Design and Implementation of Intrusion Detection System in Databases*

Anonymous
Removed

ABSTRACT

Nowadays, data plays a pivotal role in most enterprise organizations to furnish their requirement in building software products. This need to use and preserve data has led various organizations to provide utmost importance towards the notion of secure storage of information in databases such that it is free from internal and external intrusion. In order to ensure security and privacy for such complex systems, the design considerations for such systems needs to create awareness and control over both data stored at rest and in transit. Intrusion Detection Systems (IDS) are one of the popularly known mechanisms employed to handle and protect big data. The data filtered by IDS is then used for different kinds of analytics facilitating the product requirement of varied organizations. The major focus of the paper is primarily towards achieving a part of this functionality of Intrusion Detection Systems, which is to maintain integrity in a database management system. In order to maintain a trusted computer base such alert mechanisms are being employed at an increasing pace among several enterprises. The paper proposes to maintain the consistency of the queries that are being fired at databases. This can be performed by using a technique known as template matching where all the queries to the database are first monitored based on different template patterns captured from transactions that are generated during the learning and detection phase of the life-cycle of an IDS system. The filtered data is then sent to the audit section where audit trails are maintained with information regarding access of databases for a particular requirement. The IDS system alerts the application maintaining a given database management system regarding an incoming intrusion based on the pattern of the queries that are fired. This application of IDS has a wide scope for expansion especially in forensics and predictive analysis for fraud detection. This is a budding phase for such mechanisms to influence and provide an impetus to securing a given database management system with integrity and protection for fraudulent activities.

KEYWORDS

IDS, Learning Phase, Detection Phase, Template Matching, Audit Logs, Transactions, Database Administrator, Profile Matching, DBMS

*Produces the permission block, and copyright information

NA, NA

DOI: 10.1145/nnnnnnn.nnnnnnn

ACM Reference format:
Anonymous. 2017. Design and Implementation of Intrusion Detection System in Databases. In *Proceedings of NA, NA, 2017,* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Databases are used in almost every organization nowadays and all of them have a database which needs to be secured from the Internet directly or indirectly. Separation of privileges among various database management systems is evident through a layer of physical security while storing the data and, the database in itself which has a good role based security mechanism in place. But what makes databases so vulnerable despite these separation of privileges in database security is the trust that is put in all the database transactions that comes from a different layer in the application. So, there is a need for a reference monitor which validates all of these transactions. This mechanism of validation is called as an Intrusion Detection System. [2] There are a lot of good softwares in the industry when it comes to a Web Application or a Network Intrusion Detection System (NIDS), but when it comes to databases, the number of applications and the research is limited. This is the primary motivation behind building a better IDS for Database Security. [1]

This paper proposes the design and implementation of Intrusion Detection system for Database Security. This proposed model consists of two phases of detection of malicious transactions, namely, Learning phase and Detection phase [2] [6]. In the learning phase, the IDS requires a set of authorized and valid SQL queries, which serve as a template to the detection phase. The information extracted from SQL queries such as the command type, target object, transaction ID etc. is then stored in as a template in a XML/JSON file format. The Database Administrator is then responsible to extract valid transactions from previously available database audit logs. Before taking the Intrusion Detection System on-line, the valid transactions previously extracted should be fed to the system. The valid templates are generated in the learning phase and forwarded to the template matching algorithm in the detection phase. The main focus in the detection phase is to evaluate every transaction that is categorized as either valid or invalid. If the system validates the transaction as invalid, it immediately drops the transaction and alerts the database administrators. If the transaction is valid, the system allows the transaction to go through. Every transaction is registered by the system under valid and invalid transaction audit log for future reference.

The paper proposes to store the transaction logs in XML/JSON file format and uses structured queries with object relational mapping (ORM) through PostgreSQL. Extracting valid transactions and other processing will be accomplished using Python.

