,893,751,178.8	1,429,791,705.96
Facebook.com	874,806,456	62.1	54,325,480,917.6	1,810,849,363.92
Ufl.edu	2,385,137	17.5	41,739,897.5	1,391,329.9

Table 7-5. Estimated peak performance for ABACUS framework with current testing constraints.

Transactions/Sec	Response Rate (sec)	Transactions/Hour	Transactions/Day
15.53	5.73	55,908	1,341,792

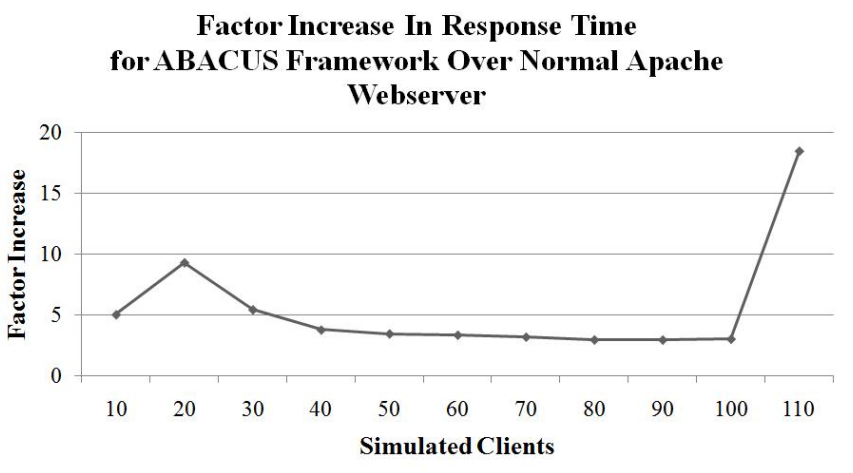


Figure 7-14. Summary of the factor increase in web server response time for the ABACUS framework version two compared to the performance of an unmodified web server.

CHAPTER 8
CONCLUSIONS

The aim of this study was primarily twofold: firstly, to demonstrate a cohesive, general approach to designing and constructing context-aware or adaptive security mechanisms and secondly to demonstrate the application of those principles by designing such a system and demonstrating its feasibility and effectiveness.

8.1 Conclusions Produced By Examination of the General Approach

The design decision was made that the nebulous concept of context awareness was best evidenced by context-aware behavior, or behavior to demonstrates an awareness of changing system state. The process of making a decision aware of context was abstracted into three phases: acquiring that data, analyzing it and applying it in the decision making process.

8.1.1 Data Acquisition

Acquisition was approached as an integration issue, particular with regards to the integration of security mechanisms that often exhibit high degrees of autonomy, heterogeneity and distribution. The trade-offs between two different integration approaches were discussed at length. While a vertical integration strategy provides a tight and seamless integration, it is also very difficult to extend. The three factors mentioned previously (autonomy, heterogeneity and distribution) are essentially dealt with by merging the distinct mechanisms into one. Horizontal integration presents many challenges and requires the use of individual integration techniques to deal with data, control and process integration. The result, however, is a highly extendable system.

8.1.2 Data Analysis

One of the primary conclusions regarding data analysis was that this phase must produce concrete, quantifiable, actionable data. This conclusion seems intuitive, but contrasts starkly with many of the efforts for analyzing security data, particularly that which comes from intrusion detection systems. One of the primary obstacles preventing

data analysis techniques from improving in a structured way was the lack of an explicit framework within which specific assessment properties could be defined precisely. Terms such as threat and risk are used frequently in the literature, but rarely - if ever - given explicit definitions that could distinguish them from related terms. For this reason an ontology of assessment properties was developed that incorporates the most critical security properties with definitions based on industry standards such as CVSS and IDMEF.

8.2 Conclusions On the Implementation and Testing of the Concrete Implementation

8.2.1 Data Quality

One of the critical issues that were confronted during this investigation was the accuracy of the assessment data. Two different filtering techniques were employed in order to increase data accuracy by eliminating false positives. The first was concrete vulnerability filtering, which allowed us to eliminate incoming alerts that did not correspond to a specific, exploitable software vulnerability. This was done by verifying that the alert referenced an entry in the common vulnerabilities and exposures database (CVE). The second technique used to improve the quality of the incoming data was configuration verification. By employing configuration verification, the system only considered alerts that presented a realizable threat to one of the nodes or services being monitored by the framework. This approach involved augmenting the local system vulnerability database with information regarding the affected software for each vulnerability. Simultaneously, a database of local system nodes, the software running on them and version information for each software product is established. By checking that the incoming vulnerability exploitation attempt would actually affect the version of software running on the node, many of the false positives generated during testing were eliminated. These two strategies were key in increasing the quality of incoming data and consequently the accuracy of the attack responses.

