Protecting Cloud Data Using the Decentralized Information Flow Control with Authorization Condition

Ye Jianwei, Xu Jie, Jiao Xulu, and Xu Zhikai

***Abstract*—In the extremely dynamic cloud computing system, traditional access control technologies provide no autonomic authorization and access control for the users on their data in remote cloud. Once data is migrated to the cloud, the user transfers the control to the providers of the cloud services and cloud hardware. So, whether the data is proper protected will be the users' most primary concerns and major challenges. This paper proposes a new decentralized information flow control model- DIFC-AC and its implementation. It expands the security label of DIFC with authorization condition used to express the control demands of the user, and access to the data is arbitrated based on their labels by intercepting IPC-relevant system calls. Thereby, the controls on the data are reached to the cloud, and sequentially the users' demands on the confidentiality, integrity and controllability of their data are meet.**

***Index Terms*—Access control, authenticity, cloud computing, confidentiality, decentralized information flow control.**

I. Introduction

Cloud computing is becoming one of the most development direction of IT field. Its concept that "the network is the computer", is attracting the users to migrate their complex computing, software, and data maintenance tasks to the cloud, and so as to reduces their hardware acquisition and maintenance cost. However, cloud also caused the serious worry about data security. According to the survey of IDC [1], 74.6% of the users said that their most concern about the cloud computing is the security issues. Recently, the various Data leakage accidents of Amazon, Google and other cloud computing sponsors encourage these worry. Therefore, the data security solution is the key of the popularization and development of cloud.

Access control is the most primary means on the data protection [2]. But, it are mainly applicable to single control domain, and so are not suitable for cloud computing, where the owners and processors of the data usually belong to various control domains and the owners will lost the control to their data once their data migrates to other control domains. Therefore, the core problem of the data protection in cloud is how the users can control their data in another control domain.

For the above problem, this paper proposes a new access control model-decentralized information flow control model with authorization condition (DIFC-AC). It annotates the data by security labels with authorization condition which express the confidentiality and integrity demands of the users, and the data access is arbitrated by intercepting IPC-relevant system calls. Thereby, the controls on the data are reached to the cloud.

This paper introduces the label model and implementation of DIFC-AC, and analyzes its security and performance overload.

Manuscript received October 4, 2014; revised December 30, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61100188.

Ye Jianwei is with Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China (e-mail: yejianwei@iie.ac.cn).

Xu Jie is with National Computer Network Emergency Response Technical Team Coordination Center of China, China (e-mail: xujie@cert.org.cn).

Jiao Xulu is with Information Center of Ministry of Industry and Information Technology of China, China (e-mail: jxl@miit.gov.cn).

Xu Zhikai is with Haerbin Institute of Technology, Heilongjiang, China (e-mail: zhikaixu@foxmail.com).

II. Related Work

So far, the traditional access control models are mainly based on the identities of the subjects and objects, such as MAC [3], DAC [4] and RBAC [5]. They are only suitable for single closed system, and thus are not applicable to open dynamic cloud computing system. Consequently, researchers proposed some improved models such as IRBAC2000, DRBAC, X-RBAC [6], ABAC [7], TBAC, TRBAC [8], etc. Unfortunately, they were only aimed at the specific aspects and cannot fundamentally solve the defect of the traditional models.

In order to refine data protection and support data flow, IFC [9] was put forward. Subsequently, Myers extended it to distributed computing system and presented DIFC [10]. Though, DIFC relatively best meet the data protection demands in cloud, its implementations [11], [12] unavoidably need to add extra codes for label setting and controlling. Hence, the difficulty of application development is greatly increased and the use of existing software is limited.

Our DIFC-AC expands the security label of DIFC with authorization condition and arbitrates access at standard operating system (OS) abstract, in order to spread the users' control capacity to the cloud without any modification to the existing cloud applications and software.

III. Label Model of DIFC-AC

A. Label

Every label is a set of security type which annotates the object's security level. There are two types of security type - the confidentiality type and the integrity type.

