Review of Database Intrusion Detection Methodologies using Attribute Dependence

Technical Report #TR-20130606-1

Adam Call
Department of Computer and Information Sciences
Indiana University South Bend

B561: ADVANCED DATABASE CONCEPTS

March 17, 2013

Review of Database Intrusion Detection Methodologies using Attribute Dependence

Adam Call

Indiana University South Bend
1700 West Mishawaka Avenue
South Bend, IN 46615

awcall@umail.iu.edu

ABSTRACT

Databases in the world today are highly exposed. To protect them more advance security systems are needed to deal with the variety of both internal and external threats. In this paper will summarize the types of threats that a database may experience. In addition it will review several different methods of implementing an Attribute Dependence Model based Intrusion Detection System. Ultimately this paper was written to present a plan for implementing one of the basic Attribute Dependence Method.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Security, integrity, and protection.

General Terms

Design, Security, Theory.

Keywords

Intrusion Detection System, Attribute Dependence Model, Database Security Threats.

1. INTRODUCTION

With the advent of the information age, databases have become highly exposed. The advantages of having your data accessible from anywhere in the world means that you now have that many more places for an attack to come from. This makes security, in this highly connected world, critically important.

There are many different types of security threats and each has its own challenges on how to deal with them. They range for the Denial of Service attack trying to cripple the database to the Legitimate Privilege Abuse of an employee. This paper will describe these different types of threats in more detail later. Not every database security method can handle all the different types of threats.

The security of a database can be largely divided into two layers. This first line of defense is prevention. This is what people typically think of when it comes to security. It focuses on preventing intruders from gaining access to the system. This includes passwords, encryption and even the locks on the doors. The main vulnerability of this system is legitimate privilege abuse or the so called "inside job". If the intruder has already made it into the system then all the prevention in the world will not help.

The second layer is a new concept called intrusion detection. Intrusion Detection Systems [2] are designed to detect, hinder and, if possible, eliminate intrusions. In modern systems the Intrusion Detection System is integrated directly into the security of the database engine as seen in Figure 1 [3]. Originally the Intrusion Detection Systems were standalone applications that essentially wrapped the databases in an extra layer of security. In these systems all transactions were sent to the Intrusion Detection System. It then decided whether or not to send them to the database. The results of the transaction would be sent to the Intrusion Detection System and then relayed on to the original requestor. This arrangement allowed Intrusion Detection Systems to be added to older database engines, since it accessed the database in the same manner a user would. The downside of this being additional overhead and a disconnect between the database security system and the database engine. This system can be seen in Figure 2. What is not shown is the result that travels back from the DBMS to the User. The modern, integrated, approach reduces some of this additional overhead and allows for further optimization, seeing as the Intrusion Detection System can be tailored to the architecture of the specific database engine.

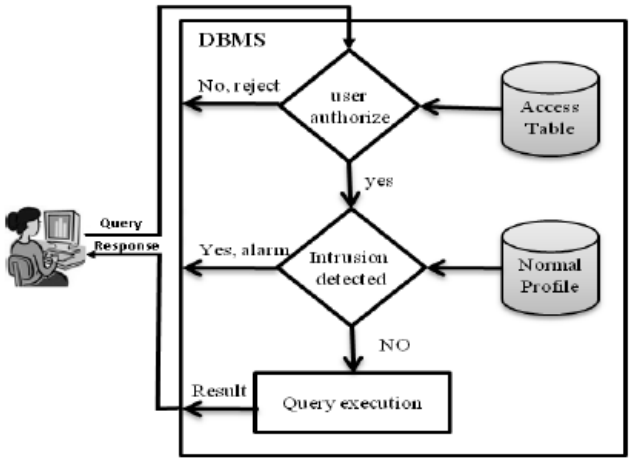

Figure 1. Integrated Intrusion Detection System. [3]

Most Intrusion Detection Systems are based on one of or a combination of two models, misuse detection and anomaly detection [5]. Misuse detection is based on rules. When configuring the system a set of rules are created that are used to trigger detection events. Each rule has a set of conditions it watch to be true, for example 10 failed password attempts if 5 minutes. Depending on the system the reaction to a triggered event could be as simple as adding an entry to a log or in the previous example, locking the account for some set period of time. The benefit of this system is that the false positive fate is very low. As long as the rules are well defined to not cause false positives, any detection is most likely valid. The reason this is important is

