ISSN:2790-1688 DOI: 10.56028/aetr.2.1.560

Research on a Dynamic Trusted Model Based on Time-sharing Control

Lixin Li[1, a], Cong Yu[1, b], Mengmeng Yang[1, c]

[1]Third academy，Information Engineering University，Zhengzhou 450001, China;

[a]alpha37266@163.com, [b]tieshan0216@163.com, [c]youngdream0215@163.com

Abstract. In this paper, a high-availability time-sharing dynamic trusted model is proposed, which controls the access of untrusted domains to trusted domains by time-sharing, dynamically adjusts the trusted domain and untrusted domains, and proves their security with non-interference theory, which improves the usability of the dynamic trusted model.

Keywords: Time-sharing control; No interference; Dynamic adjustment; Trusted computing.

1. Introduction

Trusted computing is an important theoretical achievement in the field of security. "Trusted"[1] in an information system means predictability, that is, the information system will operate in the way people expect. The predictable behavior of the system does not mean that the system is safe, because the expected behavior is not only the normal behavior, but also the risk can be expected. The non-disturbance theory[2] was proposed by Goguen and Meseguer in 1982. The main idea is that if an operation in one domain u does not change the security state of another domain v, then u is considered to be non-disturbing to v.

Rushby proposed a non-interference model[3], defined and formally described the model, and proposed a non-interference theorem for processes and systems. Most of the follow-up research is based on this model. Chen Liang[4] analyzes and determines the trust chain transfer process in the trusted computing platform. If the system complies with the conditions of the non-transitive non-interference security policy, the information flow between components will be constrained by the security policy, and the integrity measurement method is considered to be effective. Zhang Fan[5] proposed a real-time reliability measurement theory of software. By judging whether there is a deviation between the actual behavior α and the expected behavior β, the real-time measurement of software reliability is realized, and the real-time reliability measurement algorithm is given. The dynamic adjustment model[6] proposed in the literature [6] can realize the access by dynamically reducing the security level when the low security level is required to access the high security level. According to the analysis, it is generally believed that the non-interference model can be used as a general mathematical model for trusted computing. Reference [5] evaluates the measurement quality, but does not propose a feasible measurement method. The method of literature [6] will lead to the continuous reduction of the security level of the object, thus restricting the access, and the entities with higher security level can not be properly protected.

This paper proposes a dynamic trust model based on time-sharing control, which allows processes in untrusted domains to have controlled access to trusted domains, and restricts the operation of processes and expands trusted domains based on time dimensions to ensure that the access of processes does not affect the operation of trusted domains. Using the no-interference theory, this access strategy is proven to be secure.

2. Model definition

2.1 Elements of the model

Based on the interference-free model proposed by Rushby, some terminology is used and a temporal element is added to extend the model. The model is defined as follows:

Definition 1: A system M is defined as a tuple:$M = \{S, A, P, D, T, F, R\}$

S: System state set, expressed as $\{s_0, s_1, s_2, \ldots, s_n\}$, , n is a positive integer, where s_0 is the initial state of the system.

A: A set of system actions, expressed as $\{a, b, c, \ldots\}$, where $a, b, \ldots$ represent a single action, $\alpha, \beta, \ldots$ represent an action sequence, Λ represents an empty action sequence, and °represents an action connection between.

P: The set of processes that the system can run, expressed as $\{p, \mathrm{q}, \ldots\}$, $p, \mathrm{q}, \ldots$ represents a single process.

D: The set of domains contained in the system, expressed as $\{d_1, d_2, \ldots, d_n\}$, $d_1, d_2, \ldots$represent the domains in the system, where the set of trusted domains is called the set of trusted domains, Indicated by D_T, the processes in the trusted domain are all integrity-verified, complete, and loaded in order. The set of untrusted domains is called the set of untrusted domains, denoted by D_U, and arbitrary processes are allowed to load and run in the untrusted domain.

T: The time set of system operation, expressed as $\{t_0, t_1, \ldots, t_n\}$,$t_0$represents the initial moment, $\tau_0, \tau_1, \tau_2 \ldots$ represents the time series, and ° represents the connection between times.

F: The set of functions in the system, consisting of 9 functions:

(1) $step: S \times A \to S$, single-step state transition function, $step(s_1, a)$represents the state that state s_1 reaches after action a.

(2) $run: S \times A^* \to S$, multi-step state transition function, run $run(s_1, \beta)$means that state s_1 goes through the action sequence β. The state arrived after use.

(3) $pro: A \to P$, the action membership function, $pro(a)$ represents the process to which the action a belongs.

(4) $dom: P \to D$, the process membership domain function, $dom(p)$ represents the domain where the process p is located.

(5) $hung: P \times T \to P$, the process suspend function, $hung(p, t)$ represents the process state after the process p is suspended at time t , that is, process p cannot run.