2 DESIGN CONSIDERATIONS

The scope of an IDS system is determined by its system design. In order to consider a design pattern for any system it is of utmost importance to understand the types of malicious attacks possible on it. A malicious user can be categorized into 3 main types, namely, an authorized user with malicious intent, an authorized user with a compromised account, and an external user. The first 2 types of users are internal users who might be a developer of the application, a Database administrator, or anyone from the organization who is involved in working on the database with appropriate privileges. Finally, external users form the most critical part of the categorization from the security perspective of an IDS. This is mainly because the chances of a compromise into the system by an internal user is less owing to the several layers of corporate security one has to go through. So our Intrusion Detection System, mainly focuses on detecting and preventing external users. This system can also be extended to accommodate internal threats by slightly tweaking the system design.

Our system works on the transaction level where checks are based on queries and the system design comprises of two main phases: the **Learning Phase** and the **Detection Phase** .

2.1 Learning Phase

The main component in Learning phase is the transaction log. All the transactions in the transaction log file should be a valid transaction. This log will contain all the flow patterns of all types of valid transaction possible. If the transaction log is corrupt in incorrect data or malicious transaction that would raise false positive alarms and it might even flag malicious transaction as a valid transaction.

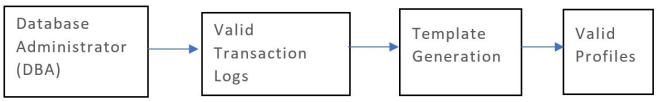

Figure 1: Learning Phase (JPG).

From the figure, we can notice that the Database administrator is responsible for feeding a valid transaction log to the IDS. These logs are then grouped with respect to an unique identifier in order for each of the transactions and records to be templated as a valid transaction pattern. These templates are then passed on to the detection phase.

2.2 Detection Phase

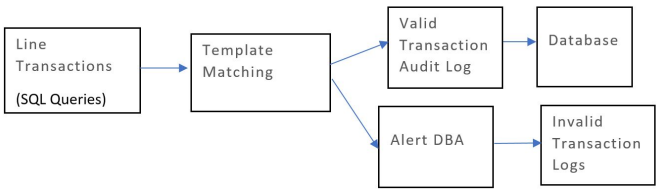

Figure 2: Detection Phase (JPG).

This phase comes into play when dealing with live SQL queries. In this phase as well the SQL queries are parsed and grouped with respect to their unique identifiers in order to complete a transaction successfully. The templates generated in the learning phase are used in this phase for template matching which leads to taking the decision whether to allow a particular transaction or not. If the transaction successfully matches with one of our valid transactions templates then the system allows the transaction to go through and finally logs it as a valid transaction log[4]. If the transaction does not match with any of the valid transaction templates then the transaction is flagged as a malicious one and the IDS alerts the database administrator and logs it in the malicious audit logs for further investigation on a dashboard as shown in the figure.

3 ARCHITECTURE

Learning and detecting phase have most the architecture same. Every SQL query is parsed the same way and stored in the same way regardless of being learning or detection phase.

In training phase, the template graph is created. This graph is can be considered as a trained model. The graph's path are all valid paths.

In the detection phase, the SQL query is parsed to get a undirected single path graph which is compared with all the valid path in the graph if there is a valid path then we allow the query to go through else it raises alarm to the database administrator. Path traversal here is done by depth-first search algorithm.

4 IMPLEMENTATION

The implementation of the Intrusion Detection System on Databases follows a 4-tiered architecture where each of the tiers can be upgraded and modified in a manner such that each tier is independent of the other. The Presentation tier is responsible to provide an user interface for the Web application in order for the user to interact with the application by performing different operations which leads to corresponding queries being fed to the Database Management System. The Web application has been currently hosted on a Localhost server for simulation and testing purposes and will later be deployed on an actual web server. The Application tier consists of the Business Logic of the application where the main implementation of the Intrusion Detection System resides. It is responsible for profiling the queries by matching it with approved templates. This has been built over python 3 and upon authentication and authorization lends its output to the persistence tier. Therefore, this stage comprises of a query parser, and an authenticator module which performs template matching and analysis.