The confidentiality *type* (CType) of a object is defined by its owner to express the type of its confidentiality information. A CType c is a triple (c^t, c^+, c^-), where c^t is the symbol of c, c^+

and c^- are authorization condition. c^+ is the condition which the processors must meet for reading the object typed c. c^- is the condition which the processors must meet for declassificating typed c. C denotes the set of all CTypes.

The *integrity type* (IType) of a object is defined by its owner to express its integrity. An IType i is also a triple (i^t, i^+, i^-), where i^t is the symbol of i, i^+ and i^- are authorization condition. i^+ is the condition which the processors must meet for endorsing the object take type i. i^- is the condition which the processors must meet for giving up type i. I denotes the set of all ITypes.

A *label* L is composed of two sub-labels, *confidentiality sub-label* C_L and *integrity sub-label* I_L. $L=(C_L, I_L)$, $C_L \subseteq C$, $I_L \subseteq I$. L denotes the set of all labels, and $L=C^* \times I^*$. The label of object o is denoted as L_o.

For two confidentiality sub-labels C_{L1} and C_{L2}, $C_{L1} \subseteq C_{L2}$ express that C_{L2} includes more CTypes and has higher confidentiality than C_{L1}, therefore the object with C_{L1} can flow into another with C_{L2}, which meets the BLP model.

For two integrity sub-labels I_{L1} and I_{L2}, $I_{L1} \supseteq I_{L2}$ express that I_{L1} includes more integrity endorsement and has higher integrity than C_{L2}, therefore the object with I_{L1} can flow into another with I_{L2}, which meets the BIBA model.

Defining a relationship $\leqq: L \rightarrow L$. For $\forall L_1, L_2 \in L$, $L_1 \leqq L_2$ expresses the object with L_1 can flow into another with L_2. "$\rightarrow$" denotes the flow is allowed and "$\mapsto$" denotes the flow is forbidden. Thereby,

$$L_1 \leqq L_2 \Leftrightarrow C_{L1} \subseteq C_{L2} \wedge I_{L1} \supseteq I_{L2} \Leftrightarrow$$
$$\forall d,e, \text{ if } L_d=L_1 \wedge L_e=L_2 \wedge L_1 \leqq L_2 \text{ then } d \rightarrow e \quad (1)$$

B. Authorization Condition

An authorization condition (AC) is a set of predicates describing a processor and its surroundings. CN denotes the set of all ACs.

Defining a relationship $\succ: P \rightarrow CN$. For $p \in P$, $cn \in CN$, $p \succ cn$ expresses that all predicates in cn is true for the processor p, where P is the set of all processors. Furthermore, for $cs \subseteq CN$, $p \succ cs$ expresses p meets all condition of cs. Thereby,

$$\forall c \in C, p \succ c^+ \Rightarrow \forall d, \text{ if } C_{Ld} \subseteq (C_{Lp}+c) \text{ then } d \rightarrow p$$
$$p \succ c^- \Rightarrow \forall d, \text{ if } (C_{Lp}-c) \subseteq C_{Ld} \text{ then } p \rightarrow d \quad (2)$$

$$\forall i \in I, p \succ i^+ \Rightarrow \forall d, \text{ if } (I_{Lp}+i) \supseteq I_{Ld} \text{ then } p \rightarrow d$$
$$p \succ i^- \Rightarrow \forall d, \text{ if } I_{Ld} \supseteq (I_{Lp}-i) \text{ then } d \rightarrow p \quad (3)$$

For $\forall l \subseteq C \bigcup I$, l^+ denotes the set of all AC with $^+$ in l, and l^- denotes the set of all AC with $^-$ in l. Thereby,

$$d \rightarrow p \Leftrightarrow p \succ (C_{Ld}-C_{Lp})^+ \wedge p \succ (I_{Lp}-I_{Ld})^-$$
$$p \rightarrow d \Leftrightarrow p \succ (C_{Lp}-C_{Ld})^- \wedge p \succ (I_{Ld}-I_{Lp})^+ \quad (4)$$

The existing DIFC implementations usually do authorization by the codes based on the implied trust to authorized process. However, DIFC-AC does this by the authorization conditions which can include trusts also by trust predicates. DIFC-AC is stricter than other DIFCs in the authorizations, but is more suitable for the low-trust cloud.

