POSTER: Towards Formal Verification of DIFC Policies

Zhi Yang[1,2], Lihua Yin[1], Miyi Duan [1], Shuyuan Jin[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

[2] Institute of Electronic Technology, Information Engineering University, ZhenZhou 450004,China

{yangzhi, yinlihua, duanmiyi, jinshuyuan}@software.ict.ac.cn

ABSTRACT

Decentralized information flow control (DIFC) is a recent important innovation with flexible mechanisms to improve the availability of traditional information flow models. However, the flexibility of DIFC models also makes specifying and managing DIFC policies a challenging problem. The formal policy verification techniques can improve the current state of the art of policy specification and management. We show that in general these problems of policy verification of the main DIFC systems are NP-hard, and show that several subcases remain NP-complete. We also propose an approach of model checking to solve these problems. Experiments are presented to show that this approach is effective.

Categories and Subject Descriptors

D.4.6 [**OPERATING SYSTEMS**]: Security and protection – *Access controls, Information flow Controls*

General Terms

Security, Verification

Keywords

DIFC, verification, formal method, NP-hard, model checking

1. INTRODUCTION

The classical information flow control (IFC) has often been overly restrictive, preventing useful systems from being built. It intends that a single administrator sets all the system-wide policies. When a system run multiple third-party applications, it is difficult for the administrator to understand every application's detailed security requirements. DIFC [1]promises to support such situations by allowing processes to transfer privileges to others and declassify information in a decentralized way.

However, realistic policies do not just and simply fall into the DIFC models. A perhaps most important challenge for information-flow security stems from the difficulty of managing complex security policies. Administrators need to not only understand what the desired security policies are, but know how to formalize those policies using annotations. Moreover, DIFC permits subjects to change the authorization states dynamically by manipulating their own labels or others subject's labels within their capabilities. These mechanisms often make administrators lack confidence in whether the resulting policy configurations indeed enforce the desired objectives.

Thus, to protect secrecy and integrity of systems, a fundamental question of DIFC safety is to answer the following: "Given the current authorization state and the policy specification, will information can ever flow from a subject (object) to another subject (object)?" It is an analysis of information propagation. Such security analysis aims to check whether the system maintains desirable security properties in all the states. The security properties are formulated from a set of access control policies. A formal verification technique then can be used to ensure the consistency of an access control specification. Security analysis provides answers to the queries whether an undesirable state is reachable or all the reachable states satisfy the desirable properties.

There has been previous work on verification of DIFC systems. Chaudhuri et al. [8] presented a logic-programming language and tool, called EON, to model and analyze dynamic access control models of the Asbestos and Windows Vista systems. Query evaluation in EON can be reduced to decidable query satisfiability in a fragment of Datalog. Harris et al. [9] presented a method of abstraction that allows a model checker to verify policy soundly for DIFC programs that manipulate potentially unbounded sets of processes. Their approach focuses on automatically extracting and checking models of applications. Harris et al [10] also studied how to instrument DIFC code automatically, given only an uninstrumented program and some policies. Nevertheless, their approaches apply only to subcases without transferring-privileges. In addition, few of these studies analyze the computational complexity of DIFC policy verification.

In this work, we study the computational complexity of policy verification of Asbestos [2], HiStar[3] and Flume[4]. Further, we study the factors that contribute to the complexity..

2. PROBLEM Complexity

2.1 Definition of DIFC Policy verification

As mentioned above, decentralized declassification and transferring-privileges complicate the information flow analysis. Since declassifying information (downgrading the sensitivity labels on the data) may violate traditional information flow policies, in general its use is limited to appropriately trusted subjects. However, it is difficult to implement this addition. Firstly, there is the potential for the channel to be abused to release more sensitive information than that intended [5][6]. For example, most login programs of operation systems will return the login result whether the visitor supplies right password or error password. It is a kind of declassification. However, an internal user can encode the sensitive information of the system into the

This work was supported by the National Natural Science Foundation of China under Grant No. 61070186

Copyright is held by the author/owner(s).
CCS'11, October 17–21, 2011, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

password and release it to external users through the channel [5]. Secondly, in some cases, administrators have to give some subjects the ability of declassifying on account of availability other than these subjects being trust. The third one is in DIFC systems, subjects can independently grant their own abilities, including declassification capabilities, to other subjects. It is very difficult to guarantee the trust of these new declassifiers when reliable trust evaluation mechanisms are absent. Therefore, our analysis assumes that declassifiers may release any information they know and information flows are only subject to the constraints of the DIFC rules.

Therefore, given an DIFC system, for arbitrary process p, q, we say p can reach q if and only if there exists a path $< v_1, v_2, ..., v_n >$, where $v_1 = p$, $v_n = q$, and for each pair of process v_i and v_{i+1}, $1 \le i \le n-1$, not only the information flow from v_i to v_{i+1} is legal in terms of DIFC rules, but also v_i and v_{i+1} are ready for the communication link.

