ведение работ, мониторинг) / В. М. Улицкий, А. Г. Шашкин. – Москва : Издательство АСВ, 1999. – 327 с., ил.

2. **Геотехническое** обоснование вариантов приспособления объекта культурного наследия «Новая Голландия» для современного использования. Том 2. Общий расчет реконструируемых корпусов № 12 и № 12а (включая разработку численных моделей системы «реконструируемое здание – фундамент – основание – набережная»). 01-2011-98-ГР, ООО «ПИ Георекострукция». – Санкт-Петербург, 2012.

3. **Технический** отчет о результатах испытаний грунтов наклонными буроинъекционными сваями в составе свайного ростверка Рм-1 вертикальной статической вдавливающей нагрузкой на объекте строительства по адресу: Адмиралтейский район, наб. Адмиралтейского канала, д. 2, лит. И, «Новая Голландия», Корп. 12–12а. ОП ООО ФПГ «РОССТРО»-«ПКТИ». – Санкт-Петербург, 2012.

4. **СП 24.13330.2011** Свайные фундаменты. Актуализированная редакция. – Москва, 2011.

5. **Отчет** по результатам измерения напряжений для определения изгибающих моментов в трубах стволов наклонных свай опытного фрагмента свайного фундамента в процессе испытания конструкций вертикальной вдавливающей нагрузкой по адресу: СПб, наб. Адмиралтейского канала, дом № 2, лит. И, «Новая Голландия», Корпус 12–12а. ЗАО «ЭРКОН». – Санкт-Петербург, 2012.

6. **Программа** испытаний статической вертикальной нагрузкой грунтов по адресу: СПб, наб. Адмиралтейского канала, дом № 2, лит. И, «Новая Голландия», Корпус 12–12а. ОП ООО ФПГ «РОССТРО»-«ПКТИ». – Санкт-Петербург, 2012.

UDK 004.056

M. A. Polyanichko, A. A. Kornienko
Petersburg State Transport University

METHOD FOR AUTOMATED DETECTION OF CONFLICTS IN INFORMATION SECURITY COMPLEX*

This paper describes a method for automatic detection of conflicts in complex of information security software, based on the analysis of system performance, finding the conflicts in configuration files, registry keys, and dynamic libraries assessment. Models of conflicting objects of computer system and a method for calculation of a composite indicator value of performance decreasing, based on linguistic rules and discriminant function, are introduced.

information security software, conflict detection.

Introduction

Most modern computer applications have high demands on resources. This particularly applies to the information security software as it interacts intensively with many hardware and software components of computer system and use them during the complex system checks and other operations. Despite the fact that developers are constantly improving the process of interaction between information security software and computer system, as well as other software, the need for system resources continues to increase due to the increased complexity and the increasing number of malicious software [1], [2].

* При поддержке гранта РФФИ 13-07-13105-офи-м-РЖД.

1 Model of conflict interaction

Three models of conflict interaction between computer system and information security software are introduced in this paper (Table):

1. Dynamic Link LibraryDLL.
2. Registry keys of operation system.
3. Configuration files.

System performance decreasing is detected using analysis of ratio set, which is most sensitive for changes of measured resource state. In this paper processor, RAM and hard drive are analyzed [3]. The second step of this procedure forms logical rules, describing potential conditions for a conflict. First logical function (1) describes potential conflict of configuration files. Configuration files are used for storing software parameters. Program paths are often stored in these files [4]. For several programs, working with the same directories, a conflict may occur because of simultaneous interaction with resources or it can affect data integrity.

TABLE. Models of conflicting object

Set	Description
$SZI = \{func_1, \ldots, func_n\}$	$func_i$ – function called from security program
$DLL = \{func_1, \ldots, func_n, size, dest\}$	$func_i$ – storedfunctioncalled from security program
	size – library size
	dest – library destination
$func = \{name, param_1, \ldots, param_n, rettype\}$	$param_i$ – parameters, passed to function
	name – function name
	rettype – return type.
$Param = \{name, type\}$	*name* – parameter name
	type – parameter type
$Type = \{type_1, \ldots, type_n\}$	$type_i$ – type name
$SZI_{DLL} = \{dll_1, \ldots, dll_n\}$	SZI_{REG_i} – registry keys used by security program
	dll – library description
$dll = \{path, name, size, date\}$	*path* – library path
	name – library name
	size – library size
	date – library creation date
$OS_{REG} = \{SZI_{REG_1}, \ldots, SZI_{REG_n}\}$	SZI_{REG_i} – reg key stored in OS
	OS_{REG} – security programs that uses registry
$SZI_{REG} = \{regkey_1, \ldots, regkey_n\}$	SZI_{REG_i} – registry keys used by security program
	regkey – registry key
$regkey = \{path, name, type, data\}$	*path* – key path
	name – key name
	type – key type
	data – key data

