http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (15): 3089-3095, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.3089.3095

Software Behavior Model Based on Multi-attribute Decision Making of System Call

[1]Hongyan Ma, [2]Fangfang Ou, [3]Nan Yang and [2]Zhen Li
[1]Industrial and Commercial College, Hebei University, 071002, Baoding, China
[2]College of Mathematics and Computer, Hebei University, 071002, Baoding, China
[3]CNPC Beijing Richfit Information Technology Co., Ltd., 100007, Beijing, China

Abstract: The system call attributes of traditional software behavior model based on system call are only related to deterministic attributes, such as system call name, system call context and system call argument policy which are not enough to model the software behavior accurately. For the problem, the fuzzy attributes of system call are introduced based on the traditional software behavior model and a software behavior model based on multi-attribute decision making of system call is presented. The model adopts information entropy to decide fuzzy attributes' weight objectively and provides anomaly detection of fuzzy attributes based on interval data by constructing the trusted model of fuzzy attributes. The experimental results verify the effectiveness of trusted model of fuzzy attributes based on interval data and the high attack detection capability against the actual software.

Key words: Software behavior, anomaly detection, system call, multi-attribute decision making, fuzzy attribute

INTRODUCTION

With the development of computer and network technology, software plays an important role in the information society. The expansion of software makes software flaws and vulnerabilities difficult to avoid. If the software behavior is always accordant with the expected behavior, we call the software is trustworthy (Wang *et al*., 2006). With continuous deepening of the software application in the sensitive fields such as finance, military affairs and economy, the trustworthiness requirement of software becomes more urgent.

The idea of trusted computing is to perfect the terminal computer fundamentally, while the present trusted computer only guarantees the static security of system resources. How to guarantee and measure the dynamic trustworthiness during the software running has become a key problem for software trustworthiness. The research on dynamic trustworthiness during software running has very important significance. When the software is untrustworthy, its behavior will deviate from the normal trace. Therefore, modeling normal behavior of software has become a current research focus in software trustworthiness.

The organization of this study is then described as follows. In Section Related Work, we give the related work on software trustworthiness and software behavior models. In Section Software Behavior Model, a software behavior model based on multi-attribute decision making of system call will be presented. In the next two sections, deterministic attribute decision making and fuzzy attribute decision making based on interval data will be discussed respectively. In Section Experiments and Analysis, the effectiveness of trusted model of fuzzy attributes based on interval data and the high attack detection capability against the actual software will be verified by experiments. In Section Conclusion, we will give concluding findings and future work ideas.

RELATED WORK

In recent years, researches on software trustworthiness have been paid more and more attention to by domestic and international researchers. Wang's research team (Wen *et al*., 2010; Ding *et al*., 2011) introduce aspect-oriented architectural design approach and relevant techniques into the design and analysis of software, offer an effective approach of software architectural design for trusted software based on monitoring and a component model that supports the online fine-grained adjustment to software adaptability. Xu *et al*. (2013) propose a dependable entity model for Internetware and present a method of trust measurement based on Bayesian networks. Wang *et al*. (2012) present a verification model for trustworthiness of interaction between software component by combining the Unified

Corresponding Author: Hongyan Ma, Industrial and Commercial College, Hebei University, 071002, Baoding, China

Modeling Language (UML) and Pi-calculus. Zhou *et al.* (2012) evaluate the component and system trustworthiness based on cloud model. Chen *et al.* (2009, 2012) propose a component security testing approach based on extended chemical abstract machine. Immonen and Palviainen (2007) propose a method and tooling for trustworthiness evaluation and testing of open source components. Mohammad and Alagar (2011) present a new architecture description language suited for describing the architecture of trustworthy component-based systems and a formal approach for the specification and verification of trustworthy component-based systems. Yan and Prehofer (2007) present an adaptive trust control model in order to support autonomic trust management for the component software platform based on a fuzzy cognitive map. Yang *et al.* (2012) model and evaluate the software security based on stochastic Petri nets in order to quantitatively predict software security in the design phase.