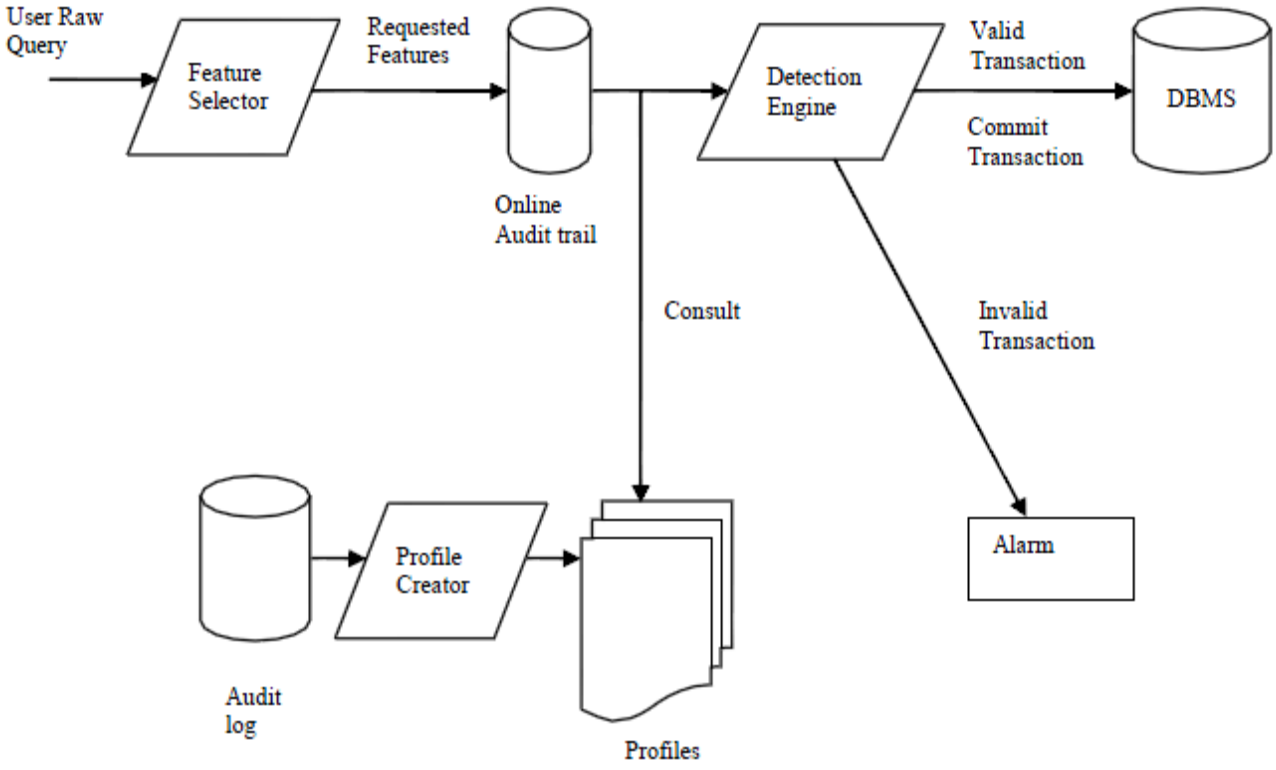

Figure 2. Architecture of Early Intrusion Detection System. [2]

because during normal operation of a database the number of valid transactions is far greater than the number of invalid ones. Too many false positive make it hard to know when an intrusion happened or not.

The problem with this system is it is static. It only knows how to react to intrusions that have a rule defined to detect them. A new type of intrusion could potentially go undetected until a later time. This type of system would also require regular updates to its rules to compensate for new types of intrusions.

Anomaly detection is a slightly more complicated system compared to the simple rule based misuse detection model. This complexity is due to it being profile based instead of rule based. By using profiles of valid user behavior, it can dynamically respond to new types of intrusions. A profile is a definition of normal behavior [2]. What is contained in this definition varies and is a focus of a lot of the research into this method of Intrusion Detection System. In the basic system the information contained is simply what actions [select, insert, update] are allowed on what attributes. These profiles can be defined for the system as a whole, on a security level or on an individual user basis.

The anomaly detection method goes through two modes of operation. This first is the learning phase. In the learning phase it analyzes what it is told are valid transactions. This is typically done via the transaction logs (audit trail). With the profile of valid behavior built, the system can them move on to the detection phase. When in this state, it is checking incoming transaction requests and comparing them to the valid behavior. Any transactions that fall outside the norm trigger a detection event.

The main disadvantage to the basic system is that it suffers from a fairly high false positive rate. False Positives are when the Intrusion Detection System says a transaction is an intrusion but it is actually valid. A lot of the research is being done into this method to find ways to correct this. This typically involves changing what information is included in the profiles.

This paper will review several methods for improving upon the basic Intrusion Detection System using variations of the Attribute Dependence Method. It will describe how these methods attempt to reduce the false positive rate or improve the detection rate. In addition, it will present an implementation plan for the basic Attribute Dependence Method.