(6) $recover: P \times T \to P$, process recovery function, $recover(p, t)$ means to recover process p at time t After the process state, that is, process p can resume running.

(7) $begintime: T^* \to T$, the start time function, $begintime(\tau)$ represents the start time of the time series τ.

(8) $endtime: T^* \to T$, the end time function, $endtime(\tau)$ represents the end time of the time series τ.

R: A collection of element relations in the system, including 3 categories:

(1) $\overset{p}{\simeq}$, the equivalence relation about the system state, $s_i \overset{p}{\simeq} s_j$ means that for process P, s_i and $s_j$are equivalent, $\overset{p}{\simeq}$ is reflexive, that is $s_i \overset{p}{\simeq} s_i$.

(2) $\sim>$, process interference relationship, $p_1 \sim> p_2$ means that the execution of process p_1 will have an impact on process p_2, $p \not\sim > p$ Indicates that the execution of processp_1 will not affect process p_2.

(3) $\prec$, the time partial order relationship, $t_1 \prec t_2$ means that time t_2 is after time t_1, and $t_1 \succ t_2$ means that time t_2 is before time t_1.

2.2 Dynamically tuned trusted access mechanism based on time-sharing control

Trusted models often restrict information from being passed from a high density to a low secret level, or a low integrity domain modifying a high integrity domain, which greatly limits the application of the computer. The interference-free theory has brought us the enlightenment that we can expand the access of the untrusted domain to the trusted domain by carefully designing, using the time-sharing control method and the dynamic adjustment of the trusted domain, as long as the access is non-interference, and make the operation of the trusted domain still trusted.

Definition 2: X is the set of objects to be accessed by the process, expressed as $\{x_0, x_1, x_2, ..., x_n\}$, where the set of objects located in the trusted domain is written as X_T, expressed as $\{x_{T1}, x_{T2}, ..., x_{Tn}\}$ is a set of objects in the untrusted domain, denoted by X_U as $\{x_{U1}, x_{U2}, ..., x_{Un}\}$, where X_T and X_U are both subsets of X.

Definition 3: Function $read: P \times T \rightarrow X$, the process read function. $read(p,t) = x$ means that at time t, the object read by process P is x.

Definition 4: Function $write: P \times T \rightarrow X$, the process write function. $write(p,t) = x$ means that at time t, the object that process P writes to is x.

Definition 5: Function $runtime: P \rightarrow T^*$, the process runtime function. $runtime(p) = \tau$ indicates that the running time series of process P is τ, that is, the time series from when process P is created to when P ends.

Definition 6: Function $expend: P \times D \rightarrow D$, an extension function of the domain. $d = expend(p, d)$ means that process p is added to domain d.

Theorem 1: the $p \in P, t \in T, dom(p) \in D_U, read(p,t) = x, x \in X_T$

If the following conditions are met:

$q \in P, q \sim> p \wedge dom(q) \in D_U$, for $t_1 \in T$,

When $t_1 \in runtime(p,x)$ then $hung(q,t)$,

When $t_1 \succ endtime(runtime(p,x))$ then $recover(q,t)$,

Then the process P that reads does not interfere with the process in D_U.

Proof: Consider time in segments.

① When $t_1 \prec begintime(runtime(p,x))$, process P does not start running, it has no interference to the process ofD_U.

② When $t_1 \in runtime(p,x)$, the process P reads the object x in the trusted domain, according to condition (1), the Processes that interfere with process P are suspended, that is, they cannot interact with process P, preventing the outflow of the information read by process P.

The read operation does not affect the integrity of the process in the trusted field D_T. The process in D_T is still trusted. also That is, process P does not interfere with processes in D_T.

③When $t_1 \succ endtime(runtime(p,x))$, the process P is finished, and its life cycle has ended Change the value in the trusted domain, which does not interfere with other processes.

Theorem 2: $p \in P, t \in T, dom(p) \in D_U, write(p,t) = x$,if $\forall q \in P, t_1 \in T, dom(q) \in D_T, \Rightarrow x \neq read(q,t_1) \wedge x \neq write(q,t_1)$ Then the process p performing the write operation this time does not interfere with the process in D_U ; if $\forall r \in P, \forall y,z \in X, t_1 \in T, dom(r) \in D_U, y = read(p,t), z = write(p,t), y \neq read(r,t_1) \wedge y \neq write(r,t_1) \wedge z \neq read(r,t_1) \wedge z \neq write(r,t_1) \Rightarrow expend(r, D_T)$, the process p joins in the domain D_T.

Proof: If an untrusted domain process p can write to an object x located in the trusted domain, then this x cannot be read or written by any process in the trusted domain. This means that x does not affect the running of the process in the trusted domain, that is, the process p does not interfere with the process in the trusted domain.