IV. Implementation of DIFC-AC

Our implementation is at the level of standard OS abstract. It labels the standard descriptors (such as process, file, socket, etc.) and divides them into three categories - data, channel and process. Then, it sets a monitor at every OS kernel to arbitrate all accesses based on the labels by intercepting IPC-relevant system calls, and sets a user-mode porter to transfer label between cloud nodes.

A. Data

A data is a bit sequence representing special meaning, such as the file content, key, ciphertext, inter-process message, and so on.

B. Channel

A channel is a media used to transfer the data between the processes, including the pipe, shared memory, disk, socket link, etc. The channels are classified into two categories by whether the operation is controlled completely by the monitor - the controllable channels and the uncontrollable channels. The controllable channels mainly include all memory channels (such as shared memory, pipe, signal, message queue, etc.) and network channels between DIFC-AC hosts, which have not labels, and where the data uses itself label. The uncontrollable channels mainly include all hardware channels (such as hard-disks, cd-roms, printer, etc.) which set label $(\varnothing, \varnothing)$. The data d can flow into the uncontrollable channel c, if and only if $C_{Ld}=\varnothing$. Accordingly, the data in c can flow into the data d, if and only if $I_{Ld}=\varnothing$.

C. Process

A process is a runtime instance of the program. Suppose process p is the instance of program pm, the monitor arbitrates the behavior of p as follows.

1) When p is generated, $L_p=L_{pm}$.
2) When p loads library file l,
   if $p \succ (C_{Ll}-C_{Lp})^+ \wedge p \succ (I_{Lp}-I_{Ll})^-$
   then $l \rightarrow p$, $C_{Lp}=C_{Lp} \bigcup C_{Ll}$, $I_{Lp}=I_{Lp} \bigcap I_{Ll}$ else
   $l \mapsto p$
3) When p reads data d,
   if $p \succ (C_{Ld}-C_{Lp})^+ \wedge p \succ (I_{Lp}-I_{Ld})^-$
   then $d \rightarrow p$, $C_{Lp}=C_{Lp} \bigcup C_{Ld}$, $I_{Lp}=I_{Lp} \bigcap I_{Ld}$ else
   $d \mapsto p$
4) When p writes to data d,
   if $p \succ (C_{Lp}-C_{Ld})^- \wedge p \succ (I_{Ld}-I_{Lp})^+$ then $p \rightarrow d$ else
   $p \mapsto d$
5) When p writes data d into channel c,
   if c is controllable then $d \rightarrow c$, $L_d=L_p$
   else if c is uncontrollable $\wedge p \succ (C_{Lp})^-$ then $d \rightarrow c$,
   $d=(\varnothing, \varnothing)$
   else $d \mapsto c$
6) When p operates process q,
   if $p \succ (C_{Lq}-C_{Lp})^+ \wedge p \succ (I_{Lp}-I_{Lq})^-$
   then $q \rightarrow p$, $C_{Lp}=C_{Lp} \bigcup C_{Lq}$, $I_{Lp}=I_{Lp} \bigcap I_{Lq}$ else
   $q \mapsto p$
   if $p \succ (C_{Lp}-C_{Lq})^- \wedge p \succ (I_{Lq}-I_{Lp})^+$ then $p \rightarrow q$ else
   $p \mapsto q$

D. Prototype System

The prototype system is composed of an entry, a CA and a group of cloud nodes based on Linux kernel 2.6.9 and GT4.2. The topology is shown in Fig. 1 and the software architecture of the nodes is shown in Fig. 2.

