OOPN-SRAM: A Novel Method for Software Risk Assessment

Xiaofei Wu, Xiaohong Li*, Ruitao Feng, Guangquan Xu, Jing Hu and Zhiyong Feng
School of Computer Science and Technology, Tianjin University
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
{yanhanwxf, xiaohongli, rtfeng, losin, mavis.huhu, zhiyongfeng}@tju.edu.cn

***Abstract*—This paper proposes a Software Risk Assessment Method based on Object-Oriented Petri Net (OOPN-SRAM), in which risk assessment procedure is divided into four steps, expressed as four corresponding objects, including asset recognition, weakness analysis, consequence property confirmation and risk calculation. Each object is modeled with Petri net. Specialists recognize software assets by the1-9 scales method of Analytic Hierarchy Process (AHP). The weaknesses in a system are found by the vulnerability scanner. The damage degree and the exploitation likelihood of a weakness are evaluated by such authorities as Common Weakness Enumeration (CWE). The consequence properties are confirmed by specialists according to the software requirements. Finally, in the risk calculation, risk degree and overall risk value are calculated by using exponential method and weighted average method respectively. Furthermore, we illustrate the application of our OOPN-SRAM method with realistic examples including web-banking and forum, and make a comparison with traditional methods. The results show that OOPN-SRAM not only increases the efficiency of the evaluation process, but also makes the evaluation result more objective and accurate.**

Keywords—software; risk assessment; vulnerability scanner; OOPN; CWE

I. INTRODUCTION

Nowadays, the security of software has become serious and complicated. Low-quality software is one of the main reasons for the rapid growth of computer security issues. Identifying and dealing with risks early in development process lessens long-term costs and helps prevent software disasters [1].This paper mainly focuses on the security risks in the software testing phase.

Software risk assessment is an integral part of risk management of software projects [2].Two main components of the software risk assessment are the assessment procedure and the evaluation method. The assessment procedure is quite complex which includes not only many objects such as assets, weaknesses and assessors, but also many operations such as asset recognition and weakness analysis. Risk assessment method is the determination of quantitative or qualitative value of risk related to a concrete situation and a recognized threat. Currently, common risk assessment methods include matrix method, multiply method, fuzzy evaluation，grey theory and so on. However, the risk factor value in these methods does not integrate vulnerability scanners to give an overall risk statement. They often rely on the advisement from related experts and the statistics of historical record, which makes the evaluated result inaccurate and subjective. Therefore, the aim of this paper is to develop an assessment method to overcome the above issues.

This paper proposes a Software Risk Assessment Method based on Object-Oriented Petri Net (OOPN-SRAM). As we know, Petri net is widely used in many areas such as software design, workflow management system and reliability engineering. Petri net is also widely used for risk management in many fields, but there isn't a software risk assessment procedure integrated OOPN. So to the best of our knowledge, in this point, this paper is the pioneering work. Using OOPN to model the software risk procedure can improve the assessment efficiency and simplify the evaluation process. As for the risk evaluation method, this paper can alleviate the double counting of risk factors by using exponential method and weighted average method to calculate the risk degree and overall risk value. In addition, the evaluation result can be more objective and correct by utilizing the weakness information retrieved from application scanner and authorities such as Common Weakness Enumeration (CWE).

The remainder of this paper is organized as follows. Some software risk assessment models are introduced in Section2. Section 3 presents the definition of the OOPN-SRAM model. The OOPN-SRAM is modeled in Section 4. Section 5 presents a case study as an application of this method. In Section 6, we analyze the assessment results and make a comparison with other methods. Finally, Section 7 concludes our work.

II. RELATED WORK

A foundation work for qualitative and quantitative risk assessment of software projects is from Barry Boehm [1]. Say-Wei et al.[3] presented a risk assessment model (SRAM) for software projects by using a questionnaire to identify the risk events and their corresponding impacts. Heng Liu [4] defined a model of cloud computing security risk assessment. Multiplication method is used to calculate the risk factors in this model. This model can only calculate sub-risk values rather than give the total risk value of the system. Ju'an Wang et al.[5] proposed an approach to define security metrics based on vulnerabilities included in the software systems and their impacts on quality. The method used Common Vulnerabilities and Exposures (CVE) and Common Vulnerability Scoring System (CVSS) in the metric definition and calculation. The main drawback of the above models is double counting of the risk factors. Another problem is that the risk factor value often relies on the advisement of experts, which causes the result of risk evaluation to be subjective. Wensheng Zhang [6] used OOPN to model the procedure of information security risk assessment. In his work, the procedure is divided into several sub-processes, and then each sub-process is modeled by Petri net.