For Table continuation see next page

Set	Description
$OS_{INI} = \{SZI_{INI_1}, \ldots, SZI_{INI_n}\}$	SZI_{INI_i} – configuration files, used by security program
	OS_{INI} – all security programs, that use configuration files
$SZI_{INI} = \{section_1, \ldots, section_n\}$	SZI_{INI_i} – configuration file, used by security program
	$section_i$ – section of configuration file
$section = \{name, var_1, \ldots, var_n\}$	var_i – variable in section
$var = \{name, key\}$	*name* – variable name
	key – variable describing key
$key = \{value, type\}$	*value* – variable value
	type – variable type
$type \in \{int, float, string, path, boolean\}$	*int* – integer variable
	float – float variable
	string – string variable
	path – destination variable
	boolean – logical variable

$$IniConflict = \begin{cases} \psi_1, \forall i,j,k \nexists l,n,m: (SZI_{INI_i} \subset OS_{INI} \wedge section_j \subset SZI_{INI_i} \wedge var_k \subset \\ \quad \subset section_j) \equiv \left(SZI_{INI_l} \subset OS_{INI} \wedge section_n \subset SZI_{INI_l} \wedge var_m \subset sect \right. \\ \psi_2, \forall i,j,k,z \exists l,n,m,q: \\ \quad (SZI_{INI_i} \subset OS_{INI} \wedge section_j \subset SZI_{INI_i} \wedge var_k \subset section_j \wedge key_z \subset v \\ \quad = SZI_{INI_l} \subset OS_{INI} \wedge section_n \subset SZI_{INI_l} \wedge var_m \subset section_n \wedge \\ \quad \wedge key_q \subset var_m) \wedge \left(key_z \subset var_k = path \vee key_q \subset var_m = path\right). \end{cases} \quad (1)$$

Logical function for dynamic libraries conflicts (2) describes situations, when the function exists in a program, but is missing in a library, or when parameters or return value of called function are differ from those in a library.

$$DLLConflict = \begin{cases} \psi_1, \forall i \exists j: func_i \subset SZI_{func} \equiv func_j \subset DLL; \\ \psi_2, \exists i,j: (func_i \subset SZI_{func} \wedge rettype \in func_i \neq func_j \subset DLL \wedge \\ \quad \wedge rettype \in func_j) \wedge (func_i \subset SZI_{func} \wedge name \in func_i = \\ \quad = func_j \subset DLL \wedge name \in func_j); \\ \psi_3, \exists i,j,k: (func_i \subset SZI_{func} \wedge param_k \in func_i \neq \\ \quad \neq func_j \subset DLL \wedge param_k \in func_j) \wedge (func_i \subset SZI_{func} \wedge \\ \quad \wedge name \in func_i = name \in func_j \wedge func_j \subset DLL); \\ \psi_4, \exists i,j: func_i \subset SZI_{func} \wedge func_j \not\subset DLL. \end{cases} \quad (2)$$

Conflict in interaction with the operating system registry keys occurs if several programs uses the same keys (3).

$$RegConflict = \begin{cases} \psi_1, \forall i,j \nexists l,k: (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i}) \equiv \\ \quad \equiv (SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k}); \\ \psi_2, \forall i,j \exists l,k: (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge path \in regkey_j \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge path \in regkey_l) \wedge \\ \quad \wedge (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge name \in regkey_j = \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge name \in regkey_l) \wedge \\ \quad \wedge (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge data \in regkey_j = \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge data \in regkey_l); \\ \psi_3, \forall i,j \exists l,k: (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge path \in regkey_j \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge path \in regkey_l) \wedge \\ \quad \wedge (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge name \in regkey_j = \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge name \in regkey_l) \wedge \\ \quad \wedge (SZI_{REG_i} \subset OS_{REG} \wedge regkey_j \subset SZI_{REG_i} \wedge data \in regkey_j \neq \\ \quad = SZI_{REG_k} \subset OS_{REG} \wedge regkey_l \subset SZI_{REG_k} \wedge data \in regkey_l). \end{cases} \quad (3)$$

2 Linguistic rules

For calculation of the composite indicator value of performance decreasing, fuzzy sets and linguistic variables are used, following the applying of linguistic rules [5]. Figure 1 shows a graph of the indicators of "Processor time", which which consists of four terms:

$$\beta, T, X, G, M, \quad (4)$$

where β = «proct»-linguistic variable name; T = {«Low», «Normal», «High», «Critical»} – basic terms; X = [0,100]; G = {∅} – procedure for generating new terms; M = {"Low" = = $L(x,25,37.5)$; "Normal" = $\pi(x,25,50)$; "High" = = $(x,50,75)$; "Critical" = $S(x,75,87.5,100)$}.