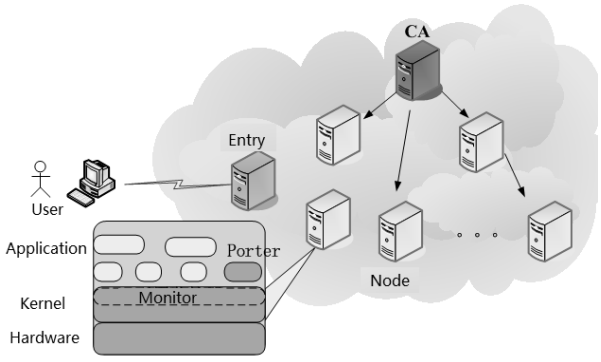

Fig. 1. Topology of DIFC-AC prototype system.

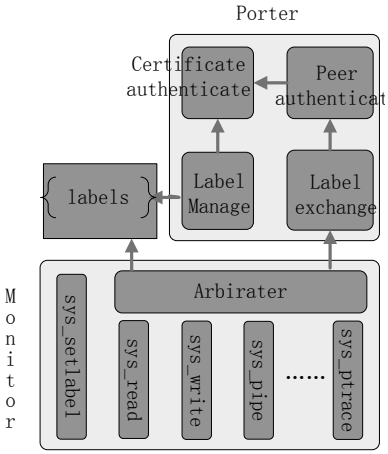

Fig. 2. Software architecture of DIFC-AC.

1) Monitor

A Monitor and a Porter are deployed in every node. Every monitor is a kernel module loaded and unloaded dynamically, and does three works.

a) Provide the user process a system-call *sys_setlabel* to set labels for the objects.

b) Modify the system-call table and insert an arbitrator before all system calls in the system to arbitrate all access by the foresaid rules. The arbitrator returns an error for a forbidden access, or returns the result of old system-call for a allowed access and manage the labels of involved objects. All involved system-calls is listed in Table I.

c) Cooperate with the upper Porter to exchange labels with other DIFC-AC nodes.

In order to save memory of the kernel and ease the communication with the Porter, the Monitor stores all labels in user space.

2) Porter

The Porter is a user process with multiple threads which does three works mainly.

a) Manage all Labels in the system.

b) Authenticate the remote nodes. Before transmitting a labeled data to or receiving a labeled data from a new remote node, the Porter must authenticate the node by its host certificate and send result to the Monitor.

c) Cooperate with the Monitor to exchange encrypting key with the remote nodes and transmit labeled data to them.

3) Transmission of labeled data

For data security in the whole lifecycle, the data and its label must be sent to other DIFC-AC host together. Since the labels are denoted at the OS abstract, they are not real parts of the labeled objects. To simplify the implementation, a single TCP link is used to transmit the labels. The procedure that node A sends a labeled data *d* to host B lists as follow.

a) A TCP link *l* is established to transmit *d* between A and B.

b) Before sending *d* into *l*, the Monitor of A checks the result of authentication to B. If the authentication is incomplete, the Porter is requested to complete it. If the result is fail, then the sending is refused and *l* is closed.

c) In A, the Monitor creates a symmetric key *k*, block size *s*, and request Porter to send *k*, *s* and L_d to the Porter of B.

d) The Porter of A builds a SSL link to the Porter of B and sends *k*, *s* and $L_{d.}$

e) In B, once the Porter receives *k*, *s* and $L_{d.}$, it notices the Monitor to modify the label of socket descriptor of *l* to $L_{d.}$, records *k*, *s* and send a successful response to A.

f) The Porter of A notices its Monitor the results of above two steps.

g) If the Monitor of A gets successful result, it splits *d* to blocks with size *s*, encrypts those blocks with *k*, and writes them into *l*. Otherwise, *l* is closed and failure is returned.

h) In B, the Monitor decrypts received data blocks with *k* to *d* and sets *d* as the output of *l*.

i) Once the transmission completes, *l* is closed.

j) If L_d changes in the transmission, step d and e must be repeated to send new L_d.

