On Verifying Stateful Dataflow Processing Services in Large-Scale Cloud Systems

Juan Du, Xiaohui Gu, Ting Yu
Department of Computer Science, North Carolina State University
Raleigh, North Carolina, USA
jdu@ncsu.edu, {gu,yu}@csc.ncsu.edu

ABSTRACT

Cloud computing needs to provide integrity assurance in order to support security sensitive application services such as critical dataflow processing. In this paper, we present a novel RObust Service Integrity Attestation (ROSIA) framework that can efficiently verify the integrity of stateful dataflow processing services and pinpoint malicious service providers within a large-scale cloud system. ROSIA achieves robustness by supporting stateful dataflow services such as windowed stream operators, and performing integrated consistency check to detect colluding attacks. We have implemented ROSIA on top of the IBM System S dataflow processing system and tested it on the NCSU virtual computing lab. Our experimental results show that our scheme is feasible and efficient for large-scale cloud systems.

Categories and Subject Descriptors: H.4 [Information Systems Applications]: General

General Terms: Security, Management, Verification

1. INTRODUCTION

With rapid adoption of the concepts of Software as a Service (SaaS) and Service Oriented Architecture (SOA), cloud infrastructures have emerged as promising service provisioning platforms. Cloud infrastructures allow multiple users, called tenants, to lease computing resources from the cloud to run their applications. Thus, application service providers can conveniently use the cloud infrastructure to provide their software as services in a cost-effective way. In particular, our work focuses on dataflow processing services [4] that often demand high-performance continuous processing over data streams. Dataflow processing service has many real world applications such as security surveillance, scientific study, and business intelligence.

However, in open cloud systems, which consist of service providers from different security domains, we can no longer assume that all service components are trustworthy. For example, service components may include security holes that can be exploited by attackers. Attackers can also pretend to be legitimate service providers to compromise service delivery. In particular, it is challenging to ensure the *result integrity* of dataflow processing services. Compared with confidentiality and privacy concerns that have been addressed by previous research [7], the result integrity concern is the most prevalent, which needs to be addressed no matter whether public or private data are processed by the cloud system. Although previous work has provided various integrity attestation solutions [2, 8] for distributed systems, those techniques often require trusted hardware or secure kernel to be co-existed with the remote computing platform, which is difficult to be applied in cloud systems where service providers often autonomously control their computing environments and run proprietary software. To this end, the goal of our research is to develop practical service integrity attestation techniques that do not require application modifications or assume trusted entities on the third-party service provider site.

It is a challenging task to perform service integrity attestation for dataflow applications in cloud systems. First, dataflow applications require *continuous runtime* integrity attestation. Traditional Byzantine fault detection schemes [6] rely on full time majority voting on a set of service replicas over all input data, which fall short for cloud infrastructures in terms of scalability. In our previous work, we proposed RunTest, a replay-based randomized attestation scheme [3] that randomly selects a subset of input data to replay and performs result consistency check to detect malicious service providers. RunTest examines each service function individually and identifies nodes that fall outside of the maximum consistency clique as malicious. However, real world dataflow applications often include stateful service functions such as windowed stream operators [4]. In contrast to a stateless function whose output merely depends on the input, the output of a stateful function depends on both the input data and the state of the service component (e.g., previously received data in sliding-window stream operators). Thus, simply replaying the input will yield inaccurate integrity checking results. Second, multiple malicious service providers may launch strategic colluding attacks to the cloud system. For example, they can always give consistent wrong results and form a majority clique in a specific service function to trick the algorithm to label benign service providers as malicious. Thus, it is insufficient to only examine consistency relationships in individual service functions separately.