Because system call is the interface provided by the operating system to access system resources, system call can reflect the features of software behavior to a great extent. A lot of software behavior models based on system call have appeared (Tao *et al.*, 2010). Hofmeyr *et al.* (1998) modeled the behavior of privileged process by short sequences of system calls and constructed the normal behavior feature library for intrusion detection. Inspired by Forrest *et al.*'s work, many researchers proposed the software behavior models, such as Finite State Automaton (FSA) model and Pushdown Automaton (PDA) model built by static analysis of source code (Wagner and Dean, 2001), Vt-Path model built by dynamic learning (Feng *et al.* (2003), HPDA model combining static analysis with dynamic learning (Liu *et al.*, 2005), software behavior model based on system objects (Li *et al.*, 2009), anomaly detection model of program behavior combining system call with Markov chain (Frossi *et al.*, 2009), automaton of program behavior based on system call attributes (Li *et al.*, 2012).

The above software behavior models based on system call can be divided into two categories. One is to construct feature library of normal software behavior through statistical learning or rule mining. The other one is to model software behavior by constructing the software behavior automaton. The first one determines whether the software behavior is abnormal or not according to the feature library of normal behavior. It cannot describe the running trace of entire software. The sec one can describe the running trace of software but it is not enough to model the software behavior accurately because it is confined to deterministic attributes of system call, such as system call name, system call context and system call argument policy.

For the above problems, we propose a software behavior model based on multi-attribute decision making of system call. The model introduces the fuzzy attributes of system call based on the traditional software behavior model. It adopts information entropy to decide fuzzy attributes' weight objectively and provides anomaly detection of fuzzy attributes based on interval data by constructing the trusted model of fuzzy attributes of system call. The model makes software behavior be monitored more comprehensively and accurately.

SOFTWARE BEHAVIOR MODEL

The study introduces the fuzzy attributes of system call based on HPDA model (Liu *et al.*, 2005) and proposes a software behavior model based on multi-attribute decision making of system call. Figure 1 shows the model expressed as an automaton for a small part of a software. The model captures the deterministic attributes and fuzzy attributes of each system call issued by the software monitored and decides whether to deviate from normal software behavior according to the abnormal value of system call. The model is defined as a 5-tuple: $(S, \sum, T, s, A)$.

- S is a finite set of states
- $\sum$ is a finite set of input symbols. In our model, the set of input symbols is defined as

$$\sum = (X \times Addr) \cup Y \cup \varepsilon \quad (1)$$

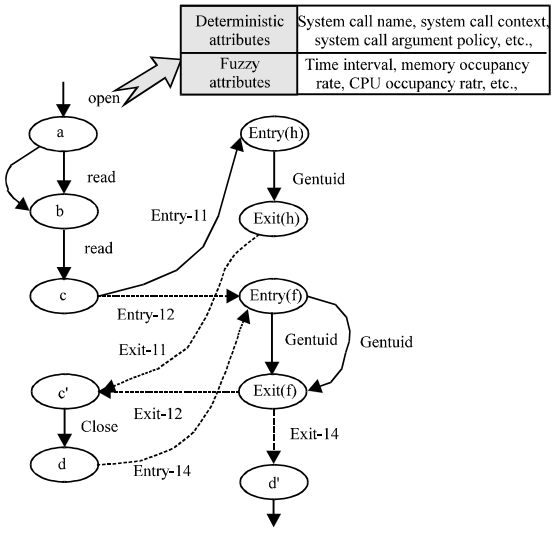

Fig. 1: Software behavior model based on multi-attribute decision making of system call

where , X = {entry, exit}, Y = {Syscall}, Σ is the empty string. Addr is the set of all possible addresses defined by the executable software and entry and exit are two new added system calls. entry is added before each function call and exit is added after each function call. entry and exit get the return address before and after the function call respectively. Y is the set of system calls. For each system call, there is a group of attributes which can be divided into two categories:

- **Deterministic attributes:** Once any deterministic attribute of system call deviates from the normal value, the system call is determined to be abnormal directly. These attributes includes system call name, system call context, system call argument policy, etc
- **Fuzzy attributes:** Fuzzy attributes cannot be expressed as accurate numbers. They are fuzzy and granted the prescribed error bounds. These attributes includes time interval, memory occupancy rate, CPU occupancy rate, etc., The time interval is the difference between the beginning time of current system call and the beginning time of last system call
- T is the set of transition functions: $(S\times\sum)\rightarrow S$
- s is the start state: $s\in S$
- A is the set of accept states: $A\subseteq S$

The abnormal value AV_i of system call S_i issued by the software depends on the abnormal value of deterministic attributes Ad_i and the abnormal value of fuzzy attributes AVf_i. The formula is as follows:

$$AV_i = \begin{cases} 1, & AVd_i = 1 \\ AVf_i, & else \end{cases} \quad (2)$$

If AV_i is greater than the threshold τ of abnormal value of system call, then S_i is abnormal.

