Data Stream Controller for Enterprise Cloud Application

Akshaya Prarthana Selvaraj[1] Dr. D.Venkata Subramanian[2]
[1]Student [2]Professor
[1,2]Department of Computer Science
[1,2]Saveetha School of Engineering Saveetha University – Chennai

Abstract— Cloud computing is an emerging computing paradigm where computing resources are provided as services over Internet while residing in a large data center. Even though it enables us to dynamically provide servers with the ability to address a wide range of needs, this paradigm brings forth many new challenges for the data security and access control as users outsource their sensitive data to clouds, which are beyond the same trusted domain as data owners. The occupier need not be concerned with how the Paas system achieves expansion under high load.MAC systems differ as security policy is defined for the entire system, typically by administrators. Information flow control (IFC) is a MAC approach, developed originally from military information management methodologies. IFC can be used to enforce more general policies, using appropriate labeling and checking schemes. The labels can be used to manage both confidentiality and integrity concerns, tracking "secrecy" and "quality" of data, respectively. Decentralized Information Flow Control (DIFC) is an approach to security that allows application writers to control how data flow between the pieces of application and the outside world. As applied to privacy DIFC allows un trusted software to compute with private data while trusted security code controls the release of that data. As applied to integrity DIFC allows trusted code to protect un trusted software from unexpected inputs.

Key words: Cloud Computing, Windows azure tool, Information flow control, Data privacy, Data base, windows 7,Visual Studio .Net 2010 Enterprise Edition, Visual Studio .Net Framework (Minimal for Deployment) version 4.0, SQL Server 2008

I. INTRODUCTION

The key technical challenge in cloud security systems from the fact that cloud infrastructures combine heterogeneous software and services written by multiple development teams with no shared approach for guaranteeing data security. The Internet had always provided some remote access but increasing bandwidth made it necessary to consider computing beyond firewall-protected local administrative domains, giving rise to new security concerns. Cloud computing gained significant momentum with widespread user adoption of dynamic websites. Cloud service offerings are typically divided into three broad categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). A major incentive for cloud tenants to use PaaS services is that the APIs often give a "scale out" capability, transparently deploying more or fewer resources as required. The tenant need not be concerned with how the PaaS system achieves expansion under high load. A major incentive for cloud tenants to use PaaS services is that the APIs often give a "scale out" capability, transparently deploying more or fewer resources as required.

We believe that DIFC is of particular relevance to the cloud, and indeed to any complex distributed system, as in these systems the security infrastructure will typically have an independent and longer life-span than the applications being managed on the platform. Individuals and organization that use cloud services to store their sensitive data generally rely on the provider to maintain an appropriate level of security. However, it is often the case that agreements (SLAs) between cloud providers and tenants are silent with respect to security guarantees, or even disclaim many types of service responsibility.

II. SCOPE OF TE PROJECT

[1] A fundamental problem is the existence of insecure information flows due to the fact that a service provider can access various employees in clouds. [2] MAC system differ as security policy is defined fir the entire system, typically by administrators. [3] Information flow control is a MAC approach developed originally from information management methodologies. [4] IFC is a data centric, and achieves protection by associating security labels with data, in order to track and limit data propagation. [5] The labels are also associated with principals in the system. IFC security policy defines permitted relationships between the labels of data and the labels of principals requesting access to data. [6] We believe that DIFC is of particular relevance to the cloud, and indeed to any complex distributed system, as in these systems the security infrastructure will typically have an independent and longer life-span than the applications being managed on the platform. [7] For IaaS, there will be situations that require collaboration across IaaS services, e.g. when Employees wish to share data. [8] IFC provides the means for managing and securing information flows both within and between multiple Employees. [9] As a result, there is potential for decentralized IFC to achieve better cloud security than is available today. [10] Since IFC security is linked to the data that it protects, both Employees and Admin of cloud services can agree on security policy, in a manner that does not require them to understand and rely on the particulars of the cloud software stack in order to effect enforcement.

III. PROPOSED WORK WITH INFORMATION FLOW CONTROL (IFC)

We investigate the feasibility of deploying IFC as part of the next generation of secure cloud infrastructures. In order to apply these techniques to a cloud environment a number of challenges need to be overcome. These include: selecting the most appropriate DIFC model, policy specification, translation, and enforcement. It is important that application developers using cloud provider IFC are aware of the trust assumptions inherent in the IFC provision. To overcome this existing problem we have used the access control and authentication to the user those who access data's from

cloud. One trivial solution to achieving secure data sharing in the Cloud is for the data owner to access his data before storing into the Cloud, and hence the data remain information-theoretically secure against the Cloud provider and other malicious users.