The Persistence tier is responsible for the Business Logic of the application to perform connections with the Database Management System and thereby pass the queries to the Data tier. The flow of data between each of the tiers in the application has been depicted in the figure . The psychopg2 library [7] in python takes care of mapping queries from Python to PostgreSQL.The Data tier is responsible to store all the data generated through queries in a Database Management System. Here, we have made use of PostgreSQL to house and query the database.The Web application is built over the Flask framework (version: 0.12) [5] and makes use of templated HTML5 and static CSS3 for frontend. The business logic

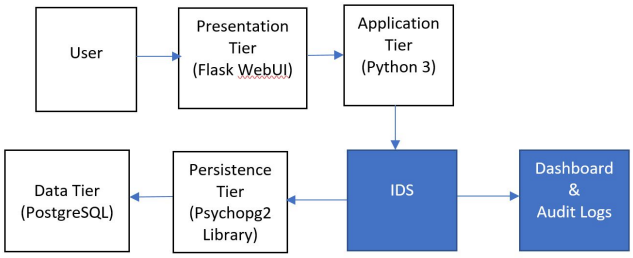

Figure 3: Flow of Data in a 4 - tiered application

is built using python 3 on PyCharm 2016.2 editor and the queries are in just plaintext (SQL) on PostgreSQL 9.6.2 [3] with PGAdmin4.

4.1 Simulation

The Intrusion Detection System on Databases has been depicted based on a simulation created via an online store. This online store provides certain functionality to the users which assists in illustration of different queries being sent to the databases. The function of adding a product on to the shopping cart simulates a select query being triggered so that the respective queried products from the database are visible. The removal of a product from the view represents a "DELETE" query being triggered so that the product can be no longer visible for the user for a particular transaction. This feature in normal circumstances must be categorized as an anomaly as the users don't have privileges to delete products from the view of the online store. A given user also has the feasibility of deleting items already loaded onto the shopping cart which would be simulated as an "DELETE" query being triggered to the database but valid in this scenario. Finally, the checkout option for the user simulates an "INSERT" operation in the database owing to the products being checked out from the shopping cart. These functionalities listed above can be used to associate the idea of simply generating queries. Once these queries are generated and checked out, they are sent to the Intrusion Detection System to be parsed and authenticated. If an irregular pattern is discovered during the parsing phase of the IDS system. The system would raise an alert by preventing the query to be triggered to the database. If the queries sent to the IDS System successfully pass all the tests and checks attributed towards the validity and integrity of the queries the application will then forward the queries to the persistence tier which will in turn fire the queries to the database.

This simulation helps us understand the complexity of an Intrusion Detection System at a micro level where the intrusion on databases is monitored and protected. Owing to the granularity the concept introduces the traditional viewpoint towards data and databases. The purpose behind restricting the functionality of an IDS system concealed to just databases is owing to the integrity principles in question. The basis of a well formed transaction is on the level of security is implemented on the databases which in turn ensures that there is no unauthorized data manipulation. This idea of data manipulation at such a granular scale makes sure that it is constrained in preserving the integrity of the data provided to a user for any application. This also brings into the picture of least privilege and referential integrity of database system being used where the IDS system acts as the reference monitor to prevent unauthorized access.

4.2 Template creation and Detection

Templating or profiling can be done using various methods and use of machine learning algorithms to solve this problem is intuitive in today's time. First proposed method was to use any of the machine learning algorithm to achieve templating creation and template matching. The use of clustering algorithm and training the model by having two classes as malicious and non-malicious transaction. But there are challenges in these type of anomaly detection and these problems are inherited form the disadvantages of machine learning techniques itself. Generating a good amount of non-malicious data is easy by just automating the application but generating the malicious traffic is the most difficult part. Without much of malicious data any machine learning algorithm will give bad results. So we have proposed and implemented an new approach which requires only the non-malicious data.