The rest of this paper is organized as follows. Section 2 reviews previous research on Database security, with a focus on Intrusion Detection System research. A list and description of the types of security threats a database needs to protect against is in Section 3. Section 4 explains the different methods for improving upon the Intrusion Detection System. After the Conclusion in Section 5, Section 6 is the implementation plan for the chosen Intrusion Detection System method.

2. LITERATURE REVIEW

Database security has been an area of research since the early days of commercial databases. Later, the advent of the internet drove research into intrusion detection to protect the databases that were becoming more and more exposed. Basharat, Azamand Muzaffar [1] explain database security in a general sense and go into the different threats facing a database exposed to a network. They go into detail on how encryption is used as a security

measure.

Raoand and Patel [2] propose a method to implement an Intrusion Detection System on a Database. Their implementation has an Intrusion Detection System filtering incoming transactions for validity. They validate against what operations are valid for what attributes for what user. The problem with this method and the focus of a lot of the Intrusion Detection System research is False Positives.

The Attribute Dependency Model was created to lessen the false positive rate. Rezk, Aliand Barakat [3] show how this method might be implemented. They use data mining of audit logs to determine what attributes show dependency. They also propose integrating the Intrusion Detection System into the Database Engine to allow it to work with the database's security instead of in front of it.

Building on the Attribute Dependence Method, Rezk, Ali, El-Mikkawy and Barakat [4] propose an enhanced data dependency model. In their implementation, transactions are compared to the profiles of valid transactions. These profiles contain only the dependencies that have been found to be valid on that type of transaction.

Srivastava, Sural and Majumdar show the advantages of using a weighted Attribute Dependence Method. They explain that by weighting the attributes of the database you can effectively lower the threshold of validity for dependency on the sensitive data without having to lower it on all the data. This allows for more dependencies on sensitive data while minimizing the increase in false positives a lowering of the threshold causes.

3. OVERVIEW OF SECURITY RISKS AND TYPES OF INTRUSIONS

There are many different types of threats to a database that has to be exposed to the internet. Follow is a list of many of these types of threats. This list was taken from a journal written by Basharat, Azam, and Muzaffar [1].

3.1 Excessive Privilege Abuse

All users are granted some privileges in order to use the system. Privileges are defined as excessive when they are not needed to do the users job. These extra privileges can lead to the misuse of the system. This misuse can take on many forms. From the sharing or modification of sensitive data to even the modification of other users privileges (Privilege Elevation). The user could even go so far as to prevent use of the system altogether (Denial of Service).

3.2 Legitimate Privilege Abuse

In a similar fashion to Excessive Privilege Abuse, Legitimate Privilege Abuse is the misuse of privileges a user does need in order to do their job. These users are typically high level users, using their privileges to do illegal or unethical things. In other words this is what you would call an inside job and is one of the harder threats to protect against. Traditional preventive security system does not help when the intruder is already in the system.

3.3 Privilege Elevation

When an intruder, ether internal or external, gains access to the system one of the dangers is privilege elevation. This is the granting of privileges to users who should not have them. The intruder could even create new users to grant privileges to. The granting of privileges need not even be from within the system. It could be caused by a flaw in the database functions or protocols. It could also come from the SQL statements [SQL injection].

3.4 Database Platform Vulnerabilities

A system is only as solid as its foundation. When it comes to security, a Database is only as secure as its underlying systems. Vulnerabilities in the operating system can allow intruders to bypass the database security measures. These vulnerabilities need not even be in the software. The security on the database will not stop someone from steeling the hardware itself, if is it not properly secured.

3.5 SQL Injection

This involves injecting SQL commands into text strings that are sent to the database, for example including a closing string identifier (i.e. ") with addition SQL commands after. The database reads the inserted closing string identifier and thinks the string is done. It then proceeds to process the rest of the string as if it were an SQL statement. If this is not checked for the intruder could gain access to unintended parts of the database or even create a new user account for them to gain further access to the system (Privilege Elevation).

3.6 Weak Audit Trail

An audit trail is an automated record of all transactions on the database. The value of the audit trail is twofold. First it allows you to analyze the activity on the database and possibly detect intrusions after the fact. This can help you protect against that sort of intrusion in the future. The second benefit is it can help you recover from an intrusion when it happens. All process transaction will be logged regardless of if they are intrusion. This means you might be able to examine the audit log to see what the intruder saw or modified.

3.7 Denial of Service

Denial of Service attack, commonly referred to as a DOS attack, is any action meant to prevent legitimate users from using the service. This can take many forms, from flooding the network with garbage communication to attempting to crash the server. If the intruder can gain access to the system, they could even corrupt the authentication systems to prevent anyone from logging in.