978-1-4799-5482-7/14 $31.00 © 2014 IEEE
DOI 10.1109/ICECCS.2014.28

In this paper, we consult this idea to model the software risk assessment procedure.

III. OOPN-SRAM MODEL

The basic concept of object-oriented Petri net (OOPN) is mapping a target system to cooperative objects, and then describing the behavior of each object and communication relations among them. So, there are different ways of definition about OOPN. This paper presents a definition of OOPN combined with software risk assessment procedure. Definitions of this extended Petri net are as follows:

IV. THE CONSTRUCTIONOF OOPN-SRAM MODEL

A. The Whole Procedure

According to the object-oriented thinking, software risk assessment procedure is divided into four sub-procedures: asset recognition ($SubSRAP_1$), weakness analysis ($SubSRAP_2$), consequence property confirmation ($SubSRAP_3$) and risk calculation ($SubSRAP_4$). Figure 1 is the Petri net model of the entire procedure, which describes the communication relations among the sub-procedures. In the following sections, the corresponding Petri net model of each sub-procedure will be described in more detail.

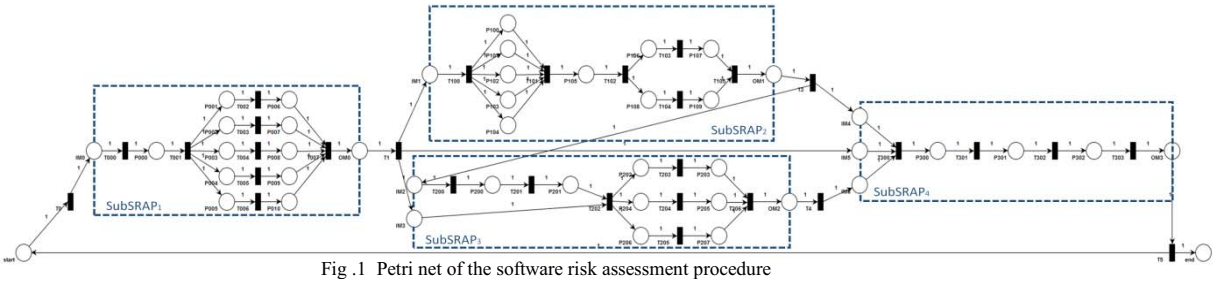

Fig .1 Petri net of the software risk assessment procedure

Definition 1: SRAP is the entire software risk assessment procedure.

$$SRAP = (\bigcup_{i=1}^{n} SubSRAP_i) \cup G_m$$

1) $SubSRAP_i \in SubSRAP$ is a sub-procedure of *SRAP*;

2) G_m is a finite set of gate transitions among the sub-procedures.

Definition 2: ISP_i is the inner structure of $SubSRAP_i$, and it is expressed as a four-tuple.

$$ISP_i = (\mathrm{P}, \mathrm{T}, \mathrm{IF}, \mathrm{OF})$$

1) $P = \{P_i, i = 1,2, \dots, n\}$ is a finite set of places;

2) $T = \{T_i, i = 1,2, \dots, n\}$ is a finite set of transitions ;

3) IF (P, T) is a directed arc between *P* and *T*. The map function is $P \times T \to N$ (N is a nonnegative integer);

4) OF (T, P) is a directed arc between *T* and *P*. The map function is $P \times T \to N$(N is a nonnegative integer).

Definition 3: OSP_i is the exterior structure of $SubSRAP_i$, and it is expressed as a five-tuple.

$$OSP_i = (\mathrm{IG}, \mathrm{OG}, \mathrm{IM}, \mathrm{OM}, \mathrm{EF})$$

1) $IG = \{IG_i, i = 1,2, \dots, n\}$ is a finite set of ingates;

2) $OG = \{OG_i, i = 1,2, \dots, n\}$ is a finite set of outgates;

3) $IM = \{IM_i, i = 1,2, \dots, n\}$ is a finite set of input information places;

4) $OM = \{OM_i, i = 1,2, \dots, n\}$ is a finite set of output information places;

5) $EF = \{EF_i, i = 1,2, \dots, n\}$ is a finite set of external flow relations of $SubSRAP_i$.

B. The Recongnition of Software Asset ($SubSRAP_1$)

OOPN-SRAM recognizes software asset from a perspective of dividing a software system into different function modules. The weight of each module is confirmed by five experts according to the 1-9 scales method of AHP. The variable meanings of $SubSRAP_1$ in Figure 1 are listed in Table I.