V. Security Analysis

The security of DIFC-AC relies on the security of all involved Monitors, Porters, kernels, hardware, data flow, authorization and access control.

A. Trusted Computing Base

The trusted computing base of DIFC-AC is the security of all involved Monitor and Porter whose security relies on their kernels and hardware. The trusted computing technologies based on TPM can be used to secure them.

B. Security of Data Flow

In DIFC-AC, all subset of label set L form grid in partial-order relationship "$\leqq$". Therefore, all data flow meet the BLP and BIBA model, and don't destroy data security, as long as they meet the formula (1).

Meanwhile, DIFC-AC controls the uncontrollable channels strictly, where only minimum security data (readable and writable to everyone) can be transmitted.

So, the security of data flow is enough.

C. Authorization and Access Control

Authorization and access control is key of DIFC-AC. They are secure if and only if the labels are secure and the Monitor

can exactly control all data access by the labels and all foresaid formulas.

For the security, all type symbols in the labels must be difficult to guess, if not the attackers can destroy the labeled data, using false data with same labels. Proper random-digit algorithms can be used to create secure labels.

Once the labels are created, they flow in controllable channel and system processes supervised by the Monitor. So, they are safe.

The security of access control can be guaranteed if the Monitor arbitrates all data access by formula (4).

VI. Experiment Result

The main overhead of our implementation dues to the additional instructions interposed on the monitored system calls by the monitor. We evaluate its performance impact by experiments. In all experiments, the cloud nodes running Linux version 2.6.9 and GT4.2 with and without the monitors are of single CPU, single-core 1.4GHz Pentium-M. The system call latencies are shown in Table I. For most system calls, the monitor adds 0.5-4 μs per system call which results in latency overhead of a factor of 0.2%-170%.

TABLE I: Microbenchmarks of the Monitored System Calls and Reference Monitor Overhead (Latencies Are in ns.)

System call	No monitor	monitor	overhead	precent
clone	90,273	94,586	4,313	4.78%
fork	30,356	35,168	4,812	15.85%
vfork	20,110	25,904	5,794	28.81%
execve	144,148	165,837	21,689	15.05%
ptrace	831	3,194	728	284.35%
wait4	6,675	7,149	474	7.10%
waitpid	1,425,980	1,490,973	64,993	4.56%
waitid	3,296	3,982	686	20.81%
open	6,872	11,843	4,971	72.34%
creat	21,202	27,913	6,711	31.65%
read	1,016,028	1,018,576	2,548	0.25%
sys_write	6,708	8,893	2,185	32.57%
sys_readv	6,096	8,401	2,305	37.81%
writev	3,991	5,947	1,956	49.01%
unlink	15,179	15,956	777	5.12%
mmap	3,799	7,101	3,302	86.92%
mmap2	1,674	4,795	3,121	186.43%
munmap	3,329	4,294	965	28.99%
msync	19,35	3,919	1984	102.50%
sendfile	1,546	2,192	646	41.79%
socketcall	6,819,423	8,955,493	2,136,070	31.32%
ipc	1,082	3,287	2,205	203.79%
pipe	3,721	5,614	1,893	50.87%
mq_timedsend	1,013	1,990	977	96.45%
mq_timedreceive	920	2,505	1,585	172.28%

VII. Conclusion

The DIFC-AC can extend the users' control on their data to the cloud, provide autonomous control ability to the users, and make the data be controllable. Compared to DIFC, its greatest advantage is that authorization condition is used to substitute code command, thus ease the software development and using of existing software. Meanwhile, the control based on authorization condition more meet characteristics of cloud computing. The experiment results show the overhead of DIFC-AC is acceptable.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 61100188).