3.8 Database Communication Protocol Vulnerabilities

The vulnerabilities in the communication protocols used to communicate can lead to intrusion into the database. Over time flaws have been found in all database retailers' communication protocols. This could allow the intruder to gather information from the data stream or even gain access to the database.

3.9 Weak Authentication

A week authentication strategy makes it easier for others to obtain login credentials. The following would fall into an authentication strategy. How often does the system require a password to be changed. How many passwords back does it keep track so that

the user cannot swap between a few passwords. How strong is the password, meaning how long, does it require both upper and lower case characters or numbers or symbols. All these make it harder for passwords to be discovered.

3.10 Backup Data Exposure

Typically people would think of backups as a good thing but they in themselves are a security risk. A backup is typically a copy of the database on some sort of portable physical media [tapes, DVDs, etc.]. Being portable the backs are even more at risk than the hardware itself, to theft or destruction.

4. INTRUSION DETECTION METHODS

To protect against all of the threats a database might experience, both a preventative and reactive security system is needed. When the security system fails to prevent an intrusion, a reactive system like an Intrusion Detection System is needed to detect the intrusion. Following is a description of several methods for improving upon the basic Intrusion Detection System.

4.1 Terminology

The three Attribute Dependence Models all use similar terminology, which require some explanation [3].

4.1.1 Sequence

A sequence is a primitive representation of a transaction. Primitive operations are reads and writes on specific attributes. A sequence is an order list of primitive operations. A primitive operation will be represented as o(a) where 'o' is either 'r' or 'w' for read and write respectively and 'a' is the attribute of the record being acted upon. A sequence will be represented as <o(a1), o(a2), …, o(an)> where 'ak' is the k^{th} attribute in the sequence with $1 \leq k \leq n$.

The support of a sequence is defined as the percentage of transactions that have the sequence as a subsequence. A subsequence is a sequence that can be created by removing but not rearranging any number of its primitive operations.

A transaction will be denoted as Ti: o(a1), o(a2), …, o(an), Commit where 'i' is the ID of the transaction Ti.

4.1.2 Rule

A rule is a sequence defined for a specific attribute. A rule will be denoted as R(a) for an attribute 'a' and has the form <o(b1), o(b2), …, o(bn), O(a), o(c1), o(c2), …, o(cm)> where 'bk' and 'cj' are attributes with $0 \leq k \leq n$ and $0 \leq j \leq m$ and 'n' and 'm' are some non-negative integer. In other words the sequence must contain an operation on 'a' that is defined as the primary operation and can contain any number of operations before and after the primary operation. Let the atomic rule of an attribute be the smallest possible rule. This sequence would only containing a single operation on itself, AR(a) = <O(a)>. Read sequences, pre-write sequences and post-write sequences can also be considered rules as they are more limited versions on the general rule.

4.1.3 Read Sequence

A read sequence of an attribute 'a' is defined as a sequence of attributes that must be read in a specific order before the read or write of 'a'. This sequence is denoted as RS(a) and has the form <r(a1), r(a2), …, r(an), O(a)>. The support of a read sequence is the same as for a normal sequence. In other words if any number of operations can be inserted into the read sequence to create the target sequence it is said that the target sequence supports the read sequence.

4.1.4 Pre-write Sequence

A pre-write sequence of an attribute 'a' is defined as a sequence of attributes that must be written in a specific order before the read or write of 'a'. This sequence is denoted as pre-WS(a) and has the form <w(a1), w(a2), …, w(an), O(a)>. The support of a pre-write sequence is the same as for a normal sequence.

4.1.5 Post-write Sequence

A post-write sequence of an attribute 'a' is defined as a sequence of attributes that must be written in a specific order after the read or write of 'a'. This sequence is denoted as post-WS(a) and has the form <O(a), w(a1), w(a2), …, w(an)>. The support of a post-write sequence is the same as for a normal sequence.

4.1.6 Confidence

The confidence of a rule is defined as the fraction of the support of the sequence and the support of its atomic rule over the full set of sequences. The equation for confidence would be as follows.

Confidence = Support(<o(b1), o(b2), …, o(bn), O(a), o(c1), o(c2), …, o(cm)>)/ Support(<O(a)>)

4.2 Attribute Dependence Model

Attribute dependence [3, 4] was one of the first methods implemented to try to create a better profile. Two attributes are said to be dependant if an operation on one attribute requires an operation on the other attribute. The attribute dependence model uses this concept to detect if a sequence is invalid.

