A NOVEL EVENT PROCESSING SYSTEMS IN LARGE-SCALE DELIVERED APPLICATIONS

Mallineni Brahmini[1], K. Praveen Kumar[2]
[1]*M.Tech, CSE Dept, MLRIT, Hyderabad*
[2]*M.Tech, Asst. Professor, MLRIT, Hyderabad*

***Abstract*— Current event processing systems lack methods to maintain solitude constraints of approaching event streams in a chain of consequently applied stream operations. This is a difficulty in large-scale delivered applications like a logistic chain where event processing operators may be expansion over multiple security domains. An opponent can conclude from legally received outgoing event streams classified input streams of the event processing system. In this paper we explained a exquisite access management for complex event processing. Each incoming event stream can be protected by the specification of an access policy and is enforced by algorithms for access consolidation. The utility of the event processing system is increased by providing and computing in a scalable manner a measure for the obfuscation of event streams. An obfuscation threshold as part of the access policy allows ignoring access requirements and delivering events which have achieved a sufficient high obfuscation level.**

***Index Terms*- Event Processing, Security, Access control.**

I. INTRODUCTION

In business processes, it is essential to detect inconsistencies or failures early. For example, in manufacturing and logistics processes, items are tracked continuously to detect loss or to reroute them during transport. To answer this need complex event processing (CEP) systems have evolved as a key paradigm for business and industrial applications [1], [2]. CEP systems allow to detect situations by performing operations on event streams which emerge from sensors all over the world, e.g. from packet tracking devices. While, traditionally event processing systems have applied powerful operators in a central way, the emerging increase of event sources and event consumers have raised the need to reduce the communication load by distributed in-network processing of stream operations [3], [4], [5], [6]. In addition, the collaborative nature of today's economy results in large-scale networks, where different users, companies, or groups exchange events. As a result, event processing networks are heterogeneous in terms of processing capabilities and technologies, consist of differing participants, and are spread across multiple security domains [7], [8]. However, the increasing interoperability of CEP applications raises the question of security [2]. It is not feasible for a central instance to manage access control for the whole network. Instead, every producer of information should be able to control how its produced data can be accessed. For example, a company may restrict certain information to a subset of authorized users (i.e. that are registered in its domain).

Current work in providing security for event-based systems covers already confidentiality of individual event streams and the authorization of network participants [9], [10], [11]. In CEP systems, however, the provider of an event loses control on the distribution of *dependent* event streams. This constitutes a major security problem, allowing an adversary to infer information on confidential ingoing event streams of the CEP system. As an example consider the logistics process illustrated in Figure 1 where a manufacturer wants to deliver an item to a destination. The shipping company determines a warehouse close to the destination, where the item will be shipped to before it will be delivered to the customer. The logistic process is supported by an event processing system, where operators are hosted in the domain of each party and exchange events including potentially confidential information (e.g. the item's *destination* is transmitted to the shipping company). If now a third party receives events related to the *warehouse*, it may draw conclusions about the original event data (i.e. *destination*), in spite of the manufacturer declaring this information as highly confidential and only providing the shipping company with access rights to it.

The goal of this work is to establish access control that ensures the privacy of information even over multiple processing steps in a multi-domain, large scale CEP system.

In particular, our contributions are i) an access policy inheritance mechanism to enforce access policies over a chain of dependent operators and ii) a scalable method to measure the obfuscation imposed by operators on

information exchanged in event streams. This allows defining as part of the access policy an obfuscation threshold to indicate when the event processing systems can ignore access restrictions, thus increasing the number of events to which application components can react to and this way increasing also the utility of the CEP system.

II. PROBLEM STATEMENT

Privacy-Preserving Public Auditing Module:

Homomorphism authenticators are unforgettable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Overview to achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphism authenticator with random mask technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF).

The proposed scheme is as follows:

- Setup Phase
- Audit Phase

Batch Auditing Module:

With the establishment of privacy-preserving public auditing in Cloud Computing, TPA may concurrently handle multiple auditing delegations upon different users' requests. The individual auditing of these tasks for TPA can be tedious and very inefficient. Batch auditing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side.

Data Dynamics Module:

Supporting data dynamics for privacy-preserving public risk auditing is also of paramount importance. Now we show how our main scheme can be adapted to build upon the existing work to support data dynamics, including block level operations of modification, deletion and insertion. We can adopt this technique in our design to achieve privacy-preserving public risk auditing with support of data dynamics.

Performance Requirements

Performance is measured in terms of the output provided by the application.