References

[1] IDC. (2013). IT Cloud Services User Survey, pt.2: Top Benefits & Challenges. [Online]. Available: http://blogs.idc.com/ie/?p=210

[2] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, pp.1278-1308, 1975

[3] X. Qian and T. E. Lunt, "A mac poliey framework for multilevel relational database," *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, pp. l-14, 1996.

[4] R. Sandhu and Q. Munawer, "How to do discretionary access control using roles," in *Proc. the 1998 3rd ACM Workshop on Role-Based Access Control*, 1998, pp. 47-54.

[5] D. F. Ferraiolo, R. S. Sandhu, and S. Gavrila, "Proposed NIST standard for role-based access control," *ACM Trans. on Information and System Security*, vol. 4, pp. 224-274, 2011.

[6] J. B. D. Joshi, R. Bhatti, and E. Bertino, "Access-control language for multidomain environments," *IEEE Internet Computing*, vol. 8, pp. 40-50, 2004.

[7] R. F. Han, H. X. Wang, and Q. Xiao, "A United access control model for systems in collaborative commerce," *Journal of Network*, vol. 4, pp. 279-289, 2009.

[8] E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: a temporal role-based access control model," *ACM Trans. on Information and System Security*, vol. 4, pp. 191-233, 2000.

[9] D. E. Denning and P. J. Denning, "Certification of programs for secure information flow," *Communications of the ACM*, vol. 20, pp. 504-513, 1977.

[10] A. C. Myers and B. Liskov, "Protecting privacy using the decentralized label model," *ACM Trans. on Computer Systems*, vol. 9, pp. 410-442, 2000.

[11] N. Zeldovich, S. B. Wickizer, and D. Mazi`eres, "Securing distributed systems with information flow control," in *Proc. 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 293-308.

[12] J. Liu, M. D. George, and K. Vikram *et al.*, "Fabric: a platform for secure distributed computation and storage," in *Proc. the 22nd ACM SIGOPS Symposium on Operating Systems Principles*, 2009, pp. 321-334.

Ye Jianwei was born in 1978, Zhejiang China. He received the degree of B.E. in computer science in 2001 from Harbin Institute of Technology, Heilongjiang China. The degree of M.E. in computer system structure was taken in 2003 from Harbin Institute of Technology, China. And the degree of D.E. in computer structure was obtained in 2011 from Haerbin Institute of Technology, China. The major field of study includes distributed computing, network security and information security.

He worked at Harbin Institute of Technology (Harbin, China) as a lecturer from 2003 to 2012. Then, he worked at Institute of Information Engineering of Chinese Academy of Sciences (Beijing, China) as a senior engineer since 2013. He has published more than 20 papers. His research interests include distributed computing, cloud computing, grid, computer network security, information security, etc.

Xu Jie was born in 1982, Shanxi, China. His degree of D.E. in signal process and multimedia was obtained in 2013 from Institute of Computing Technology, Chinese Academy of Sciences, China. The major field of his study includes signal process, network security and information security.

He worked at National Computer Network Emergency Response Technical Team Coordination Center of China (Beijing, China) as an engineer since 2013.

Jiao Xulu was born in 1978, Heilongjiang China. Her degree of B.E. in computer science was taken in 2001 from Harbin Institute of Technology, Heilongjiang China. The degree of M.E. in computer system structure was taken in 2003 from Harbin Institute of Technology, China. Her major field of study includes distributed computing, network security and information security.

She worked at National Computer Network Emergency Response Technical Team Coordination Center of China as an engineer from 2003 to 2010. Then, she worked at Information Center of Ministry of Industry and Information Technology of China as a senior engineer since 2010.

Xu Zhikai was born in 1988, Shandong China. The degree of B.E. in computer science was taken in 2010 from Harbin Institute of Technology, Heilongjiang China. His degree of M.E. in computer system structure was taken in 2012 from Harbin Institute of Technology, China. His major field of study includes distributed computing, cloud computing. He is working for his degree of D.E. since 2012 in Harbin Institute of Technology, China.