The Attribute Dependence Model is an anomaly detection method as opposed to a misuse detection method. Like most anomaly detection methods, it requires an initial learning phase to build its profiles. These profiles contain dependency rules as defined for each user or user-group. The generation of these rules involves three steps: (1) Frequent sequence mining, (2) Potential dependency rule generation (these are read sequences, pre-write sequences and post-write sequences) and (3) Dependency rule validation.

4.2.1 Frequent Sequence Mining

Frequent sequence mining is the process of finding all sequences in an audit log that meet some minimum user defined threshold of frequency. These sequences include all possible subsequences of each transaction, not just the complete transaction sequences.

These transactions are analyzed on a per user basis. Meaning the audit log of transactions is divided up by user and analyzed individually for frequent sequences. Table 1 shows an example of this.

4.2.2 Potential Dependency Rule Generation

After mining out the frequent patterns, the potential rules can be generated. Each of these rules will be either a read sequence, pre-write sequence or a post-write sequence. The procedures for extracting these are fairly simple. For each operation O(a) in the frequent sequences perform the following:

1) Add a read sequence <r(a1), r(a2), …, r(an), O(a)> to the dependency rules for attribute 'a', where {r(a1), r(a2), …, r(an)}

Table 1. Example Frequent Sequences. [4]

Trans. ID	User ID	Transaction Operations
1	U1	r(7), r(1), r(6), w(5), r(1), w(4)
2	U1	r(1), r(5), w(1), r(4), r(5), w(4)
3	U1	r(7), r(6), r(2), w(4), r(7), r(3), w(6), r(1), r(6), w(2), r(3), r(5), r(2), w(5)
4	U1	r(2), w(2), r(4), r(7), w(3), r(6), w(5), r(1), w(4)
5	U1	r(6), r(1), w(3), r(1), w(6), r(2), r(7), r(4), w(2)
6	U2	r(5), r(3), r(6), w(7)
7	U2	r(2), r(5), w(6)
8	U2	r(4), w(6)
9	U2	r(6), r(5), w(5), r(3), r(4), w(4), r(3), w(7)
10	U2	r(5), r(6), w(5), r(5), w(4)

Table 2. Example Dependency Rules[4]

User1		User2	
Dependency rules	**Confidence**	**Dependency rules**	**Confidence**
Read rules			
r(1) → r(6)	80%	r(3) → r(5)	100%
r(5) → r(1)	100%	w(4) → r(6),r(5)	100%
r(6) → r(7)	75%	w(5) → r(5)	100%
w(2) → r(2)	100%	w(5) → r(6)	100%
w(4) → r(1)	75%	w(7) → r(6)	100%
w(4) → r(7),r(6)	75%	w(7) → r(5),r(3)	100%
w(5) → r(7),r(6)	100%		
w(6) → r(6)	100%		
Pre-write rules			
		w(4) → w(5)	100%
Post-write rules			
w(4) → r(7)	75%	w(5) → w(4)	100%
w(5) → r(7)	75%	w(7) → r(3)	100%
w(4) → r(6)	75%		
w(5) → r(6)	75%		
w(2) → w(6)	100%		

is the set of all read operations in the sequence before O(a).

2) Add a pre-write sequence <w(a1), w(a2), …,w(an), O(a)> to the dependency rules for attribute 'a' where {w(a1), w(a2), …, w(an)} is the set of all write operations in the sequence before O(a).

3) Add a post-write sequence <O(a), w(a1), w(a2), …, w(an)> to the dependency rules for attribute 'a' where {w(a1), w(a2), …, w(an)} is the set of all write operations in the sequence after O(a).

4.2.3 Dependency Rule Validation

The last step is to validate the potential read, pre-write and post-write rules against the original set of sequences generated from the audit log. This is done by finding the confidence of each rule on the original sequence set. Any rules that do not obtain some minimum confidence level will be removed from the dependency rules for that attribute. The rules that are validated will be used during the detection phase to validate the transactions. Table 2 shows the results of validating the rules generated from Table 1.

4.2.4 Detection Phase

The detection phase is used during normal operations to detect when a transaction in malicious. It does this by comparing each transaction against the profile for the user that owns the transaction. To validate the transaction each operation in the transaction has to be checked against the dependency rules for that operation's attribute. This is done by simply testing to see if the sequence supports the given rule. The methods used during the learning phase are reused here for the conversion of the transaction to a sequence and the check for support of each rule. If the transaction fails any of the dependency rules it is considered malicious and a detection event is raised.

4.2.5 Attribute Dependence Problems

There are three problems with this method. This first is with the frequent sequence mining. This will only allow frequent sequences to be considered for dependencies. The problem is when you have infrequent transactions on sensitive data. This sensitive data will not have rules generated for it and thus are effectively unprotected by the Intrusion Detection System.