DETERMINISTIC ATTRIBUTE DECISION

The deterministic attributes of system call considered include system call name, system call context and system call argument policy. Among them, System call context can be represented by calling stack of system call. If the calling stacks of two system calls are same, the system call contexts of them are same; otherwise, the system call contexts of them are different. For current call stack, if the return address of function main is α_0, the return addresses of internal functions are $\alpha_1,\ldots,\alpha_{m-1}$ and the return address of last system call is α_m, then the system call context can be expressed as $(\alpha_0,\alpha_1,\ldots,\alpha_{m-1},\alpha_m)$.

Let $SC = \{S_1,S_2,\ldots\}$ be the set of system calls and let $A^{si} = <A_1^{si},\ldots, A_n^{si}>$ be the vector of formal arguments for system call S_i. Each invocation S_{ij} of S_i has a concrete vector of values for A^{si} defined as $A^{sij} = <\alpha_1^{sij},\ldots, \alpha_n^{sij}>$. An argument set for system call S_i in the system call context C is the set of all argument vectors A^{sij}observed for the chosen system call issued in the context C. This is denoted by AS (C, S_i).

We limit our attention to unary and binary relations on system call arguments in our model.

Unary relations are attributes of a single argument. They can be represented using the form A^{Sij}. Such a relationship can be learnt from software behavior traces. The set AS (C, S_i) is represented using an enumeration by listing all elements of A^{Sij} in the given system call context C.

Binary relations are relationships between two system call arguments or between one system call argument and another system call return value. Let's take equality relation represented as equal for example. The file descriptor returned by open system call in context C_1 is equal to the operand of a subsequent write system call in context C_2.

$$fd_1(C_1, \text{open}) \quad \text{equal} \quad fd_2(C_2, \text{write}) \quad (3)$$

Considering the characteristics of deterministic attributes of system call, if any deterministic attribute deviates from the normal value, the abnormal value of the system call is 1; otherwise the abnormal value is 0.

FUZZY ATTRIBUTE DECISION MAKING BASED ON INTERVAL DATA

Trusted model of fuzzy attributes: The normal values of fuzzy attributes cannot be expressed as accurate numbers and are granted the prescribed error bounds. Therefore, the normal values of fuzzy attributes need to be changed from a large sample of raw data to symbol data by training. Each fuzzy attribute of system call can be represented as interval data.

Let the number of fuzzy attributes be n and α^1_{IJ}, $\alpha^2_{IJ},\ldots, \alpha^K_{IJ}$ be k sample values of the j th fuzzy attribute of system call S_i. First of all, fuzzy attributes need to be dealt with dimensionless method. The k sample values of the j th fuzzy attribute of system call $S_i$after pretreatment are denoted by $\alpha^{1'}_{IJ}$, $\alpha^{2'}_{IJ},\ldots, \alpha^{K'}_{IJ}$:

$$a_{ij}^{1'} = \frac{a_{ij}^1 - \overline{p_{ij}}}{e_{ij}} \; (1 \le j \le n, 1 \le l \le k) \quad (4)$$

Where:

$$\overline{p_{ij}} = \frac{\sum_{l=1}^{k} a_{ij}^1}{k}$$

$$e_{ij} = \sqrt{\frac{\sum_{l=1}^{k}(a_{ij}^{l} - \overline{p_{ij}})^2}{k-1}}\ \overline{p_{ij}}$$

$\overline{p_{ij}}$ and e_{ij} denote the mean and standard deviation of the jth fuzzy attribute P_{ij}, respectively.