Requirement specifications play an important part in the analysis of system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

III. SYSTEM DEVELOPMENT

1. **Event Processing**
2. **Manufacturer**
3. **Shipping Company**
4. **Customer**

Event processing

Event processing systems respond to events in the system's environment or user interface. The key characteristic of event processing systems is that the timing of events is unpredictable and the system must be able to cope with these events when they occur.

Manufacturer

The manufacturer, insert the product details and also view product request from shipping company. Send details to shipping company to delivery date and pickup time.

Ship Company

The ship company, view product request from customer. Then company forward the request to manufacturer or reject the request.

Algorithm: Scalable Access Policy

```
procedure INITIALIZE(ω)
    for all operator ω do
        D_ω ← FINDMULTIPATHOPERATORS(ω)
    end for
    for all ω ∈ D_ω do
        relAtts ←FINDRELATEDATTRIBUTES
        for all (att_new, att_old) ∈ relAtts do
            TRANSMIT P(att_new|att_old) TO ω
        end for
    end for
end procedure

procedure UPONRECEIVEEVENT(e)
    for all att ∈ e do
        if ∃ multPathDependency(att) then
            CALCULATEWORSTCASEOBFUSCATION(ATT)
        else
            CALCULATELOCALOBFUSCATION(ATT)
        end if
    end for
end procedure
```

Customer

The customer, product order from Ship Company
and also views the order from Ship Company. Customer views the import details.

IV. RELATED WORK

Component based software systems are organized as a collection of hierarchically composed components each of which contains private data and operation implementations. Components communicate by sending messages to each other. Messages may be sent directly to another component or may be sent to an output port that is connected to the input port of one or more target components.

A message is a structured value with a name and a collection of arbitrarily complex data values. Different styles of message passing lead to different types of architecture. A *Service Oriented Architecture* (SOA) involves the publication of logically coherent groups of business functionality as *interfaces*, that can be used by components using synchronous or asynchronous messaging. An alternative style, argued as reducing coupling between components and thereby increasing the scope for component reuse, is Event Driven Architecture (EDA) whereby components are event generators and consumers. EDA is arguably more realistic in a sophisticated, dynamic, modern business environment, and can be viewed as a specialization of SOA where communication between components is performed with respect to a single generic event interface.

There are two important differences between SOA and EDA. Firstly EDA provides scope for *Complex Event Processing* (CEP) where the business processes within a component are triggered by multiple, possibly temporally related, events. In SOA there is no notion of relating the invocation of a single business process to a condition holding between the data passed to a collection of calls on one of the component's interfaces. Secondly, EDA can support dynamic extensibility through the introduction of new types of message both produced and consumed by a component. In general, SOA provides static interfaces that require architecture to be rebuilt when new services are introduced.

Service oriented architecture

Service Oriented Architecture (SOA) organizes a system in terms of components that communicate via operations or *services*. Components publish services that they implement as business processes. Interaction amongst components is achieved through *orchestration* at a local level or *choreography* at a global level.

Its proponents argue that SOA provides loose coupling, location transparency and protocol independence [4] when compared to more traditional implementation techniques. The organization of systems into coherent interfaces has been argued [16] as having disadvantages in terms of: extensions; accommodating new business functions; associating single business processes with complex multi-component interactions.

Enterprise architecture

Enterprise Architecture (EA) aims to capture the essentials of a business, its IT and its evolution, and to support analysis of this information: `[it is] a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems and infrastructure.'[11].

A key objective of EA is being able to provide a holistic understanding of all aspects of a business, connecting the business drivers and the surrounding business environment, through the business processes, organizational units, roles and responsibilities, to the underlying IT systems that the business relies on. In addition to presenting a coherent explanation of the *what*, *why* and *how* of a business, EA aims to support specific types of business analysis including [6, 14, 7, 12, 13]: *alignment* between business functions and IT systems; *business change* describing the current state of a business (*as-is*) and a desired state of a business (*to-be*);

maintenance the de-installation and disposal, upgrading, procurement and integration of systems including the prioritization of maintenance needs; *acquisition and mergers* describing the alignment of businesses and the changes that occur on both when they merge.

EA has its origins in Zach man's original EA framework [9] while other leading examples include the Open Group Architecture Framework (TOGAF)[15] and the framework promulgated by the Department of Defense (DoDAF)[13]. In addition to frameworks that describe the nature of models required for EA, modelling languages specifically designed for EA have also emerged. One leading architecture modeling language is ArchiMate[12].