When the object read and written by the untrusted domain process r is no longer read or written by any process in the untrusted domain, the process r is considered to have no interference with the untrusted domain, and the process r is added to the trusted domain to expand it to a new trusted domain. The new untrusted domain is non-interfering with the new trusted domain.

Certificate completed.

Theorem 3: If the trusted domain D_T that does not allow access from the untrusted domain is trusted to operate, allowing the untrusted domain D_U to access the trusted domain is called an extended trusted domain $D_T^{'}$. D_T is non-interfering, then the extended trusted domain $D_T^{'}$ is still trusted to run.

Proof: Denote the process set in D_T as $\{p_1, p_2, \ldots, p_n\}$, and the access process set of the untrusted domain D_U to the trusted domain as $\{q_1, q_2, \ldots, q_n\}$.

Known, $\forall i, j\ 1 \le i \le n \wedge 1 \le j \le k \Rightarrow q_k \not\sim > p_n$

Summarize the number of q:

① When there is only one q, suppose the action sequence in q is $c_1 \circ c_2 \circ \ldots \circ c_m$, the action being executed in D_T before execution is a, the system state is s_1, and the system state during the process execution is $s_{11}, s_{12} \ldots s_{1m}$.

$$q \not\sim > pro(a) \Rightarrow s_{11} \overset{pro(a)}{\simeq} s_1 \wedge s_{12} \overset{pro(a)}{\simeq} s_1 \wedge \cdots \wedge s_{1m} \overset{pro(a)}{\simeq} s_1$$

It can be seen that the execution of q does not add a new system state, and the process in $D_T^{'}$ is still executed according to predictable steps, that is, $D_T^{'}$ is trusted to run.

② If there are n pieces of q, $D_T^{'}$ can be trusted to run, Prove that when there are n+1 pieces of q, $D_T^{''}$ can still be trusted to run.

Assuming that the action sequence in q_{n+1} is $c_1 \circ c_2 \circ \ldots \circ c_m$, the action being executed in the previous $D_T^{'}$ is a, the system state is s_1, and the system state during the execution of the process is $s_{11}, s_{12}, \cdots, s_{1m}$.

If $pro(a) \in D_T$, the situation is similar to ①, it can be proved that $D_T^{''}$ can be trusted to operate.

If $pro(a) \in \{q_1, q_2, \ldots, q_n\}$, assuming that the last action in the action sequence is executed in D_T as b, and the system state is s_2. After b is executed, the untrusted domain access process $q_i, q_j, \ldots, q_n, q_{n+1}$ is executed, and the system state during the execution of the process is $s_{i1}, s_{i2}, \cdots, s_{j1}, s_{j2}, \cdots, s_{n1}, s_{n2}, \cdots, s_{(n+1)1}, s_{(n+1)2}, \cdots$.

$$q \not\sim > pro(b) \Rightarrow s_{i1} \overset{pro(b)}{\simeq} s_2 \wedge s_{i2} \overset{pro(b)}{\simeq} s_2 \wedge \cdots \wedge s_{(n+1)1} \overset{pro(b)}{\simeq} s_2 \wedge s_{(n+1)2} \overset{pro(b)}{\simeq} s_2 \wedge \cdots \wedge s_{(n+1)m} \overset{pro(a)}{\simeq} s_2$$

It can be seen that the execution of q_{n+1} does not add a new system state, and the process in $D_T^{''}$ is still executed according to predictable steps, that is, $D_T^{''}$ is a trusted operation.

Induction can be obtained, no matter what the number of q is, the extended trusted domain $D_T^{'}$ is trusted to operation.

Theorem is proven.

3. Summary

Based on the interference-free theory, this paper proposes a dynamic trust model of time-division control, which expands the usability of the traditional model by controlling the process time-sharing and extending the trusted domain. The model is formally described, and the sufficient conditions for the time-division control mechanism and the trusted domain extension mechanism are proposed, and the formal proof is given. The results show that the model does not interfere with the trusted operation of the trusted domain.

References

[1]. HuJun, ShenChangxiang, et al.trusted computing 3.0 Engineering Fundamentals[M]. People Post Press,2017.

[2]. J. A. Goguen, J. Meseguer. Security Policies and Security Models [C]. In Proc. IEEE Symp. on Security and Privacy, 1982,p.11-20.

[3]. J Rushby. Noninterference, Transitivity, and Channel-Control Security Policies[R]. Stanford Research Institute, 1992,92(02).

[4]. Chen Liang, Zeng Rong-ren, et al. Trust Chain Transfer Model Based on Non-interference Theory[J]. Computer Science, 2016, 43(10): 141-144.

[5]. Zhang F, Xu MD, et al. Real-time Trust Measurement of Software: Behavior Trust Analysis Approach Based on Noninterference[J]. Journal of Software, 2019, 30(8): 2268-2286(in Chinese).

[6]. Trusted Computing Group. TCG specification architecture overview[EB/OL]. 2010.