8.2.2 Changes From the General Approach to the Concrete Implementation

Extensive testing and multiple iterations of the Abacus framework led to the conclusion that although the process of context-aware decision making may be abstracted into three processes, the instantiation of those processes into actual software modules may require some adaptation. The best performance was achieved with an implementation that virtually joined the acquisition and analysis phases, such that all of the analysis tasks were performed as new data was acquired. The initial strategy of generating the analysis data when it was requested by the client proved to be prohibitively slow given the amount of data being generated in the system. This strategy, however, was only feasible because the system was event based (essentially that the data was discrete and not continuous). For systems where there are no such events to trigger the analysis tasks it will still be necessary to collect the data and perform the analysis tasks when the information is requested - although in these cases, there will be no such time penalty for analyzing multiple events. The only other situation to favor the on-demand analysis would be where most of the incoming data proved to be irrelevant to the final analysis and thus paying the time penalty for analyzing each event would prove unreasonable.

Another key question that needed to be answered was how to design an attack response that was tempered and still effective. We chose to use a strategy of restricting access permissions as the response to likely intrusive behavior. This response was in concert with the application domain being investigated and it was also less invasive than other response techniques in the literature that involve taking action against the suspected intruder. A risk assessment was synthesized from the provided data on vulnerability exploitation attempts in order to provide a quantifiable measurement of the changing state of system entities in relation to their prospect of being attacked. Because the risk assessments were calculated for individual system entities, the assessment data also allowed for more granular responses.

8.2.3 Effectiveness and Performance

One of the important conclusions we can draw from the testing data for the proposed approach is the feasibility of adaptive security mechanisms. The actual results of the attack simulations showed a marked improvement for the ratio of intrusive requests that were denied using the risk assessments. In the scenario that simulated an attacker performing vulnerability probing against the web server, 99% of the intrusive requests were denied, while only .7% of the normal requests were denied. In the case of multiple intruders for one target attack, the framework denied 93.5% of the intrusive requests while only denying 9.2% of the non-intrusive requests. Even in the scenario of multiple intruders on multiple resources, where authentication was employed as a response, more intrusive requests were authenticated than non-intrusive ones (93.5% to 87.9%, respectively), leading to a more efficient use of resources over the approach of authenticating all requests in situations of elevated risk.

The performance of the framework was another aspect of the approach that needed to be demonstrated and validated. The testing results showed that the framework, given limited server resources, was able to receive and process requests at a rate of over 1.3 million per day, exceeding the processing requirements for many high traffic domains and web sites.

8.3 Future Work

The area of designing adaptive security mechanisms is very broad and there remains a significant amount of work to provide systems with such capabilities that are suitable for use by industry and the general public. The system for acquiring context data could be extended to include a greater variety of sensors. Other types of correlation (besides matching values) could also be incorporated into the acquisition approach to enable assessments that are more predictive.

A reasoning engine based on the proposed assessment ontology could be added to the analysis server to make all of the different types of assessments available at the application

phase. This, however, would also demand that a wide variety of sensors are integrated in the acquisition phase so that the necessary inputs are present.

The various servers (alert and analysis) could be relocated to independent multi-core machines to investigate the impact of greater parallelism on expanding the capabilities for the handling of context information. Additional measures to secure the data transmissions between security components could also be added.

REFERENCES

[1] CERT Coordination Center, "Overview of attack trends," 2002.

[2] IBM Global Technology Services, "IBM Internet Security Systems X-force 2007 Trend Statistics," tech. rep., Internet Security Systems - IBM Global Technology Services, 2007.

[3] E. Bertino and L. D. Martino, "A service-oriented approach to security - concepts and issues," in *ISADS '07: Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems*, (Washington, DC, USA), pp. 7–16, IEEE Computer Society, 2007.

[4] R. Sandhu and P. Samarati, "Authentication, access control, and audit," *ACM Comput. Surv.*, vol. 28, pp. 241–243, 1996.

[5] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, pp. 186–205, 2000.

[6] C. Abad, J. Taylor, C. Sengul, W. Yurcik, Y. Zhou, and K. Rowe, "Log correlation for intrusion detection: a proof of concept," *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 255–264, 2003.

[7] N. Carey, A. Clark, and G. Mohay, *IDS Interoperability and Correlation Using IDMEF and Commodity Systems*, pp. 252–264. 2002.

[8] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pp. 202–215, 2002.

[9] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pp. 85–103, 2001.

[10] B. Morin, L. Mé, H. Debar, and M. Ducassé, *M2D2: A Formal Data Model for IDS Alert Correlation*, vol. Recent Advances in Intrusion Detection of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, October 2002.

[11] P. Ning, Y. Cui, and D. S. Reeves, *Analyzing Intensive Intrusion Alerts via Correlation*, vol. Proceedings of Recent Advances in Intrusion Detection : 5th International Symposium, RAID 2002, Zurich, Switzerland, October 16-18, 2002, pp. 74–94. 2002.