The problem of reachability analysis is a decision problem.

2.2 Complexity of the Problem for Flume Policy Verification

Flume uses *tags* and *labels* to track data as it flows through a system. Let T be the set of all the possible *tags*. *Labels* are subsets of T. Each process P maintains a secrecy label S_P, an integrity label I_P, and a capability set O_P. For tag t, capabilities are denoted t^+ and t^-. A process P with $t^+ \in O_P$ can add t to its label. A process P with $t^- \in O_P$ can remove t from its label.

In Flume, for a process P, let the label L be S_P or I_P, and let L' be the requested new value of L. The change from L to L' is *safe* if and only if: $L' - L \subseteq O_P^+$ and $L - L' \subseteq O_P^-$, where $O_P^+ = \{t \mid t^+ \in O_P\}$, $O_P^- = \{t \mid t^- \in O_P\}$. The information flow from p to q is safe if and only if $S_p - D_p \subseteq S_q \cup D_q$ *and* $I_p \cup D_p \supseteq I_q - D_q$, where $D_P = \{t \mid t^+ \in O_P \wedge t^+ \in O_P\}$

.Further, a process P can grant capabilities in O_P to process q so long as P can send a message to Q. P can also subtract capabilities from O_P as it sees fit.

According to the problem definition and the Flume model definition, we have the following conclusions.

Theorem 1 *The problem of policy verifications of Flume system is NP-hard.*

That processes grants their abilities to others seems to be a main factor that causes the exponential-time complexity, while the next theorem shows that it still takes exponential time, preventing processes from granting their abilities to others.

Theorem 2 *The problem of policy verifications of Flume system without processes granting their abilities to others is NP-complete.*

We use the reduction method to prove these two theorems. The problem used to reduce from is 3SAT problem [7], and it is NP-complete. We omit the proof due to space constraints.

Theorem 3 *The problem of policy verifications of Flume system without declassification is in P.*

Theorem 3 indirectly shows that the subject's capabilities for deleting secrecy tags or adding integrity tags from its own levels is the key factor that causes the exponential-time complexity. Such system without declassification is monotonic, where the process's secrecy label change is non-decreasing and the process's integrity label change is non-increasing, which means that processes will soon be unable to write other objects. In practice, in order to decrease the complexity, we should be discreet in entitling subjects to declassification abilities.

2.3 Complexity of the Problem of Asbestos/ HiStar Policy Verification

In Asbestos, each process P has two labels, a *send label* P_S and a *receive label* P_R. Sending a message from process P to process Q requires that: $P_S \subseteq Q_R$ .Once communication, Q is tainted by P: $Q_S \leftarrow Q_S \cup P_S$. Asbestos categories are named by *handles*. Handle privileges are represented by *levels*, which are members of the ordered set [$\star$, 0, 1, 2, 3], with minimum $\star$ and maximum 3. A label L is a mapping from handles to levels. Labels form a lattice under the partial order relation $\subseteq$. $L_1 \subseteq L_2 \Leftrightarrow \forall h$, $L_1(h) \le L_2(h)$, where L_1 and L_2 range over labels, h over handles. Operators $\bigcup$ and $\bigcap$ are defined as follows:

$$(L_1 \bigcup L_2)(h) = \max(L_1(h), L_2(h)), (L_1 \bigcap L_2)(h) = \min(L_1(h), L_2(h)).$$

. Asbestos decentralizes declassification using the special $\star$ level. Further, a process P with declassification privilege for handle h can decontaminate a processes Q's labels with respect to h by lowering Q's *send* labels and raising Q's *receive* labels, so long as P can send a message to Q.

HiStar tracks and enforces information flow using Asbestos labels, while it uses the explicit label adjustment to replace the implicit label adjustment used in Asbestos.

According to the problem definition and the Flume model definition, we have the following conclusions.

Theorem 4 *The problem of policy verifications of Asbestos/HiStar system is NP-hard.*

Theorem 5 *The problem of policy verifications of Asbestos/HiStar system without processes granting their abilities of* declassification *to others is NP-complete.*

We still use the reduction method to prove these two theorems. We respectively reduce the problem in Theorem 1 to the problem in Theorem 4, and reduce the problem in Theorem 2 to the problem in Theorem 5.

Theorem 6 *The problem of policy verifications of Flume system without declassification is in P.*

The proof of Theorem 6 is similar to that of Theorem 3.