A number of commercial EA analysis and simulation tools are available [16]. Man of these are based around industrial standards such as UML and BPMN. However they are generally very complex and lack a precisely defined semantics.

Event driven architecture

As described in Ref. [6] and Ref. [4], complex events can be the basis for a style of EA design. Event Driven Architecture (EDA) replaces thick interfaces with events

that trigger organizational activities. This creates the flexibility necessary to adapt to changing circumstances and makes it possible to generate new processes by a sequence of events [8]. Whilst a complex event based approach to architectural design must take efficiency concerns into account, the primary concern is how to capture, represent and analyses architectural information as an enterprise design.

EDA and SOA are closely related since events are one way of viewing the communications between system components. The relationship between event driven SOA and EA is described in Ref. [2] where a framework is proposed that allows enterprise architects to formulate and analyses research questions including `how to model and plan EA-evolution to SOA-style in a holistic way' and `how to model the enterprise on a formal basis so that further research for automation can be done.'

Complex event processing

Complex Event Processing (CEP) [12] can be used to process events that are generated from implementation-level systems by aggregation and transformation in order to discover the business level, actionable information behind all these data.

It has evolved into the paradigm of choice for the development of monitoring and reactive applications [7].

CEP can be viewed as a specialization of SOA where components are decoupled from multiple interfaces and where each component implements a single generic event interface. Components both raise and handle events in terms of this interface and therefore it is more flexible in terms of extension and maintenance. In addition, CEP implements events in terms of business rules compared to SOA that implements operations using business processes. Typically, a business rule can depend on multiple, possibly temporally related, events, whereas a business process is invoked on receipt of a single operation request. Therefore, SOA can implement CEP by enforcing a single operation interface across architecture and by providing special machinery to aggregate multiple operation calls.

The approach described in Ref. [9] is based on logic programming for complex event processing and in a way is the opposite to our forward-driven approach. The authors use Prolog-style backtracking to find solutions to goals.

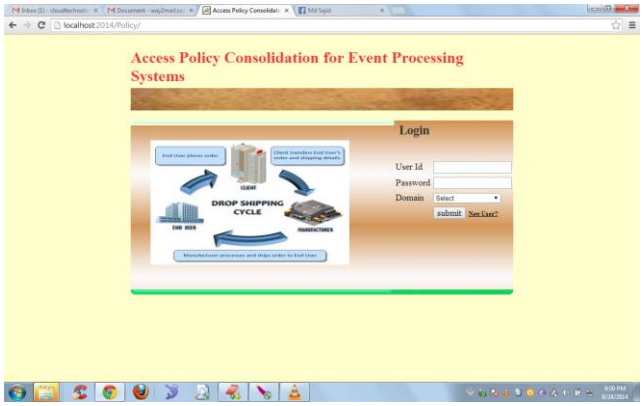
Component reconfiguration

Batista et al [5] identify two types of run-time reconfiguration in component based systems: *programmed* reconfiguration where changes can be foreseen at design time and *ad-hoc* reconfiguration are changes that cannot be predicted. The authors describe an ADL called Plastic that uses rules and reflection to reconfigure component connections at run-time. The system does not support the kind of ad-hoc reconfiguration described in this article whereby the behavior of an existing component can change without access to the implementation of the component.

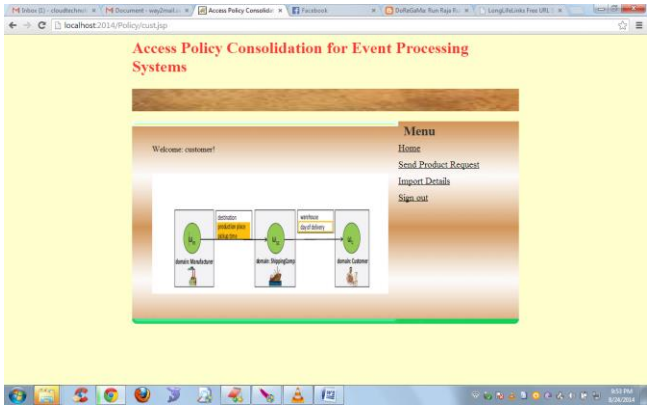
The dynamic reconfiguration approach described in this article is similar to the *mixin* approach of Frag[4] whereby new behavior is introduced by adding new super- classes and defining a method lookup mechanism. However we do not require specific language constructs for dynamically changing class based inheritance and therefore considers our approach to be at a finer level of granularity.