[12] P. A. Porras, M. W. Fong, and A. Valdes, *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation*, pp. 95–114. 2002.

[13] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system," in *In Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2004.

[14] Symantec, "Deepsight threat management system." https://tms.symantec.com/, 2008.

[15] MyNetWatchman, "http://mynetwatchman.com/," 2008.

[16] Dshield, "http://www.dshield.org," 2008.

[17] K. Henricksen, J. Indulska, and A. Rakotonirainy, *Modeling Context Information in Pervasive Computing Systems*, pp. 79–117. 2002.

[18] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *Journal of Network and Computer Applications*, vol. 28, pp. 1–18, 2005.

[19] "Context," *The American Heritageő Dictionary of the English Language, Fourth Edition*, Feb 2009. http://dictionary.reference.com/browse/context.

[20] "Context," *Merriam-Webster Online Dictionary*, Feb 2009. http://www.merriam-webster.com/dictionary/context.

[21] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Comput.*, vol. 5, pp. 4–7, 2001.

[22] P. Brezillon, G. K. Mostefaoui, and J. Pasquier-Rocha, "Context-aware computing: A guide for the pervasive computing community," *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, 2004.

[23] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*, (Nottingham, England), 2004.

[24] R. A. Kemmerer and G. Vigna, "Intrusion detection: A brief history and overview (supplement to computer magazine)," *Computer*, vol. 35, pp. 27–30, 2002.

[25] W. Hasselbring, "Information system integration," *Communications of the ACM*, vol. 43, pp. 32–38, 2000.

[26] V. Stavridou, "Integration in software intensive systems," *Journal of Systems and Software*, vol. 48, pp. 91–104, 1999.

[27] M. K. Perry, "Vertical integration: Determinants and effects," in *Handbook of Industrial Organization* (R. Schmalensee and R. Willig, eds.), vol. 1, ch. 4, pp. 183–255, Elsevier, July 1989.

[28] V. N. L. Franqueira, "Access control from an intrusion detection perspective," Technical Report TR-CTIT-06-10, Center for Telematics and Information Technology, Universitt of Twente, February 2006.

[29] T. Ryutov and C. Neuman, "The specification and enforcement of advanced security policies," *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pp. 128–138, 2002.

[30] T. Ryutov, C. Neuman, K. Dongho, and Z. Li, "Integrated access control and intrusion detection for web servers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 14, pp. 841–850, 2003.

[31] T. Ryutov, C. Neuman, and D. Kim, "Dynamic authorization and intrusion response in distributed systems," *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1, pp. 50–61vol.1, 2003.

[32] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt, *A specification-based intrusion detection system for AODV*. ACM Press, 2003. 986876 125-134.

[33] P. Uppuluri and R. Sekar, *Experiences with Specification-Based Intrusion Detection*, p. 172. 2001.

[34] J. Garcia, F. Autrel, J. Borrell, S. Castillo, F. Cuppens, and G. Navarro, *Decentralized Publish-Subscribe System to Prevent Coordinated Attacks via Alert Correlation*, pp. 223–235. 2004.

[35] R. Bhatti, E. Bertino, and A. Ghafoor, "A trust-based context-aware access control model for web-services," *Web Services, 2004. Proceedings. IEEE International Conference on*, pp. 184–191, 2004.

[36] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody, "Using trust and risk in role-based access control policies," *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pp. 156–162, 2004.

[37] N. Dimmock, "How much is "enough"? risk in trust-based access control," *WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies*, p. 281, 2003.

[38] L. Teo, G.-J. Ahn, and Y. Zheng, "Dynamic and risk-aware network access management," *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pp. 217–230, 2003.

[39] N. Stakhanova, S. Basu, and J. Wong, "A taxonomy of intrusion response systems," *Int. J. Inf. Comput. Secur.*, vol. 1, no. 1/2, pp. 169–184, 2007.

[40] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz, "A data mining analysis of rtid alarms," *Computer Networks*, vol. 34, pp. 571–577, 10 2000.

[41] J. Wang, B. Jin, and J. Li, *An ontology-based publish/subscribe system*. Springer-Verlag New York, Inc., 2004. 1045676 232-253.

[42] H. Wache, V. Ogele, T. Visser, U. Stuckenschmidt, H. Schuster, G. Neumann, and H. Ubner, "Ontology-based integration of information - a survey of existing approaches," (Seattle, WA), pp. 108–117, 2001.

[43] H. Debar, D. A. Curry, and B. S. Feinstein, "The intrusion detection message exchange format (idmef)," 2007. Request For Comments (Experimental).

[44] Sun Microsystems, "Xacml implementation," Accessed November 2007. http://sunxacml.sourceforge.net/.