The rule base, sometimes called a linguistic model, is used to form outcome variables based on the input, consists of a set of fuzzy rules $R(k)$, $k = 1,\ldots, N$, where:

$$R^{(k)}: \mathbf{IF}\left(x_1 isA_1^k \mathbf{AND} x_2 isA_2^k \ldots \mathbf{AND} x_n isA_n^k\right) \mathbf{THEN}\left(y_1 isy_1^k \mathbf{AND} y_2 isy_2^k \ldots \mathbf{AND} y_m isB_m^k\right), \quad (5)$$

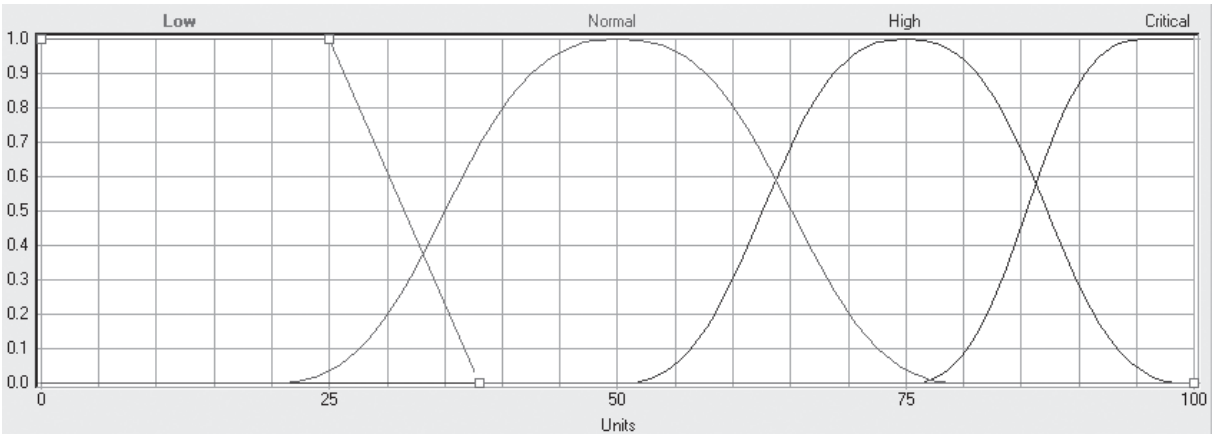

Fig. 1. «Processor time» indicator

where N – number of fuzzy rules; A_i^k – fuzzy sets $A_i^k \subseteq X_i \subset R, i = 1,\ldots,n$; B_i^k – fuzzy sets $B_i^k \subseteq Y_i \subset R, j = 1,\ldots,m$; x_1, x_2, …, x_n – input variables; $(x_1, x_2, \ldots, x_n)^T = x \in X_1 \times X_2 \times \times \ldots \times X_n$; y_1, y_2, …, y_n – output variables; $(y_1, y_2, \ldots, y_m)^T = y \in Y_1 \times Y_2 \times \ldots \times Y_m$.

Figure 2 shows an example of fuzzy inference scheme, the result of which is the vector Λ, representing system state. Further analysis of this vector displays the presence of conflict interaction in computer system security complex.

During the last step of the procedure the classification of the state vector is performed. It is a good practice to use the discriminant analysis for determination of the type of conflict interaction. Discriminant analysis step is divided into two stages. At the first stage it is necessary to identify and formally describe the differences between the observed objects, on the second stage the new objects are classified and assigned to one of the several groups. Signs, used to distinguish one subset from another, are called discriminant variables.

For differentiation of conflict states discriminant functions are used:

$$f_i(X) = a_{0i} + a_{1i}x_1 + a_{2i}x_2 + \ldots + a_{mi}x_m. \quad (6)$$

Discriminant function coefficients (a_i) are determined so that $f_i(X)$ are separate as much as possible.