In this paper, we present a novel RObust Service Integrity Attestation (*ROSIA*) framework that can efficiently verify the integrity of stateful dataflow processing services and pinpoint malicious service providers within a large-scale cloud infrastructure. ROSIA achieves robustness in two major aspects. First, ROSIA supports replay-based consistency check for stateful data processing services. Second, ROSIA can pinpoint malicious service providers even when the system is under strategic colluding attacks. Specifically, this paper makes the following contributions: 1) We provide two safe consistency check schemes for stateful services. Our schemes do not require direct service state reset, which may alert malicious service providers about the integrity attestation and is impractical for implementation-specific state changes; 2) ROSIA performs comprehensive integrity attestation by examining both consistency and inconsistency relationships. This scheme can both

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

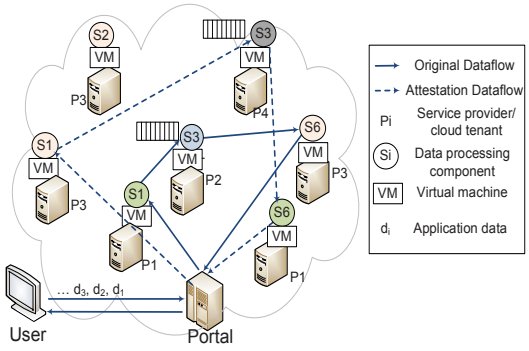

Figure 1: Multi-tenant cloud infrastructure.

achieve higher attack detection accuracy and limit the scope of the damage caused by colluding attackers; and 3) We have implemented the ROSIA system on top of the IBM System S stream processing system [4] and tested it on the NCSU virtual computing lab (VCL) [1]. Our experiment results show that ROSIA can efficiently verify the integrity of stateful dataflow applications while imposing low overhead to the cloud system.

2. SYSTEM MODEL AND ASSUMPTIONS

We now introduce the multi-tenant cloud system, illustrated by Figure 1. Each service provider p_i can lease a set of virtual machines (VMs) from the cloud system to provide one or more service components. A service component s_i is a self-contained software unit providing a certain service function f_i. Users can access the services by making a request through some portal node, who composes composite services from different service components according to the user's function specification [5]. The portal node accepts input data tuples d_i from the user, forwards to different service components for processing, and delivers final results to the user. A service function may be provided by different service providers.

Our work focuses on detecting service integrity attack to dataflow processing applications where a malicious service component may give untruthful data processing results. We assume that the total number of malicious service components is less than that of benign ones in the entire cloud system. But we do not assume benign service components have to be the majority for each specific service function as in [3]. Attackers could *sporadically collude*, which means an attacker can collude with an arbitrary subset of its colluders at an arbitrary time. Attackers can be *selective cheating*, which means they can misbehave on selective input data and/or on selective subset of service functions. These characteristics require that the detection scheme must be robust and scalable to capture unpredictable and occasional misbehavior.

3. DESIGN AND ALGORITHMS

Our algorithm pinpoints malicious service providers based on the consistency / inconsistency relationships between service providers. It includes three parts: 1) A runtime attestation scheme, called *replay-based consistency check*, to derive the consistency / inconsistency relationships between functionally equivalent service providers; 2) *Consistency graph* and *inconsistency graph* models to aggregate attestation results; and 3) A pinpointing algorithm that takes the attestation graphs as input and outputs malicious service providers.

Replay-based Consistency Check. The basic idea is to feed the same input data into functionally equivalent service components and compare output results to find out consistency/inconsistency relationships between service providers. Two service providers have consistency relationship if they always give consistent output results on all input data, or have inconsistency relationship if they give inconsistent outputs on at least one input data. Result consistency is defined as either result equality, or the distance between the results according to some distance function falling within a threshold. Note that we perform integrity attestation by replaying a subset of original input data at a later time. Thus, the malicious attackers cannot avoid the risk of being detected when they produce false results on the original data.