RESULTS

We implemented the presented approach within the DHEP framework which enables CEP in a heterogeneous environment. That means, hosts may be spread among different security domains and have differing processing capabilities or use different correlation engines. Hence, using the framework allows us to create multi-domain distributed CEP networks.



To achieve policy consolidation, every operator receiving a request provides the requester with the information needed for further processing: the access policy as well as the obfuscation policy.



V. CONCLUSION

This paper addressed the inheritance and consolidation of access policies in heterogeneous CEP systems. We identified a lack of security in multi-hop event processing networks and proposed a solution to close this gap. More specific, we presented an approach that allows the inheritance of access requirements, when events are correlated to complex events. Our algorithm includes the obfuscation of information, which can happen during the correlation process, and uses the obfuscation value as a decision-making basis whether inheritance is needed. We presented an implementation of our approach, based on Bayesian Network calculations.

The analysis and evaluations show that the approach is computation-intensive, once the Bayesian Network grows, hence rising the processing time of an event. To deal with the calculation cost, we introduced a local approach, where every participant calculates local obfuscation achieved during the correlation process. We use a variable elimination optimization to further reduce the computational effort for calculating obfuscation. Future work will concentrate on enhancing the obfuscation calculation and methods to increase the Bayesian Network size so we are able to measure obfuscation over more than one correlation steps.

REFERENCES

[1] A. Buchmann and B. Koldehofe, "Complex event processing," *it - Information Technology*, vol. 51:5, pp. 241–242, 2009.

[2] A. Hinze, K. Sachs, and A. Buchmann, "Event-based applications and enabling technologies," in *Proceedings of the Third ACM International Conference on Distributed*
Event-Based Systems, ser. DEBS '09. New York, NY, USA: ACM, 2009, pp. 1:1–1:15.

[3] P. Pietzuch, "Hermes: A scalable event-based middleware," Ph.D. dissertation, University of Cambridge, 2004.

[4] G. Li and H.-A. Jacobsen, "Composite subscriptions in
content-based publish/subscribe systems," in *Proc of the 6th Int. Middleware Conf.*, 2005, pp. 249–269.

[5] G. G. Koch, B. Koldehofe, and K. Rothermel, "Cordies: expressive event correlation in distributed systems," in *Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2010, pp. 26–37.

[6] B. Koldehofe, B. Ottenw¨alder, K. Rothermel, and U. Ramachandran, "Moving range queries in distributed complex event processing," in *Proc. of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2012, pp. 201–212.

[7] B. Schilling, B. Koldehofe, U. Pletat, and K. Rothermel, "Distributed heterogeneous event processing: Enhancing scalability and interoperability of CEP in an industrial context," in *Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2010, pp. 150–159.

[8] B. Schilling, B. Koldehofe, and K. Rothermel, "Efficient and distributed rule placement in heavy constraint-driven event systems," in *Proc. of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2011, pp. 355–364.

[9] M. A. Tariq, B. Koldehofe, A. Altaweel, and K. Rothermel, "Providing basic security mechanisms in broker-less publish/ subscribe systems," in *Proceedings of the 4th ACM Int. Conf. on Distributed Event-Based Systems (DEBS)*, 2010, pp. 38–49.

[10] L. I. W. Pesonen, D. M. Eyers, and J. Bacon, "Encryption-enforced access control in dynamic multidomain publish/subscribe networks," in *Proc. of the 2007 ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2007, pp. 104–115.

[11] J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch, "Access control in publish/subscribe systems," in *Proc. of the 2nd ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2008, pp. 23–34.

[12] M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel, "Meeting subscriber-defined QoS constraints in publish/subscribe systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 17, pp. 2140–2153, 2011.

[13] S. Rizou, F. D¨urr, and K. Rothermel, "Providing qos guarantees in large-scale operator networks," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, 2010, pp. 337 –345.

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 2nd ed.* Prentice Hall, 2002.

[15] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 721 –741, 1984.

[16] A. E. Gelfand and A. F. M. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

AUTHOR DETAILS:

First Author:MALLINENI BRAHMINI received B.Tech Degree in Computer Science and Engineering from Malla Reddy College of Engineering for Women in the year 2012. She is currently M.Tech student in the Computer Science and Engineering from MLR Institute of Technology. Her research interested areas are in the field of Networking, Cloud Computing and Mobile Computing.

Second Author: K.PRAVEEN KUMAR, working as an Asst. Professor in MLR institute of technology. He has completed his M.Tech CSE and he has 8 years of teaching experience. His research interested areas are Programming languages(C, C++, JAVA) Databases (DBMS, DMDW).