We use n-dimensional data vector$X_i \in R^n$ to describe the normal scope of the fuzzy attributes after pretreatment of training samples:

$$x_i = ([p_{i11}, p_{i12}], [p_{i21}, p_{i22}], \cdots, [p_{in1}, p_{in2}])^T$$

Where:

$$p_{ij1} = \min_{1\le l\le k}\{a_{ij}^{l'}\},\ p_{ij2} = \max_{1\le l\le k}\{a_{ij}^{l'}\}$$

The normal scopes of n attributes of system call S_i form a normal scope hyper-rectangle NH_{in} denoted by $[u_{in}, v_{in}]$ where $u_{in} = (p_{i11}p_{i21},\ldots,p_{in})^T$, $V_{in} = (p_{i12}p_{i22},\ldots,p_{in})^T$ u_{in} and V_{in} are the left lower vertex and the right upper vertex of NH_{in}. Considering the sensitivity of boundary value of normal scope hyper-rectangle NH_{in}, we use the method of γ-truncated NH_{in} to form a credible interval hyper-rectangle NH_{in} which is smaller than Nh_{in}. $CH_{in} = [\tilde{u}_{in}, \tilde{v}_{in}]$, where $\tilde{u}_{in} = (\tilde{p}_{i11}, \tilde{p}_{i21},\ldots, \tilde{p}_{inl},)^T$, $\tilde{v}_{in} = (\tilde{p}_{i12}, \tilde{p}_{i22},\ldots, \tilde{p}_{in2})^T$. CH_{in} has the same midpoint $(u_{in}+v_{in})$ as NH_{in} and it can be adjusted by the value of γ, $\tilde{u}_{in} = u_{in}+\gamma$, $\tilde{v}_{in} = v_{in}-\gamma$. Boundary credible interval hyper-rectangle BH_{in} is denoted by $[u'_{in}, v'_{in}]$, where $u'_{in} = (p'_{i11}, p'_{i21},\ldots, p'_{inl})^T$, $v'_{in} = (p'_{i12}, p'_{i22},\ldots, p'_{in2})^T$, $u'_{in} = u_{in}-\gamma$, $v'_{in} = v_{in}-\gamma$.

The trust level can be divided into three levels: credible level, boundary credible level and incredible level. The credible region CR_{in}, boundary credible region BCR_{in} and incredible region ICR_{in} of fuzzy attributes can be obtained through the set operations of CH_{in} and BH_{in}.

$$CR_{in} = CH_{in},\ BCR_{in} = BH_{in} \cap CH_{in},\ ICR_{in} = BH_{in} \quad (5)$$

Computation of abnormal value of fuzzy attributes: For system call S_i, the vector of fuzzy attributes V_i is denoted by $[p_{i1}, p_{i2},\ldots, p_{in}]$. In the jth dimension, the distance between V_i and CR_{in} $d_j(V_i, CR_{in})$ is expressed as follows:

$$d_j(v_i, CR_{in}) = \begin{cases} 0 & p_{ij} \in CR_{in} \\ \min\{|p_{ij} - \tilde{p}_{ij1}|, |p_{ij} - \tilde{p}_{ij2}|\}, & p_{ij} \in BCR_{in} \end{cases} \quad (6)$$

Then we can calculate the abnormal value of fuzzy attributes AVf_i.

$$AVf_i = \begin{cases} 1, & (\exists j)(p_{ij} \in ICR_{in}) \\ \tanh\sum_{j=1}^{n}[w_{ij}d_j(v_i, CR_{in})], & \text{else} \end{cases} \quad (7)$$

where, W_{ij} is the weight of the *j*th dimension fuzzy attribute.

We give the computational method of whereW_{ij} below. For k samples of system call S_i, after dealt with dimensionless method, the lth sample X_l is denoted by $[\alpha^{l'}_{i1}, \alpha^{l'}_{i2},\ldots, \alpha^{l'}_{in}](1 \le l \le k)$. The sample data should be pretreated according to Eq. 8:

$$a_{ij}^{l''} = \begin{cases} 1.0 - \dfrac{\tilde{p}_{ij1} - a_{ij}^{l'}}{c_j}, & a_{ij}^{l'} < \tilde{p}_{ij1} \\ 1.0, & a_{ij}^{l'} \in [\tilde{p}_{ij1}, \tilde{p}_{ij2}] \\ 1.0 - \dfrac{a_{ij}^{l'} - \tilde{p}_{ij2}}{c_j}, & a_{ij}^{l'} > \tilde{p}_{ij2} \end{cases} \quad (1 \le j \le n) \quad (8)$$

where, $c_j = \max\{\tilde{p}_{ij1} - m_j, M_j - \tilde{p}_{ij2}\}$ M_j and m_j are the upper bound and lower bound of α^{ll}_{ij}, respectively.