The second problem is with the dependency rule validation. Consider when there are multiple types of transactions that use a piece of sensitive data that are each frequent enough to be considered. The problem is if these transactions are dissimilar, the dependency rules generated from them might not have enough confidence to be used.

The third problem is when ever you have a rule that does not have 100% confidence. This means there are some valid transactions when this dependency rule is not supported. Meaning, whenever one of those exception transactions is issued, it will be marked as malicious when it is actually valid. Thus while this method reduces the number of False Positives it still does not eliminate them.

To show this behavior consider the transaction T2: r(1), r(5), w(1), r(4), r(5), w(4) as seen in Table 1. While this is a normal transaction it conflicts with two of the dependency rules for user 1, r(1) -> r(6) and w(4) -> r(7), r(6). This means it will failed those tests and thus be marked as a malicious transaction, causing a false positive. Conversely, let us consider the transaction T: r(2), w(2), r(4), r(7), w(1), r(6), w(5), r(1), w(4). This transaction passes all relevant dependency rules but is actually a malicious transaction.

4.3 The Enhanced Data Dependency Model

Building off attribute dependence, the enhanced data dependency model [3, 4] is a method to significantly reduce the false positive rate. The main difference with this method is the inclusion of transaction information in the profiles. The additional information stored is the number of operations plus the read and write attribute sets. This information is gathered after the frequent sequence mining (Section 4.2.10). In the case of the example, assume that a write with its proceeding reads is one operation. The results of this extra information can be seen in Table 3.

The detection phase also changes slightly (Figure 3). First the list of transactions is searched for a transaction with the correct number of operations. If none are found then the transaction is marked as malicious. This list of transactions is then searched for a transaction with a similar read write set. A read write set is only what columns are read and written regardless of how much data is actually used in the transaction. If one is not found the transaction is marked as malicious. Otherwise the transaction is compared against the dependency rules for the transactions that passed the first two steps. If any of these transactions passes then the transaction is valid otherwise it is malicious.

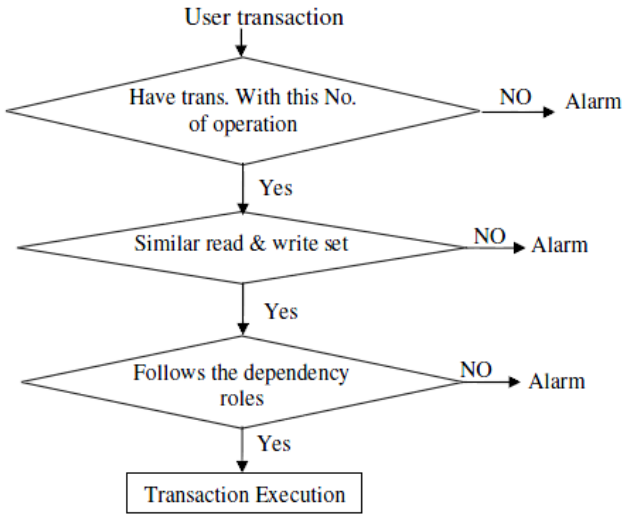

Figure 3. Enhanced Data Dependency Model Detection Sequence. [4]

Trans. ID.	No. of Operations	Read data set	Write data set
1	2	7,1,6,1	5,4
2	2	1,5,4,5	1,4
3	4	7,6,2,7,3,1,6,3,5,2	4,6,2,5
4	4	2,4,7,6,1	2,3,5,4
5	3	6,1,1,2,7,4	3,6,2
6	1	5,3,6	7
7	1	2,5	6
8	1	4	6
9	3	6,5,3,4,3	5,4,7
10	2	5,6,5	5,4

Table 3. Example Transaction Information. [4]

Trans. ID.	Read set	Pre- write set	Post-write set
1	w(5) →r(7),r(1),r(6) w(4) → r(1)		
2	w(1) → r(1),r(5) w(4) → r(4),r(5)		
3	w(4) → r(7),r(6),r(2) w(6) → r(7),r(3) w(2) → r(1), r(6) w(5) → r(3),r(5),r(2)	w(2) → w(6) w(5) →w(2)	w(6) →w(2) w(2) →w(5)
4	w(2) →r(2) w(3) → r(4),r(7) w(5) →r(6) w(4) → r(1)		
5	w(3) → r(6),r(1) w(6) →r(1) w(2) → r(2),r(7),r(4)		
6	w(7) → r(5),r(3),r(6)		
7	w(6) → r(2),r(5)		
8	w(6) → r(4)		
9	w(5) → r(6),r(5) w(4) → r(3),r(4) w(7) → r(3)		
10	w(5) → r(6),r(5) w(4) → r(5)	w(4) → w(5)	w(5) → w(4)

Table 4. Example Dependency Rules by Transaction. [4]

This method addresses the second and third problems described in Section 4.2.5. By keeping the dependencies on a per transaction level, the issue of dissimilar transactions on the same data driving each other's confidence rate down goes away. This eliminates the second problem all together. In a similar sense, the separation of the transactions allows the confidence rates of the individual rule

to go higher. The higher the confidence rate on a rule the less likely it is to cause a false positive. While the false positives will not be completely eliminated they will be significantly reduced.