[45] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, pp. 38–47, 1996.

[46] R. S. Sandhu and P. Samarati, "Access control: Principles and practice," *IEEE Communications Magazine*, vol. 32, pp. 40–48, 1994.

[47] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system," Aug. 1990.

[48] E. Fisch, *Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior*. PhD thesis, Texas A&M University, 1996.

[49] C. Carver, Jr. and U. Pooch, "An intrusion response taxonomy and its role in automatic intrusion response," *IEEE Workshop on Information Assurance and Security*, 2000.

[50] H. Debar, D. Curry, and B. Feinstein, *The Intrusion Detection Message Exchange Format (IDMEF)*. No. 4765 in Request for Comments, IETF, Mar. 2007.

[51] F. Cuppens, "Managing alerts in a multi-intrusion detection environment," *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pp. 22–31, 2001.

[52] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 01, pp. 146–169, 2004.

[53] G. Giacinto, R. Perdisci, and F. Roli, "Alarm clustering for intrusion detection systems in computer networks," *Machine Learning and Data Mining in Pattern Recognition*, pp. 184–193, 2005.

[54] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, pp. 105–136, 2002.

[55] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 274–318, 2004.

[56] D. Xu and P. Ning, "Alert correlation through triggering events and common resources," in *20th Annual Computer Security Applications Conference, 2004*, pp. 360–369, 2004.

[57] P. Ning, D. Reeves, and Y. Cui, "Correlating alerts using prerequisites of intrusions," Dec. 2001.

[58] B. Morin and H. Debar, *Correlation of Intrusion Symptoms: An Application of Chronicles*, vol. Recent Advances in Intrusion Detection, pp. 94–112. 2003.

[59] S. Godik, T. Moses, and et al, "Extensible access control markup language (xacml) version 2.0." OASIS Standard, February 2005.

[60] D. J. Weber, *A taxonomy of computer intrusions*. PhD thesis, Massachusetts Institute of Technology., 1998.

[61] C. Alberts and A. Dorofee, "Octave criteria, version 2.0," Dec. 2001.

[62] F. Swiderski and W. Snyder, *Threat Modeling*. Redmond, Wash: Microsoft Press, 2004.

[63] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0." http://www.first.org/cvss/cvss-guide.pdf, June 2007.

[64] L. Viljanen, *Trust, Privacy and Security in Digital Business*, vol. Volume 3592/2005 of *Lecture Notes in Computer Science*, ch. Towards an Ontology of Trust, pp. 175–184. Springer Berlin / Heidelberg, August 31 2005.

[65] D. J. Essin, "Patterns of trust and policy," in *Proceedings of the 1997 New Security Paradigms Workshop*, ACM Press, 1997.

[66] A. Avizienis, J. Laprie, B. Randell, and C. C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, Jan.-March 2004.

[67] T. Ryutov, C. Neuman, D. Kim, and L. Zhou, "Integrated access control and intrusion detection for web servers," *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pp. 394–401, 2003.

[68] C. A. Carver, *Adaptive Agent-Based Intrusion Response*. PhD thesis, Texas A&M University at College Station, May 2001.

[69] M. Ahmed, E. Al-Shaer, and L. Khan, "A novel quantitative approach for measuring network security," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 1957–1965, April 2008.

[70] MIT Lincoln Laboratory, "2000 DARPA Intrusion Detection Scenario Specific Data Sets." , http://www.ll.mit.edu/mission/communications/ist/ corpora/ideval/data/2000data.html, Accessed September 2008.

[71] A. Hess and N. Karowski, "Automated protection of end-systems against known attacks," in *Proceedings of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation*, (Tuebingen, Germany), 2006.

[72] Common Vulnerabilities and Exposures, "Common Vulnerabilities and Exposures List." http://cve.mitre.org/, Accessed September 2008.

[73] C. Kruegel and W. Robertson, "Alert verification: Determining the success of intrusion attempts," in *1st Workshop on the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2004)*, July 2004.

[74] U. Shankar and V. Paxson, "Active mapping: Resisting nids evasion without altering traffic," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, (Washington, DC, USA), p. 44, IEEE Computer Society, 2003.

[75] Joe Dog Software, "Siege." http://www.joedog.org/index/siege-home, November 2008.

[76] "Compete.com." http://www.compete.com, February 2009.

BIOGRAPHICAL SKETCH

Hassan Rasheed was born in 1981 in Florida to Howard and Barbara Rasheed. He graduated from King High School in Tampa, Florida in 2000. In 2004, he earned a Bachelor of Science degree in computer engineering from the University of Florida. After completing his bachelor's degree, he began working on his Doctor of Philosophy in computer engineering at the University of Florida. His graduate studies focused on distributed systems, information security and the design and implementation of context-aware systems.