We adopt information entropy to decide fuzzy attributes' weight objectively according to the dispersion degree of fuzzy attributes. Let the possible values of the jth fuzzy attribute of k samples be $V_1, V_2,\ldots, V_{hj}$, the entropy of the jth fuzzy attribute is as follows:

$$e_j = -K\sum_{r=1}^{h_j} q_{rj} \ln q_{rj} \quad (9)$$

Where:

$$K = \begin{cases} 1, & h_j = 1 \\ \dfrac{1}{\ln h_j}, & h_j \ge 2 \end{cases} \quad (10)$$

And q_{rj} is the probability of the value $V_r(1 \le r \le h_j)$ in k samples:

$$lq_{rj} = \frac{|V_r|}{\sum_{r=1}^{h_j}|V_r|} \quad (11)$$

$|V_r|$ is the frequency of V_r for the jth fuzzy attribute. The weight of the jth fuzzy attribute can be calculated as follows:

$$w_j = \frac{1 - e_j}{\sum_{\alpha=1}^{n}(1 - e_\alpha)} \quad (12)$$

EXPERIMENTS AND ANALYSIS

Anomaly detection of fuzzy attributes based on interval data: Our experiment uses machine learning databases (MLDBs) from UCI Repository (http://archive.ics.uci.edu/ml/index.html) including lots of different databases. This study selects Wine database which decides the place of origin through chemical analysis. It has 13 attributes which are all continuous real variables and has three classes. We select Class 1 as the normal dataset (60 samples in total) and Class 2 and Class 3 as abnormal dataset.

Because Wine database is high dimension data and the fuzzy attributes of system call usually have a lower dimension, we take any 6 attributes from 13 attributes in our experiment. We train the first 55 samples of class 1. Let $M_j = 5$ and $m_j = -5$ during pretreatment. The attributes' entropy can be computed according to Eq. 9. Figure 2 shows the entropies of six attributes when τ is 0.2, 0.5 and 0.8.

When τ is 0.2, the credible region CR_{in} is relatively large and the boundary credible region BCR_{in} is relatively small. The difference among the entropies of attributes calculated is very little. Therefore, the dispersion degree of attributes cannot be shown in a better way. When τ is 0.8, the attributes which values are centralized originally become dispersed because the credible region is too small which doesn't correspond with our requirement. Therefore, we select the compromise value $\gamma = 0.5$. Table 1 shows the weights of attributes computed when$\gamma = 0.5$.

The last five samples in Class 1 and all samples in class 2 and 3 are tested. Let $\gamma = 0.5$, $\tau = 0.25$. All the abnormal values of the test samples in Class 1 are less than τ and these test samples are determined as

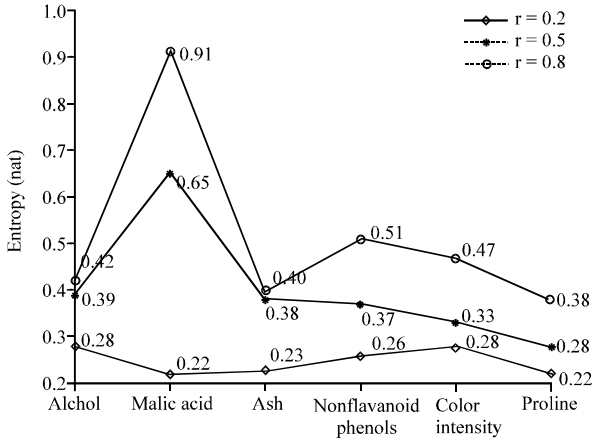

Fig. 2: Entropies of attributes for wine dataset

normal. The abnormal values of 94.3% test samples in class 2 and 82.7% test samples in class 3 are greater than τ and these test samples are determined as abnormal. The rest test samples in class 2 and 3 cannot be determined as abnormal only by these six attributes. Table 2 shows the detection accuracies of two methods of weight calculation for test samples. The weight of each attribute for average method is about 0.17. The results show that the detection accuracy of the information entropy method is higher than that of the average method.

Attack detection capability: We have made experiments on a PC with Intel (R) Core (TM)2 Duo E7500 2.93 GHz and 2 GB of main memory running Linux kernel 2.4.20. Each system call is intercepted by Loadable Kernel Module (LKM) and modified to capture the attributes' value of the system call. We form the software behavior model of vi 6.1 in Red Hat 9 Linux after training and make some attacking experiments.