For stateless functions, given the same input data, two benign service providers will always return consistent output results. However, it is challenging to perform replay-based consistency check on stateful functions, such as windowed aggregation and join [4]. Even if the same input data is used for attestation, two benign service providers at different states may produce different results. Thus, for stateful functions, both input data and the states have to be replayed. We propose two methods to attest stateful functions. One method is called *indirect state recovery*, which relies on replaying a sequence of historic input data to indirectly bring back the state. For sliding-window stream operators, we record the data tuples sent to one service provider and resend them to another service provider to form the exact same window. The other method is called *difference check*, which derives consistency relationship between two stateful service components by comparing result difference produced by two consecutive input data. For example, the state can be a counter to to count the number of received tuples. Even though the counters of two service components may have different values, they both will increase by one when accepting a new input data.

In this paper, we focus on window-based stateful service functions, which is adopted by IBM System S [4]. We adopt a master-slave mechanism for data attestation. For each function, the portal randomly designates a service provider as the master and the rest as slaves. The portal sends all the data to the master service provider and sends only attestation data to the slave service providers. Specifically, the portal duplicates a data tuple d_i with a probability P_{dup}, and buffers a window size w of continuous tuples $d_i, d_{i-1}, ..., d_{i-w+1}$, which serve as the state on the master service provider. After the portal receives the processing result of d_i from the master, it sends the buffered window to a randomly selected subset of slave service providers. Results from the master and the slave service providers can then be compared to derive the consistency / inconsistency relationships.

Attestation Graph Model. Consistency/inconsistency relationships are stored in graphs for further analysis. For each service function, we maintain a consistency graph, which is an undirected graph, with all the attested service providers that provide the same service function as the vertex set and consistency relationship as the edges, as shown in Figure 2. Note that two service providers that are consistent in one function may be inconsistent in another function. Consistency graphs alone cannot efficiently reflect the relationship between service providers. Thus, we also maintain a global inconsistency graph, with all the service providers in the system as the vertex set and inconsistency relationship as the edges, as shown in Figure 3. The graphs reflect the consistency/inconsistency relationships across multiple service providers over time.

Pinpointing Malicious Service Providers. We leverage both per-function consistency graphs and global inconsistency graph to pinpoint malicious service providers. With consistency graphs, we adopt the method proposed in our previous work [3]. The intuition is that benign service providers always give the same correct results, thus can form a clique in the consistency graph. If the number of benign service providers is larger than that of malicious ones,

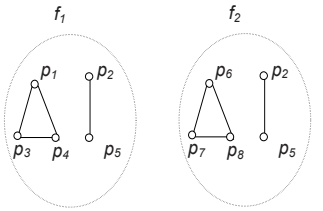

Figure 2: Per-function consistency graphs.

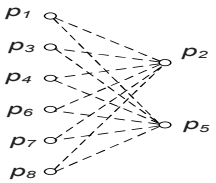

Figure 3: Global inconsistency graph.

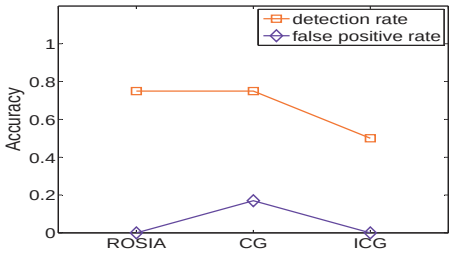

Figure 4: Comparison between ROSIA, the consistency graph (CG) scheme and the inconsistency graph (ICG) scheme.

we can identify nodes that fall outside of the maximum clique as malicious.

Note that malicious service providers can escape from being detected by trying to form a majority clique in the per-function consistency graph. However, if attackers try to maximize their damage by attacking multiple functions, we can still detect them by leveraging the global inconsistency graph. Intuitively, given two service providers that are inconsistent, we can claim that at least one of them is malicious. Thus, we can conclude that the number of malicious service providers in the inconsistency graph G should be no less than the size of the minimum vertex cover of the inconsistency graph, denoted by $|C_G|$. If we assume that the total number of malicious service providers in the whole system is no more than K, we can pinpoint those nodes that are definitely malicious by examining each individual node p in G [9]. The intuition is that if a node p is benign, its neighbors must be malicious. Then total number of malicious nodes, which is the sum of p's neighbor size $|N_p|$ and the number of malicious nodes in the residual graph after removing p and its neighbors from the inconsistency graph, should be no less than K. The latter has a lower bound equal to the minimum vertex cover of the residual graph, denoted by $|C_{G'_p}|$. Thus, node p is a malicious service provider if and only if $|N_p| + |C_{G'_p}| > K$. This method forces the attackers to limit the number of inconsistency links in order to escape from being detected.