Using Table 4, the previous example can be reexamined. Looking again as T2: r(1), r(5), w(1), r(4), r(5), w(4) as seen in Table 1, it can be seen that there are no longer any conflicts with its two dependency rules w(1) -> r(1),r(5), w(4) -> r(4),r(5) as shown in Table 4. Now let use consider again the transaction T: r(2), w(2), r(4), r(7), w(1), r(6), w(5), r(1), w(4). This transaction has 4 operations and thus could possibly be matched with T3 or T4. Its read set is {2,4,7,6,1} and its write set is {2,1,5,4}. While it's read set matches 4 its write set does not. Thus this transaction would be marked as malicious due to not having a compatible read write set.

4.4 Weighted Sequence Method

Further improving on the attribute dependency model is the weighted sequence method [5]. This method adds sensitivity weightings to attributes in an attempt to solve the first problem listed in Section 4.2.5. In an unweighted Attribute Dependence Method, the transactions are chosen only by how frequently they appear in the audit logs. In a weighted Attribute Dependence Method, each transaction is given a weight equal to the most sensitive attribute in the sequence. An addition weight is also added if a sensitive value is written. This weight multiplies the frequency of the transaction before it is compared to the threshold value. This has the effect of making transactions on sensitive data more likely to be considered for dependency rule generation. Figure 4 shows this slightly modified learning phase.

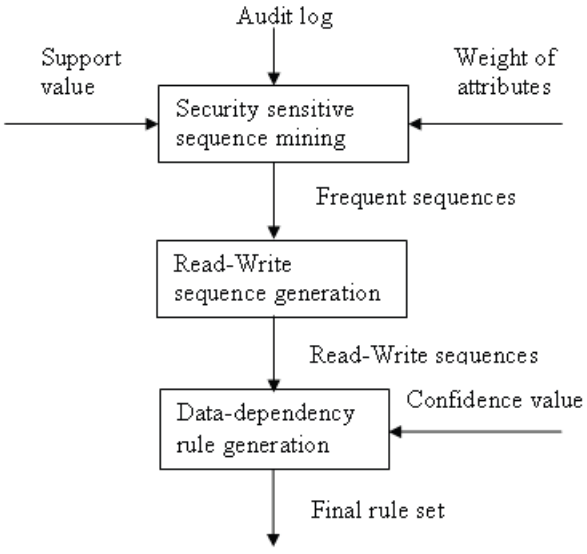

Figure 4. Components of the Weighted Data Dependency Rule Generation. [5]

The advantage of this system is transactions on weighted attributes effectively have a lower threshold of frequency to be considered than those of less sensitive data. This allows more dependences to be generated for the sensitive data without lowering the threshold of frequency of the whole system. Lowering the threshold allows more sequences to be considered and thus introducing more chances for false positives. In other words, this method puts more security on the sensitive data and less on the insensitive data.

5. CONCLUSION

Protecting against all threats is not possible with a tradition preventative security system. Using an Intrusion Detection System will help fill in the holes where the preventative security fails but can introduce the problem of False Positives. In this study it was shown that using an Attribute Dependence Model can mitigate the Intrusion Detection System Anomaly Detection Model's inherent problem of False Positives. The Enhanced Data Dependency Model further mitigates the false positive problem with minimal addition overhead. Lastly the Weight Sequence Method seeks to improve the detection rate on critical data without increasing the false positive rate on the less sensitive data.

Seeing as all three methods employ the Attribute Dependence Model, only implementing it is sufficient for an initial implementation. The basic method gives plenty of security with an actable False Positive Rate for an initial implementation. Working from the systems created for the Attribute Dependence Method, the more advanced methods can be more easily implemented at a later point in time.

6. IMPLEMENTATION PLAN

To implement an Attribute Dependence Method Intrusion Detection System in MiniDB will require the creation of several additional systems to support it. The learning phase will need an audit log from which to learn from. The audit log will have to also contain transactional information of some kind. This will require the implementation of at least a rudimentary transaction system. The audit log will also have to contain user information, thus necessitating the implementation of a login of some sort. With these systems to supporting the learning phase, the Intrusion Detection System can be created.