Red Hat 9 Linux distribution includes vi 6.1 which exists potential TOCTTOU (Time of Check to Time of Use) vulnerabilities (Wei and Pu, 2005). Specifically, if vi is run by root to edit a file owned by a normal user, then the normal user may become the owner of sensitive files such as /etc/passwd. The sequence of system calls that the vulnerability window <open, chown32> is related to is as follows:

> …, open, write, close, chmod, stat64, chown32, chmod,...

Let, $M_j = 5$, $m_j = -5$, $\gamma = 0.5$

Attack 1: The attack consists of a tight loop constantly checking whether the owner of the wfname file has become root. Once this happens, the attacker replaces the file with a symbolic link to /etc/passwd. When vi exits, it should change the ownership of /etc/passwd to the attacker.

If the attack succeeds, when vi runs to chown32 system call, the abnormal can be detected by our model because the attribute is not satisfied with the following argument policy:

Table 1: Weights of Attributes when = 0.5

Attribute	Alcohol	Malic acid	Ash	Nonflavanoid henols	Color intensity	proline
Weight	0.17	0.10	0.17	0.17	0.19	0.20

Table 2: Detection accuracies of two methods for test samples

Method	Class 1	Class 2	Class 3
Information entropy method	100%	94.3%	82.7%
Average method	100%	76.5%	69.2%

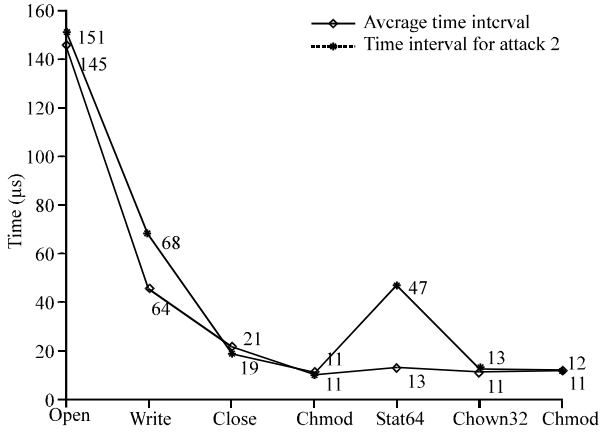

Fig. 3: Average time intervals and the time intervals when attack 2 happens

chown32(wfname) equal open(wfname)

The abnormal value of chown32 system call is 1, so the software behavior deviates from the normal.

Attack 2: The attack is against fuzzy attributes of system call. We take time interval attribute of system call for example and just focus on time interval. We inject a segment of code between the first *chmod* and *stat64* system call. The segment of code doesn't invoke system call, doesn't change the arguments of *stat64* and doesn't change memory occupancy rate and CPU occupancy rate basically.

Figure 3 shows the average time intervals of system calls gained during the training phase and the time intervals when attack 2 happens. From Fig. 3, we can see that write and stat64 system calls have a relatively large deviation from their average time intervals when attack 2 happens.

The time of open system call is related to the size of file to open and it affects the time interval of write system call, so for write system call the values of time interval in the training phase are dispersed and the weight of time interval is low. While, the values of stat64 system call's three fuzzy attributes are centralized and the difference of the weights of three attributes is very little. When attack 2 happens, for write system call, the abnormal value of fuzzy attributes calculated by (7) is 0.13 which is less than τ, so write system call is normal; for stat64 system call, the abnormal value of fuzzy attributes is 1 which is greater than τ, so stat64 system call is abnormal and the attack is detected.

CONCLUSION

The study introduces the fuzzy attributes of system call based on the traditional software behavior model and presents a software behavior model based on multi-attribute decision making of system call. The model constructs trusted model of fuzzy attributes, uses information entropy to decide fuzzy attributes' weight objectively and provides anomaly detection of fuzzy attributes based on interval data. The attributes of system call in our model are not limited to the attributes listed in the study and have good expandability. However, there is also some limitation. Our model may miss some attacks that cause the fuzzy attributes to deviate from normal a little. The value γ in the trusted model of fuzzy attributes and the threshold τ of abnormal value of system call should be adjusted for better effect of intrusion detection.

ACKNOWLEDGEMENTS

This work was supported by the Natural Science Foundation of China (Grant No.11101115), the National Natural Science Foundation of China (Grant No. 61170254), the Natural Science Foundation of Hebei Province (Grant No. F2011201039) and the Science and Technology Research and Development Guidance Plan Project of Baoding City (Grant No. 13ZG012).