Our algorithm finalizes the list of malicious service providers based on the results of both consistency graphs and inconsistency graphs. The idea is that any node identified through inconsistency graph is definitely malicious. If the node is also identified as malicious in a consistency graph, which may indicate that benign nodes form the majority in this function. Then the rest nodes that are outside of the majority clique may also be malicious. In summary, the consistency graph based pinpointing method forces malicious attackers to form majority in every service function they participate in, while the inconsistency graph based pinpointing method limits the number of functions malicious service providers can attack. By considering both consistency graphs and inconsistency graph, we can detect malicious service providers with a higher probability. The damage of malicious attacks is also bounded.

4. EXPERIMENTAL EVALUATION

We have implemented the ROSIA system in C++ on top of the IBM System S stream processing system [4], and deployed it on the NCSU virtual computing lab (VCL) [1], which consists of hundreds of blade servers and provides similar virtual resources as Amazon EC2. We compare our scheme with two other schemes: the *consistency graph* (CG) scheme and the *inconsistency graph* (ICG) scheme, which identify malicious nodes based on consistency graphs only and inconsistency graph only, respectively. Figure 4 shows one of our initial results, the advantages of ROSIA in terms of higher detection rate and lower false positive rate in certain attack scenarios.

5. CONCLUSION

In this paper, we have presented the design and implementation of ROSIA, a robust service integrity attestation system for processing stateful dataflow applications in cloud systems. ROSIA employs replay-based consistency check to efficiently verify the integrity of dataflow processing service components and pinpoint malicious service providers. ROSIA supports both stateless and stateful service functions and performs integrated analysis over both per-function consistency graphs and global inconsistency graph to effectively pinpoint colluding attackers. We have implemented ROSIA on top of the IBM System S stream processing system and tested it on the NCSU virtual computing lab. Our experimental results show that ROSIA is effective and imposes low overhead for dataflow processing in cloud infrastructures.

Acknowledgment: This work was sponsored in part by U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI), NSF CNS-0915567, and NSF IIS-0430166.

6. REFERENCES

[1] Virtual Computing Lab. http://vcl.ncsu.edu/.

[2] S. Berger, R. Caceres, and et. al. TVDc: Managing security in the trusted virtual datacenter. *ACM SIGOPS Operating Systems Review*, 42(1):40–47, 2008.

[3] J. Du, W. Wei, X. Gu, and T. Yu. Runtest: Assuring integrity of dataflow processing in cloud computing infrastructures. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2010.

[4] B. Gedik, H. Andrade, and et. al. SPADE: the System S Declarative Stream Processing Engine. *Proc. of SIGMOD*, April 2008.

[5] X. Gu, K. Nahrstedt, and et. al. QoS-Assured Service Composition in Managed Service Overlay Networks. *Proc. of ICDCS, 194-202*, 2003.

[6] T. Ho, B. Leong, R. Koetter, and et. al. Byzantine modification detection in multicast networks using randomized network coding. In *IEEE ISIT*, 2004.

[7] I. Roy, S. Setty, and et. al. Airavat: Security and privacy for MapReduce. In *NSDI*, April 2010.

[8] E. Shi, A. Perrig, and L. V. Doorn. Bind: A fine-grained attestation service for secure distributed systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2005.

[9] Q. Zhang, T. Yu, and P. Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM TISSEC*, 11(3), 2008.