Vector of the function coefficients a_{ij} ($j = \overline{0,m}$) is calculated using the following formula:

$$a_{ij} = \overline{X}_i \cdot S_*^{-1}. \quad (7)$$

Constant term:

$$a_{0i} = -\frac{1}{2}\overline{X}_i \cdot S_*^{-1}\overline{X}_i. \quad (8)$$

The discriminant function for each object are calculated by substituting the obtained values of the coefficients to the formula (7). The boundary that separates the set is called the discrimination constant and its value equidistant from the averages of functions [6]. For example:

$$c = \frac{1}{2}\left(\overline{f}_2 - \overline{f}_2\right). \quad (9)$$

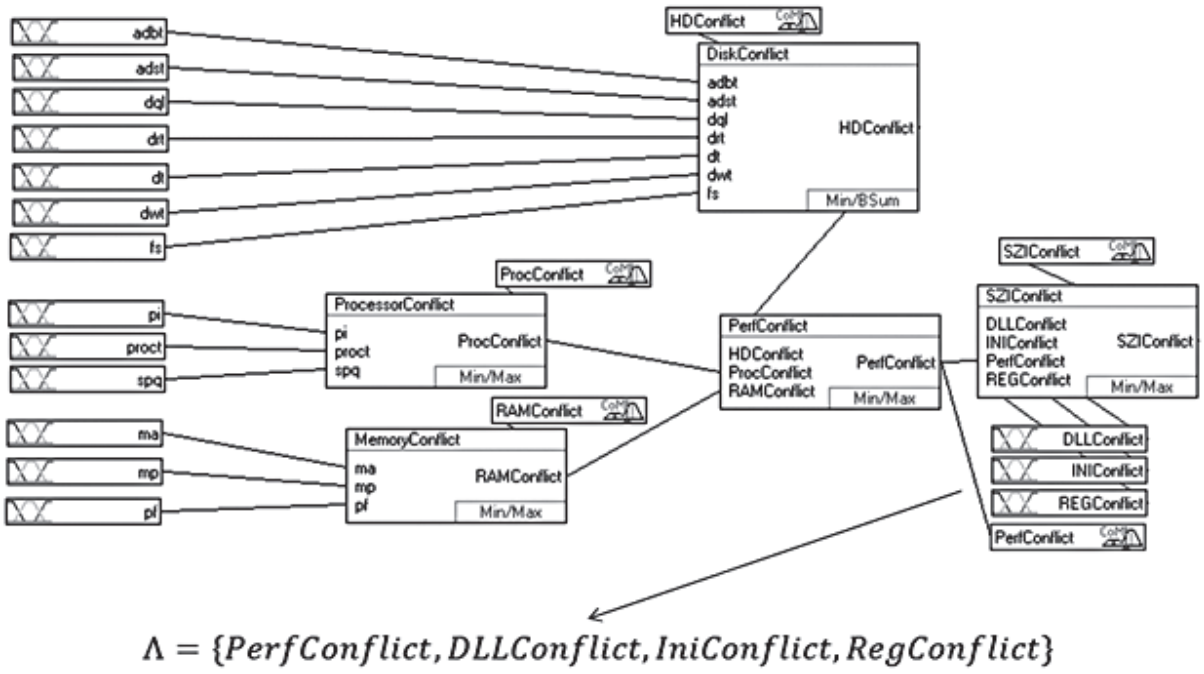

Fig. 2. Fuzzy inference scheme

Conflict state with variable $Y_1 = PerfConflict$, $Y_2 = DLLConflict$, $Y_3 = IniConflict$, $Y_4 = RegConflict$ will be attributed to the variety of M_i, for which the value $f_i = c_i + a_{1i}Y_1 + a_{2i}Y_2 + a_{3i}Y_3 + a_{4i}Y_4$ will be maximum.

Conclusion

Application of conflict detection method will improve the efficiency of system administration and will reduce the time, required for detection of conflict interaction existence. A rough estimate of the time, required to find the conflict, shows that the application of the proposed method can save up to 30 % of the time, by providing a clear indication of the conflict.

Referenses

1. **Zaitsev**, O. V. *Adaptive configuration of conflicting applications*. US Patent 7925874 B1, November 30, 2010.

2. **Ding**, Y., Guo, X., Su, H., Wang, Z., Zhao, S. *Method and system for avoidance of software conflict.* US Patent 20070180441 A1, December 22, 2006.

3. **Mcmillan**, J. J., Chirhart, G. D. *Method and system of managing software conflicts in computer system that receive, processing change information to determine which files and shared resources conflict with one another.* US Patent 7028019 B2, November 11, 1998.

4. ***Introduction*** *to Monitoring Performance Thresholds*, available at: http://msdn.microsoft.com/ru-ru/library/bd20x32d (v=vs.90).aspx

5. **Rutkovskaya**, D., Pilinskiy, M., Rutkovskiy, L. (2006). Neural networks, generic algorithms and fuzzy systems. Trans. from Pol. [*Neyronnyye seti, geneticheskiye algoritmy i nechetkiye sistemy*], Moscow, 452 p.

6. **Lectures** – Mathematical statistics. Applied mathematics chapters [*Lektsii – Matematicheskaya statistika. Spetsialnyye glavy prikladnoy matematiki*], available at: http://gendocs.ru/v25963/?cc=2