REFERENCES

Chen, J.F., Y.S. Lu and H.H. Wang, 2012. Component security testing approach based on extended chemical abstract machine. Int. J. Software Eng., 22: 59-83.

Chen, J.F., Y.S. Lu and X.D. Xie, 2009. A fault injection model of component security testing. J. Comput. Res. Dev., 46: 1127-1135.

Ding, B., H.M. Wang, D.X. Shi and X. Li, 2011. Component model supporting trustwor thiness-oriented software evolution. J. Software, 22: 17-27.

Feng, H.H., O.M. Kolesnikov, P. Fogla, W. Lee and W. Gong, 2003. Anomaly detection using call stack information. Proceedings of the IEEE Symposium on Security and Privacy, May 11-14, Berkeley, CA., pp: 62-76.

Frossi, A., F. Maggi, G.L. Rizzo and S. Zanero, 2009. Selecting and improving system call models for anomaly detection. Proceedings of the 6th Detection of Intrusions and Malware and Vulnerability Assessment, July 9-10, 2009, Milan, Italy, pp: 206-223.

Hofmeyr, S.A., S. Forrest and A. Somayaji, 1998. Intrusion detection using sequences of system calls. J. Comput. Sec., 6: 151-180.

Immonen, A. and M. Palviainen, 2007. Trustworthiness evaluation and testing of open source components. Proceedings of the 7th International Conference on Quality Software, October 11-12, 2007, Portland, OR., USA., pp: 316-321.

Li, W., Y.X. Dai, Y.F. Lian and P.H. Feng, 2009. Context sensitive host-based IDS using hybrid automaton. J. Software, 20: 138-151.

Li, Z., J.F. Tian and X.H. Yang, 2012. Program behavior monitoring based on system call attributes. J. Comput. Res. Dev., 49: 1676-1684.

Liu, Z., S.M. Bridges and R.B. Vaughn, 2005. Combining static analysis and dynamic learning to build accurate intrusion detection models. Proceedings of the 3rd IEEE International Workshop on Information Assurance, March 23-24, College Park, MD., USA., pp: 164-177.

Mohammad, M. and V. Alagar, 2011. A formal approach for the specification and verification of trustworthy component-based systems. J. Syst. Software, 84: 77-104.

Tao, F., Z.Y. Yin and J.M. Fu, 2010. Software behavior model based on system call. Comput. Sci., 37: 151-157.

Wagner, D. and D. Dean, 2001. Intrusion detection via., static analysis. Proceedings of the IEEE Symposium on Security and Privacy, May 14-16, Oakland, CA., USA., pp: 156-168.

Wang, D., J.S. Chang and W.B. Zhao, 2012. Verification model for trustworthiness of interaction between software components with Pi-calculus. J. Frontiers Comput. Sci. Technol., 6: 419-429.

Wang, H., Y. Tang, G. Yi and L. Li, 2006. The trustworthiness mechanism of network software. Sci. China (Series E): Inform. Sci., 36: 1-14 (In Chinese).

Wei, J. and C. Pu, 2005. TOCTTOU vulnerabilities in UNIX-style file systems: An anatomical study. Proceedings of the 4th USENIX Conference on File and Storage Technologies, December 13-16, 2005, San Francisco, CA., USA., pp: 155-167.

Wen, J., H.M. Wang, S. Ying, Y.C. Ni and T. Wang, 2010. Toward a software architectural design approach for trusted software based on monitoring. Chinese J. Comput., 33: 2321-2334.

Xu, J., G.N. Si, J.F. Yang, S. Wen and B. Zhang, 2013. An internetware dependable entity model and trust measurement based on evaluation. Sci. China (Series E): Inform. Sci., 43: 108-125.

Yan, Z. and C. Prehofer, 2007. An adaptive trust control model for a trustworthy component software platform. Proceedings of the 4th Autonomic and Trusted Computing International Conference, July 11-13, 2007, Hong Kong, China, pp: 226-238.

Yang, N.H., H.Q. Yu, Z.L. Qian and H. Sun, 2012. Modeling and quantitatively predicting software security based on stochastic Petri nets. Math. Comput. Model., 55: 102-112.

Zhou, Y., C.Y. Gu and C.T. Cheng, 2012. Software dependability evaluation based on cloud model. Appl. Res. Comput., 29: 597-605.