Any member within the group should be able to gain access to the data anytime with admin authentication. No-one, other than the data owner and the members of the group, should gain access to the data, including the Cloud Service Provider. The data owner should also be able to revoke access rights against any member of the group over his or her shared data.

IV. INFORMATION FLOW CONTROL IMPLICATION PROCEDURE

A. Input (I)

(1)Admin enter his user id and password for login.(2)User enters his user id and password for login.(3)Admin enter user id or date for track the user login information(4)New users give his completed personnel, address and phone details for registration.(5)Admin gives different kind of user information for search the user data.(6)User gives his user id, hint question, answer for getting the forgotten password.(7)Employee asking customer service details before process the queries.(8)Employees search the customer information while process

B. Outout (II)

(1)Admin can have his own home page.(2)Users enter their own home page.(3)The user defined data can store in the centralized database.(4)Admin will get the login information of a particular user.(5)The new user's data will be stored in the centralized database.(6)Admin get the search details of different criteria.(7)User can get his forgot password.

C. Input Design (III)

Input design is a part of overall system design. The main objective during the input design as given below

1) Input States (a):

The main input stages can be listed as below, Data recording, Data transcription, Data conversion, Data verification, Data control, Data transmission, Data validation, Data correction

2) Input Media (b):

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to type of Input, Flexibility of Format, Speed, Accuracy, Verification methods, Rejection rates, Ease of correction, Storage and handling requirements, Security, Easy to use, Portability

A source document differs from a turnaround document in that the former contains data that change the status of a resource while the latter is a machine readable document. Transaction throughput is the number of error-free transactions entered during a specified time period. A document should be concise because longer documents contain more data and so take longer to enter and have a greater chance of data entry errors. Numeric coding substitutes numbers for character data (e.g., 1=male, 2=female); mnemonic coding represents data in a form that is easier for the user to understand and remember. (e.g., M=male, F=female). The more quickly an error is detected, the closer the error is to the person who generated it and so the error is more easily corrected. An example of an illogical combination in a payroll system would be an option to eliminate federal tax withholding. By "multiple levels" of messages, means allowing the user to obtain more detailed explanations of an error by using a help option, but not forcing a lengthy message on a user who does not want it. An error suspense record would include the following fields: data entry operator identification, transaction entry date, transaction entry time, transaction type, transaction image, fields in error, error codes, date transaction re-entered successfully. A data input specification is a detailed description of the individual fields (data elements) on an input document together with their characteristics (i.e., type and length).

D. Error Messages to Be Displayed for the End User(C)

Be specific and precise, not general, ambiguous, or vague. (BAD:Syntax error, Invalid entry, General Failure). Don't JUST say what's wrong---- Be constructive; suggest what needs to be done to correct the error condition.

V. SYSTEM DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate an employee's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

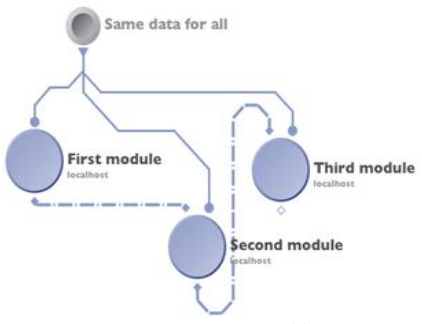

Fig. 1: Diagram(I)

VI. NORMALIZATION ON INFORMATION FLOW CONTROL WITH AZURE

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updating, deletion anomalies. Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

- Insertion anomaly: Inability to add data to the database due to absence of other data.
- Deletion anomaly: Unintended loss of data due to deletion of other data.
- Update anomaly: Data inconsistency resulting from data redundancy and partial update
- Normal Forms: These are the rules for structuring relations that eliminate anomalies.

A. First Normal Form:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

B. Second Normal Form:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules. Primary key is a not a composite primary key No non key attributes are present. Every non key attribute is fully functionally dependent on full set of primary key.

C. Third Normal Form:

A relation is said to be in third normal form if their exits no transitive dependencies.