Our approach in creating the template uses only the logs of non-malicious transactions. In the learning phase it takes all the SQL queries one by one parses to blob saving all the data required to restrict back the SQL query other the values in the query. This blob of data or the Node is implemented like the graph data structure. For example, select query mentioning column names, FROM table and WHERE conditions will be parsed and saved as SELECT Node with multiple connections to FROM nodes and FROM nodes will have connections to WHERE nodes. This kind of architectural parsing to all SQL query allows easy and faster way of template matching and this method also blocks one of major application flaw which is affect databases: SQL Injection. As we know that SQL Injection is a application flaw which allows user to modify the SQL query with specially crafted user inputs. But we know that SQL injections usually have extra attributes or multiple union operations which will not be part of the non-malicious traffic, so these user injected malicious queries are also detected by our intrusion detection system

4.3 Testing

Several testing schema have been employed for ensuring a smooth transitioning of data at both the REST end-points and the data in transit in a secure fashion. First of all, preventive mechanisms have been put in place against most common attacks discussed in the "planning for investigation teams" section. Next, regression testing was performed using scripts for random generation of queries and querying the database with it. Method of injection has been satisfactorily managed using a modular design and separation of the front-end simulation with the back-end processing. Login and Registration forms have been tested with different attack patterns which have been successfully upheld through protective measures incorporated. This has been also communicated to us through the feedback of other investigating teams. Preventive mechanisms on REST end-points through Wekzeug Security feature in Flask has been tested using on-line REST clients such as Advanced REST Clients(ARC) and also through direct access to those respective end-points. Thus, testing and evaluation of security measures and

privacy parameters have been taken care of at each of the phases of the software development life-cycle of the web application for an efficient implementation of Intrusion Detection Systems in Databases.

5 LESSONS LEARNED

Database intrusion detection system(DIDS) provides for a secure means of access to a database management system with a priority on security measures implemented and privacy aware control framework incorporated. Our research provides proof of improvement in the performance and reliability of use of applications for personal and enterprise use. Employing privacy-aware mechanisms in distributed environment on an enterprise level which include principle of security and privacy such as separation of privileges, least privilege etc. Implementing a modular design for ease of access to the user which also helps provide access control at a granular level for each module. While sharing sensitive information over unknown networks either secure the channel through necessary means or encrypt the data being sent over an insecure network. Besides, the lessons learned include the various transitions in the implementation of security and manipulation of policies that can be looked at from various stages of a software life-cycle of a project.

5.1 Response to Investigating Teams

Provided Feedback and analysis as part of phase 4 submission.

5.2 Other Lessons

The authors always learn from the experiences they gain during the various researches done for developing a project. This helps in learning new things every day thus helping in enriching the knowledge and understanding. This project is one such example where learning was immense. It was known from the start that author may come across to some interesting things during designing and implement the Intrusion Detection System in Databases.

It is necessary to inspect all inbound and outbound network activities and also identifying suspicious patterns which may result in the attack on a network or system from someone who wants to break into a system. To detect and avoid these suspicious activities an intrusion detection system is used. These IDS are categorized into several ways.

Misuse Detection vs Anomaly Detection: All gathered information are stored in the database and later it gets compared to large databases of attack signatures by misuse detection. Essentially, the IDS looks for a specific attack that has already been documented. After few types of researches, the author can conclude that misuse detection software works as good as a virus detection system because these both work based on attack signatures that it uses to compare packets against. On the other hand in anomaly detection, the system administrator defines the baseline, or normal, the state of the network's traffic load, breakdown, protocol and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.

Network-based vs. Host-based systems: The individual's packets which flow through the network can be analyzed through the network-based system. Packets which are intending to harm can be detected by network based system which is designed to be overlooked by a firewall. Whereas, in a host based system the IDS detects all the activities on each computer or host individually.

Passive system vs. Reactive system: Use of passive systems assist you with the potential of detecting the security breaches. After detecting the breaches it logs into the information and sends the notifications or signals an alert. Whereas, in a reactive system all malicious activities are responded by IDS. These activities get blocked by the firewall to avoid network traffic from the malicious sources. Though these both works for the same purpose of securing the network.

