A Language Based Security Approach for Securing Map-Reduce Computations in the Cloud

Doaa Hassan*
Computers and Systems Department
National Telecommunication Institute
Cairo, Egypt
Email: doaa@nti.sci.eg

Raquel Hill
School of Informatics and Computing
Indiana University
Bloomington, Indiana
ralhill@indiana.edu

Abstract—We present a model for securing Map/Reduce computation in the cloud. The model uses a language based security approach to enforce information flow policies that vary dynamically due to a restricted revocable delegation of access rights between principals. The decentralized label model (DLM) is used to express these policies. We present a scenario that shows the flexibility and efficiency of our model in securing Map/Reduce computations running on the cloud infrastructure.

I. INTRODUCTION

Cloud computing [1] has been widely known for its potential to provide computing resources as services over Internet. In this paradigm, organizations offload their software services to cloud providers where users can rent virtual machines (VMs) from them instead of having to own the hardware themselves. At present, cloud computing has been facilitated by frameworks such as Map/Reduce [3]; a framework for processing large amounts of data in parallel by distributing jobs across large clusters. Hadoop [4] is an open-source implementation of Map/Reduce that has been widely used for large scale computation and data processing in the cloud computing paradigm. The distribution of jobs across nodes in Hadoop infrastructure creates an opportunity for the unauthorized sharing of data between nodes and processes operating on those nodes [2].

Recently, language based security [6] has been shown as a promising approach for enforcing an application’s information flow policy. In this approach, programming languages are used to specify and enforce information flow policies that implement confidentiality and integrity policies for each data element. However to our knowledge such an approach has not been used to model the information flow policies that vary dynamically due to a restricted delegation and revocation of access rights in the cloud computing infrastructure.

The goal of this paper is to enable temporary information flow by an authorized entity to sensitive information that has been uploaded to the cloud. To achieve this we investigate the possibility of using the restricted delegation/revocation (RDR) monad [5]. RDR monad is a library that implements DLM and allows mutually-distrusting principals to express individual confidentiality and integrity policies. It extends DLM to address the issue of restricted delegation and revocation of access rights. The RDR monad includes first-class references, higher order functions, and declassification/endorsement of policies. It also includes user authority in the presence of global unrestricted delegation using ActsFor relation (⪰), which states that principal p acts for principal q if any action taken by q can be performed by p. These features make it more suitable for usage to prevent insecure information flows in a huge distributed environment such as cloud computing. Also it enables a restricted form of a temporary dynamic delegation of access rights to a predefined chain of labels that can be later revoked by the data owner. The chain consists of a sequence of plain labels and an index pointing at the current active label in the sequence. Further delegations are only allowed if they are implied by the chain and, if so, they result in the index moving forward.

The RDR monad enables and enforces both static and dynamic information flow policies. Static policies are enforced by annotating each location, channel and values with a security label that specifies their owner and confidentiality policy. Dynamic policies are enforced by tracking the change in the dynamic security label, which consists of all label chains that were produced by the restricted revocable delegation of access rights to that value.

II. A CLOUD SCENARIO

In this scenario, we consider a large sales data set that has been outsourced to the cloud by a company. The company specifies policies that allow it to run various computations on the data. In addition there might also be various situations in which other entities are given access to the data: an accounting firm might be a permanent trusted delegatee that can act for the company, and an advertiser might be given restricted access to run some aggregate computations. Each of these possibilities requires different security parameters, but all can be easily expressed using RDR monad.

To make the presentation concrete, we consider that the company’s data will be stored on the NameNode of the cluster. The data is then distributed to the nodes where the map and reduce functions will be executed. Figure 1 illustrates the example. During this discussion, we focus on the labels attached to the data, where data stored on the NameNode has a label that states the company as the only owner and reader of that data. Moreover, the channels over which the data is distributed to the computation nodes have also that label.

Initially, the owner’s data is labeled with a policy $rp\{\top : owner\}$ to indicate that the owner is the only reader.
the data is partitioned and stored on various computation nodes, each segment gets an additional label indicating the “principal” controlling the computation node. For example, we might have two segments $S_1$ and $S_2$ with labels $\text{rp}\{\top : \text{owner}\} \sqcup \text{rp}\{\top : \text{node}_i\}$ where $i = 1, 2$. Following a conventional Map/Reduce idiom [3], local computations may run on each node, storing the intermediate results locally and returning a key to the owner. These results are also labeled $\text{rp}\{\top : \text{owner}\} \sqcup \text{rp}\{\top : \text{node}_i\}$ indicating that the data is shared. Since the resulting computation should be returned to the NameNode, it should be declassified by removing Node1’s label and leaving that node with just the label of company. In the standard Map/Reduce model, these keys are then given to the “reducers” so that they can collect the intermediate results, process them, and produce the final results. These reducers must be given an explicit delegation, i.e., they are allowed to “act for” the owner by the actsFor relation $\succeq$ which is set up in such a way that $\text{reducer}_i \succeq \text{company}$, and hence can access the data. Any principal trusted by the user can similarly be given a full delegation and perform the above computation on behalf of the user.

The more interesting situation is when the delegation needs to be restricted to an advertiser for example. Our generalized chains of labels allow us to express this situation as follows. At the point when the advertiser is given access, the owner retrieves the data, creates a version with a dynamic chain consisting of the label of the advertiser, and stores this back in the computation node. The advertiser can access the data as long as this dynamic chain is in place and will be denied access as soon as the chain is revoked. A crucial point is that the advertiser cannot exploit the temporary access given to him to leak the data to a public channel. Indeed the data is still tagged with the label of the owner and the owners policies are in full effect. Thus, if the advertiser tries to declassify the delegated value (granted to him by the owner) to the label of the output channel (the least restrictive label $\bot$) by removing his own policy, he cannot as the value is still tagged with the owner’s policy. The advertiser can declassify the data by removing his label, and send the results to the owner who can then examine these results and decide whether to allow their public release.

III. CONCLUSION AND FUTURE WORK

In this paper we present a new approach for securing Map/Reduce computations. Secure data flow is achieved using a language based security mechanism that enforces confidentiality policies on owner’s data. Moreover, our approach enables a restricted form of a revocable delegation among all entities involved in those computations.

For future work, we look forward to providing a real implementation of our approach as well as proving security properties of our model.

ACKNOWLEDGMENT

We are so grateful to Amr Sabry for his significant contribution to this work and his insightful comments.

REFERENCES