Transitive Dependency (i): If two non-key attributes depend on each other as well as on the primary key then they are said to be transitively dependent. The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

VII. SYSTEM TESTING AND IMPLEMENTATION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

A. Strategic Approach to Software Testing (I):

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing will progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

VIII. ADVANTAGE

1) The cloud provider staff's may be granted access to perform maintenance on the database in order to provide technical support, which could causes privacy issues.
2) Permitting different parts of the IFC system to introduce new labels into the runtime system, and for the existing security policy to be enforced for these new labels as in the static case.
3) We believe that augmenting existing approaches to cloud security with DIFC is a promising way forward.
4) The cloud interface may have a less subtle and compressive view of access control than is required for the application.

IX. SECURITY SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls.

A. Client Side Validation (I)

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.

Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load. Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

B. Server Side Validation (Ii)

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.

- User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
- Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User- name, passwords and permissions are controlled o the server side.
- Using server side validation, constraints on several restricted operations are imposed

X. FUTURE ENHANCEMENT

For data security and privacy protection issues, the fundamental challenges are separation of sensitive data and access control. Our objective is to design a set of unified identity management and privacy protection frameworks across applications or cloud computing services. As mobility of employees in organizations is relatively large, identity management system should achieve more automatic and fast user account provisioning and de-provisioning in order to ensure no un-authorized access to organizations' cloud resources by some employees who has left the organizations.

XI. CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in ASP.NET and C#.Net web based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with "Evidence stream controller for protected haze". It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

REFERENCES

[1] Abstractions for Usable Information Flow Control in Aeolus Winnie Cheng_ Dan, R. K. Ports David Schultz, Victoria Popicy, Aaron Blanksteinz, James Cowling, Dorothy Curtis Liuba, Shrirax Barbara Liskov MIT CSAIL_IBM Research yStanford zPrinceton xBrandeis.
[2] An Information Flow Tool For Gypsy An Extended Abstract Revisited John Mchugh Cert/Cc R, Software Engineering Institute, Carnegie Mellon University E-Mail: Jmchugh@Cert.Org.
[3] Auditing Cloud Administrators Using Information Flow Tracking Afshar Ganjali A.ganjali@utoronto.ca, David Lielie@eecg.utoronto.ca, Department of Electrical and Computer Engineering University of Toronto.
[4] The Protection Of Information in Computer Systems Jerome H. Saltzer, Senior Member, Ieee, And Michael D. Schroeder, Member, Ieee.
[5] Securing Distributed Systems with Information Flow Control ,Nickolai Zeldovich, Silas Boyd-Wickizer, and David Mazi`eres Stanford University.
[6] Secure Information Flow and CPS Steve Zdancewic Andrew C. Myers Cornell University, Ithaca NY 14853, USA {zdance,andru}@cs.cornell.edu.
[7] Information Flow Control in Cloud Computing Ruoyu Wu1, Gail-Joon Ahn1, Hongxin Hu1, Mukesh Singhal21Laboratory of Security Engineering for Future Computing (SEFCOM), Arizona State University, Tempe, AZ 85287, USA 2Department of Computer Science, University of Kentucky, Lexington, KY 40506,USAEmail:{ruoyuwu,gahn,hxhu}@asu.edu, singhal@cs.uky.edu.
[8] Language-Based Information-Flow Security Andrei Sabelfeld And Andrew C. Myers Ieee Journal On Selected Areas In Communications, Vol. 21, No. 1, January 2003.
[9] JFlow: Practical Mostly-Static Information Flow Control Andrew C. Myers Laboratory for Computer Science Massachusetts Institute of Technology Proceedings of the 26th ACM Symposium on Principles of Programming Languages (POPL '99), San Antonio, Texas, USA, January 1999.
[10] T. Mather, S. Kumaraswamy, and S. Latif. Cloud Security and Privacy:An Enterprise Perspective on Risks and Compliance. Oreilly & associates Inc, 2009
[11] M. Vouk. Cloud computing Issues, research and implementations. Inv30th International Conference on Information Technology Interfaces, 2008. ITI 2008, pages 31–40, 2008.
[12] M. Vouk. Cloud computing Issues, research and implementations. In 30th International Conference on Information Technology Interfaces, 2008. ITI 2008, pages 31–40, 2008.
[13] D. E. Denning. A lattice model of secure information flow. Communications of the ACM, 19(5):236–243, May 1976.
[14] S. Tse and S. Zdancewic. Run-time principals in information-flow type systems. In Proceedings of the 2004 IEEE Symposium on Security and Privacy, pages 179–193. IEEE Computer Society, 2004.
[15] A. C. Myers. JFlow: Practical mostly-static information flow control.In Proceedings of the 26th ACM Symposium on Principles of Programming Languages, pages 228–241, 1999