6 PLANNING FOR INVESTIGATING TEAMS

Our plan for investigation of applications of other teams involved rigorous testing and evaluation of performance of different modules in terms of the level of security implemented and parameters used for protecting privacy thorough policies. The assessment of the application can be completed for different parts of the application by distributing the application into different tiers to evaluate the modularity and scalability of the application. The application can also be looked at from different perspectives of usage, the application can be evaluated based on the design considerations incorporated for personal and enterprise use. The presentation tier must provide for feasibility to use in terms of user interaction and user experience (UI/UX). In this phase, functional features implemented and availability of the application i.e. the uptime for the environment provided for testing the application play a significant role. Next, in order to identify and define the attack surface for the investigation, the application can be analyzed by exploring common attacks that can be used for exploitation such as SQL Injection on the login forms, input query parameters for querying the database, Man-in-the-middle attacks, launching Distributed Denial of Service(DDoS) etc. Security testing can be performed based on the SWOT analysis on the system which involves evaluation of strengths, weaknesses, opportunities, and threats. Next, the access control mechanisms can be tested in order to obtain a measure of the granularity in the level of security implemented for role and attribute based access. Runtime attacks can be proposed to be launched in order to evaluate functional features through the sessions generated, information stored through cache, retrieved information through REST end-points for cross-site request forging, thereby, analyzing the vulnerabilities to the application which can be exploited. At this stage it would be appropriate to explore an attack on the networking module if used. Sniffing and Snooping information being shared over the network forms a significant part of the attack surface. Here, policies and security measures to prevent Denial of Service needs to be checked. Also, by injecting traffic onto that network the service could be delayed, besides, loss of information can be a probable threat as well. Weak encryption and protocols could be thoroughly checked in this phase. Finally to stage persistence and database layer attacks, authorization strength of the application will be measured, as in most cases it happens to be the biggest threat for a failure to ethically and legally store data based on the security and privacy paradigm. However, hacking techniques may vary from application to application depending on the varied level of security imcorporated in order to strengthen the application.

7 ETHICAL AND LEGAL ISSUES

In today's world, intrusion detection system is yet to gain total penetration amongst enterprise applications that could allow organizations to leverage this feature to prevent legal and ethical issues from arising. After a significant amount of research we have identified some important stand points which could lead to ethical and legal issues. Some of these issues and challenges are listed as follows:

- Faulty Detection Algorithms.
- Issues due to man-in-the-middle attack
- Issues arising on account of cross-site request forging
- Mismanagement of Audit Logs
- On account of Poor Design
- Attacks during Maintenance of an IDS
- Testing & Evaluation of IDS

Now, let us look at these issues and challenges in greater detail. Policies should be enforced on detection algorithms used in IDS according to the data privacy paradigm. Detection forms an essential part of an IDS and helps to detect malicious queries and intrusion within the application. Therefore, vulnerabilities in this phase could lead to a major breach in terms of product related information.

Man-in-the-Middle attacks are possible when the client tries to query the database as the query request is forwarded to the IDS which could be residing in some unknown network. As a result of which, the data being shared over the network needs to be encrypted or sent through a secure channel. In effect of the attack, there is a possibility of a violation of privacy of a user trying to purchase different products from the application website.

Cross-site Request Forging (CSRF) attacks could make the application vulnerable due to loss of information or by gaining access to privileged information being routed over various REST endpoints. But from the projectfis point of view, a defense mechanism is in place as a result of using Flask for routing which uses a Werkzeug security module that prevents unauthorized GET/POST and illegal calls to random REST endpoints for obtaining private information that could leverage ethical issues for a given client.

All the transactions contain information regarding a purchase or a requisition made by the customers. If this data is breached and unknowingly accessed, there could be a possibility that the private details of a shopping transaction for customers is leaked. All the information such as the product transaction information and the user login credentials is both stored in the same database then if one of the tables in the database management system is compromised there is a good chance of the data in other tables of the same database will be compromised. But, our application follows a modular design, thereby, stores user login information and user purchase information in different databases. This avoids any ethical or legal issues that could arise due to leakage of private information when an attacker gains access of any one of the many tables in the database. Similarly, if one of the databases goes down, the integrity of the other parts of the database management system is still maintained. Therefore, maintaining such a modular design consideration also avoids a single point of failure of the entire application by enforcing access control over each module individually.

In order to keep a track of transactions in the system, two types of logs are generated and maintained, one is generated upon access to the DB known as the audit logs which contains the information on the list of people who access the database at any given time and the second is the audit trail which gets generated during a transaction, which contains the transaction details of the shopping transaction for the customers. These details logged by the application can help us to keep a track of access times to the database for a given transaction. In the case of any attacks to the application the point of intrusion can be obtained with ease.