The top layer of the system will be a new security class to contain all these new systems. This security class will handle the user login and audit log generation, in addition to the learning and detection phases of the Intrusion Detection System. Finally on top of all that a text based user interface will have to be created to allow for user interaction. Since the purpose of this effort is to implement an Intrusion Detection System, the support systems will be rudimentary and only do a minimum of what is needed to support the Intrusion Detection System.

The first system that will be implemented in the security class will be the Login system. This will be accomplished with a table of integer login ids and string passwords. The interface at the top level will allow for the creation but not the deletion of users. This is so that the audit logs remain valid with respect to the user base. The system will require login before any transaction can occur. This login system will not be very secure though. This is because database containing the user IDs and passwords is an unencrypted plain text file. Future work on this system could be to implement a real login system and not just a placeholder.

The next system to implement is the concept of transactions. This will be done is a rather simplistic way. After someone has logged in they will be able to make the choice to open a transaction. After this they will be allowed to issue supported relation commands. A choose in the supported commands will be to commit. Note this system merely marks what transaction a command belongs to and offers no support for concurrency or

recovery. The commit in fact does not actually commit. It only ends the transaction. The commands previous will be committed as they are received. This is acceptable for this revision since the only features needed to support the Intrusion Detection System is the grouping of commands by transaction. Further work on this system would be to implement support of concurrency and recovery.

The last support system will be the Audit log. This log will contain all the information about the transactions. There will be two types of entries into the Audit log. The first is entries about transactions. These are mainly about when the transaction starts and stops and who owns it. The other entry is about the individual actions performed on the database. These entries will contain the type of action (project, insert, delete or update) and on which attributes these effected. Figuring out which attributes will require some work though. For insert and delete it is simply all attributes are written. A project's parameters explicitly tell what attributes should be considered read. If the project is being done on a results relation, the operation that generated it will be used to determine the attributes. Update will be the hardest and will require a change to the lower level code to compare to the new record to the original. These differences will be the attributes that are written. This audit entry will also contain the relation names and any parameters associated with the commands.

For the Intrusion Detection System a function for analyzing transactions that will be used in both the learning and detection phases will need to be created. This function will reduce the transaction to a read write sequence. A data structure for expressing a read write sequence and a data structure for conveying rules will also be needed. Before either of these a way to uniquely identify attributes must be created. To do an addition column will be added to the metadata stored for every field. This column will contain a unique integer across the whole database. These integers will just be added incrementally and will not make any adjustments is a filed is deleted. They will also carry forward through selects and projects. The read write sequence then becomes a linked list of nodes each containing an attributes unique identifier and either a 'r' or a 'w'. The expression of the rules will be a structure containing the attribute identifier the rule is for and a linked list of the sequences node the rule states should be present. The rules that apply to each attribute will be linked together and index to a single array by their unique identifier.

The detection phase will have several steps. The first will be to generate the read write sequences from the Audit log. Next is the generation of all the possible read, pre-write and post-write rules from the sequences. Multiple such potential rules will likely be generated from each sequence. These will then be evaluated over all sequences for validity. Those above a certain validity threshold are added to the detection rules. The detection phase is a lot simpler. It first reduces the transaction to a read write sequence. This sequence is then compared against all rules that apply to each of its attributes. If the sequence fails a certain percentage of applicable rules it is considered to be an intrusion and an alarm is raised.

Lastly a text based user interface and test program will have to be created. The interface will support user creation and login in. It will also allow the user to initialize the learning phase. Once logged in, it will support transactions and manipulation of the data. It will also alert the user when they try to commit is their transaction was considered and intrusion. Parallel to this will be an API implementation of the user interface to allow the test program to automatically exercise the system.

7. REFERENCES

[1] Basharat, I., Azam, F., and Muzaffar, A. W. Database Security and Encryption: A Survey Study. International Journal of Computer Applications, 47 (12). 28-34.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] Rao, U. P., and Patel, D. R. Design and Implementation of Database Intrusion Detection System for Security in Database. International Journal of Computer Applications, 35 (9). 32-40.Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[3] Rezk, A., Ali, H., and Barakat, S. I.Database Security Protection based on a New Mechanism. International Journal of Computer Applications, 49 (19). 32-38.Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 [Mar. 2003], 1289-1305.

[4] Rezk, A., Ali, H., El-Mikkawy, M., and Barakat, S.MINIMIZE THE FALSE POSITIVE RATE IN A DATABASE INTRUSION DETECTION SYSTEM. International Journal of Computer Science & Information Technology, 3 (5). 29-38.

[5] Srivastava, A., Sural, S., and Majumdar, A.K.Database Intrusion Detection using Weighted Sequence Mining.Journal of Computers, 1 (4). 8-17.