3. MODEL CHECKING APPROACH FOR DIFC POLICY VERIFICATION

Modeling the problem We use the model checking approach to analyze some instances of these problems. In this approach, a system property is expressed in temporal logic and a state transition system is modeled as a Kripke structure[11]**.** The structure is a 4-tuple $M = (S, S_0, R, L)$ with a finite set of states S, a set of initial states S, a transition relation $R \subseteq S \times S$, and a labeling of the states $L : S \to 2^{AP}$ with subsets of a set of atomic

propositions *AP*. In the context of DIFC policy verification, each state should involve the current labels, the capabilities of all the processes. The state transitions are triggered by the information flows that satisfy the DIFC rules and the specified communication links. For a problem instance of whether there exists a path from a processes p to a processes q, we introduce a Boolean variable $ta\,\mathrm{int}_r$ for each processes r to record whether r has been contaminated by the information flow originated from p in the current state. Then, we can use a property in CTL with the form "$!EF\ ta\,\mathrm{int}_q$" or in the LTL with the form "$G\,!ta\,\mathrm{int}_q$"to express the decision problem. The property asserts that there exists no state where the variable $ta\,\mathrm{int}_q$ becomes true.

Preprocessing: Some processes may be irrelevant to the verification process. We use a preprocessing stage to remove these processes and thus can reduce the size of the search space. Take Flume for example. Firstly, we remove processes that haven't any ability. Secondly, we pick out the processes that only have the abilities of adding tags, and let them grant all their capabilities to q on condition that the information flow from them to q is legal, then remove these processes. Thirdly, we remove processes whose abilities of removing tags are irrelevant to the problem. We design an algorithm to find these processes.

Experimental results: Take Flume for example again, we first create an automatic generator of instances, and use the *capability settings parameter* to control the probability of processes owning declassification capabilities. Instances are divided into eight groups, i.e., G1, G2…, and G8. We analyze three kinds of runtime. *Runtime-A* is obtained when the tested instances are normally executed. *Runtime-B* is obtained when the tested instances are preprocessed. *Runtime-C* is obtained when no transferring privileges is permitted. Experiments were performed using NuSMV 2.5.2 on a Windows XP with an Intel P4 2.8G CPU and 2GB of memory. The experiment results are shown in Table 4, where N/A indicates that at least one instance in a group cannot be solved within 30 minutes.

Table 1 algorithm performance under different parameter values

	G1	G2	G3	G4	G5	G6	G7	G8
Num of processes	10	20	40	40	80	120	160	200
Num of Tags	4	6	8	8	15	20	25	30
Capability settings parameter	1	1	1	3	2	2	2	2
Num of processes after preprocessing	6	15	34	13	68	91	132	167
Runtime-A(s)	2	8	50	33	280	1080	N/A	N/A
Runtime-B(s)	2	6	42	10	215	650	1565	N/A
Runtime-C(s)	1	3	18	15	60	100	300	1200

Table 1 shows that although the problem complexity is exponential-time, our program can effectively solve nontrivial-sized instances in reasonable amounts of time, and the preprocessing step plays an important role in improving the efficiency. Meanwhile, slightly change of num of declassification privileges influences the computational time remarkably. And transferring privileges results in the double runtime.

4. CONCLUSIONS

We have shown that in general the problems of policy verification of Asbestos, HiStar and Flume are NP-hard. We have proposed an approach of model checking to solve the problem.

5. REFERENCES

[1] A. C. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410-442, 2000.

[2] P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazieres, F. Kaashoek, and M. Robert. Labels and event processes in the asbestos operating system. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*, pages 17-30. ACM, 2005.

[3] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazieres. "Making information flow explicit in HiStar," in *Proceedings of Usenix Association 7th Usenix Symposium on Operating Systems Design and Implementation)*, pages 263-278. USENIX Association, 2006.

[4] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris, Information flow control for standard OS abstractions. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles*, pages 321-334. ACM, 2007.

[5] S. Zdancewic and A. C. Myers. Robust declassification. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 15-23. IEEE Computer Society, 2001.

[6] A. Sabelfeld and D. Sands. Dimensions and Principles of Declassification. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, pages 255-269. IEEE Computer Society, 2005.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman & Co., 1979.

[8] A. Chaudhuri, P. Naldurg, S. K. Rajamani, G. Ramalingam, and L Velaga. EON: Modeling and Analyzing Dynamic Access Control Systems with Logic Programs. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 381-390. ACM, 2008.

[9] W. R. Harris, N. A. Kidd, S. Chaki, S. Jha, and Reps T. Verifying information flow control over unbounded processes. In *Proceedings of 16th International Symposium on Formal Methods*, pages 773-789. Springer, 2009.

[10] W. R. Harris, S. Jha, and T. Reps. DIFC Programs by Automatic Instrumentation. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 284-296. ACM, 2010.

[11] A. Biere, A. Cimtti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. *In Proceedings of the 36th Annual Conference on Design Automation*. Pages 317-320. IEEE Computer Society, 1999.