Ethical and Legal Issues arising due to Insider attacks forms an important aspect of the data security and privacy paradigm. In our project the passwords from the user login credentials are encrypted using SHA256, which will restrict any inside attackers (developers, employees) from accessing the private information using the login username and password to leverage private and sensitive information.

Maintenance of an IDS can lead to interruption of service and thereby create a vulnerability to the database management system being used on account of queries not being filtered before executing on the database. This could create major breaches in terms of critical information being either lost or modified. Therefore, a critical infrastructure plan should be in place backing up the server in case the IDS goes into maintenance.

For testing and evaluation purposes, certain mechanisms such as load balancing need to be in place for an IDS that would help protect it against DDoS attacks. This would ensure that a part of the service running in the application does not go down on account of such attacks leaving other parts of the application vulnerable.

8 CURRENT STATUS & FUTURE WORK

Currently, the architecture of the template matcher in the IDS is such that it requires to be manually run and trained before the Simulation is run. But this process can be automated in future such that supervised learning can be introduced using an integration of neural networks in the learning and detection phases of the architecture that could auto-generate templates for the template matcher thereby significantly smoothing the functionality of the IDS implemented for different database schemes. The networking module set up for communication between the simulation of the web application and the IDS can incorporate security mechanisms to encrypt the data being sent in TCP/IP or can use secure channels for data transfer. Currently, Highly interactive and effective web pages have been built for the user to access the website with ease and get a sense of the underlying idea behind the simulation. A database schema is maintained for PostgreSQL database in the persistence tier module which relies on object relational mapping(ORM) of different attributes. A NoSQL database can be supposedly used instead of a traditional database management system to avoid incorrect relational mapping and for a higher speed of access to the database. Although, each module has been simulated and tested by ensuring the security of the data being shared and taking the privacy of the application, load balancing is something that has future scope that would ensure that the entire system becomes scalable to use.

Other features that can be proposed for an IDS on databases include maintaining timestamps in audit logs and audit trails for

transactions in the dashboard created. This will ensure reliability and avoid non-repudiation. Reference monitors can be placed to authorize and authenticate access to the IDS that can prevent attacks on the system where IDS is located, thereby, making it a trusted computer base. Role based and Attribute based access control mechanisms can be implemented to ensure granular security and privacy. Honey pot techniques can be also employed in order to purposefully attract and track attackers. Thus, the IDS on databases has a wide scope of research and modules that can be implemented for improving the efficiency and accuracy of the algorithms, thereby, making it scalable to use.

8.1 Tables, Figures, and Citations/References

REFERENCES

[1] E. Bertino and R. Sandhu. 2005. Database security - concepts, approaches, and challenges. *IEEE Transactions on Dependable and Secure Computing* 2, 1 (Jan 2005), 2–19. DOI: http://dx.doi.org/10.1109/TDSC.2005.9

[2] J. Chen, Y. Lu, and X. Xie. 2007. An Auto-Generating Approach of Transactions Profile Graph in Detection of Malicious Transactions. 1 (Nov 2007), 562–565. DOI: http://dx.doi.org/10.1109/IIH-MSP.2007.76

[3] PostgreSQL Global Development Group. 2017. PostgreSQL: Object Relational Database Management System. Github. (February 2017). https://www.postgresql.org/, Accessed March 8, 2017.

[4] A.C Newman. 2014. Intrusion Detection and Security Auditing in Oracle. *Application Security Inc.* (2014), 36–47.

[5] Armin Ronacher. 2012. psychopg: . Github. (April 2012). flask.pocoo.org, Accessed March 8, 2017.

[6] Ricardo Jorge Santos, Jorge Bernardino, and Marco Vieira. 2014. Approaches and Challenges in Database Intrusion Detection. *SIGMOD Rec.* 43, 3 (Dec. 2014), 36–47. DOI: http://dx.doi.org/10.1145/2694428.2694435

[7] Daniele Varrazzo. 2017. psychopg: A PostgreSQL library. Github. (March 2017). http://initd.org/psycopg/, Accessed March 8, 2017.