TABLE I. $SUBSRAP_1$ VIRABLE DESCRIBTION

OSP_1	IM_0: accept software function information; OM_0: send asset recognition information.	
ISP_1	T_{000} : divide the software into different function modules; T_{001}: specialists allocate weights for modules by 1-9 scales method of AHP; T_{002}-T_{006}: security specialist 1-5 allocate weights; T_{007}: summarize the result.	P_{000}: module list; P_{001}-P_{005}: security specialist 1-5; P_{006} - P_{010} : the evaluation results of security specialist 1-5.

The outputs of $SubSRAP_1$ are as follows:

Risk Data1: Function Modules $\{m_i | i = 1,2, \cdots, moduleNum\}$
Risk Data 2: Function Module Weights $\{mwt_i | i = 1,2, \cdots . moduleNum\}$

C. The Analysis of Software Weakness ($SubSRAP_2$)

We use Common Weakness Enumeration (CWE), which is a commonly used standard for the ontology integration of weakness management to identify the weakness in software. The weaknesses in each function module can be found by vulnerability scanners such as IBM AppScan and HP WebInspect. We use the damage degree and exploitation likelihood to evaluate each weakness. The damage degree is divided into four levels by referring to the severity description of IBM AppScan. The exploitation likelihood is divided into eight levels according to the CWE which has given a description of this attribute. The variable meanings of $SubSRAP_2$ in Figure 1 are listed in Table II.

TABLE II. SUBSRAP$_2$ VIRABLE DESCRIBTION

OSP$_2$	IM$_1$: receive asset information from OM$_0$; OM$_1$: send weakness analysis information.	
ISP$_2$	T$_{100}$: scan weaknesses; T$_{101}$: integrate weakness lists; T$_{102}$: analyze weaknesses; T$_{103}$: analyze damage degree; T$_{104}$: analyze exploit likelihood; T$_{105}$:integrate weakness analysis information.	P$_{100}$-P$_{104}$: each function module weakness list; P$_{105}$: weakness recognition results; P$_{106}$: damage degree; P$_{107}$: damage degreeanalysis list; P$_{108}$: exploit likelihood; P$_{109}$: exploit likelihood analysis list.

The outputs of $SubSRAP_2$ are as follows:

***Risk Data* 3**: Weaknesses $\{w_j | j = 1,2,\cdots,m\}$(The number of weaknesses is m)
Risk Data 4: Weakness Damage Degree $\{dd_j | j = 1,2,\cdots,m\}$
Risk Data 5: Weakness Exploitation Likelihood $\{el_j | j = 1,2,\cdots,m\}$
Risk Relationship 1: The many-to-many relationship between function modules and weaknesses

D. The Confirmation of Consequence Property(SubSRAP$_3$)

Consequence properties include losing integrity, losing confidentiality, losing availability and so on. Assessors need to confirm the consequence properties according to different requirements of different software. The weights of consequence properties can also be confirmed by the 1-9 scales method of AHP. Then, specialists need to confirm the relationships between consequence properties and weaknesses by referring to the attribute of *Common Consequence* in CWE. The variable meanings of $SubSRAP_3$ in Figure 1 are listed in Table III.

TABLE III. SUBSRAP$_3$ VIRABLE DESCRIBTION

OSP$_3$	IM$_2$:receive the weakness information from OM$_1$; IM$_3$: receive asset information from OM$_0$; OM$_2$: send consequence property information.	
ISP$_3$	T$_{200}$:confirm consequence property types; T$_{201}$: confirm consequence property weights; T$_{202}$: confirm the relationship between weaknesses and consequence properties; T$_{203}$-T$_{205}$: analyze the relationships between weaknesses and each consequence property; T$_{206}$: integrate consequence property results.	P$_{200}$: consequence property list; P$_{201}$: consequence property weight list; P$_{202}$: losing confidentiality; P$_{203}$: the evaluation result of losing confidentiality; P$_{204}$: losing integrity; P$_{205}$: the evaluation result of losing integrity; P$_{206}$: losing availability; P$_{207}$: the evaluation result of losing availability.

The outputs of $SubSRAP_3$ are as follows:

Risk Data 6: Consequence Properties $\{c_k | k = 1,2,\cdots,conNum\}$
Risk Data 7: Consequence Property Weights $\{cwt_k | k = 1,2,\cdots,conNum\}$
Risk Relationship 2: The many-to-many relationship between weaknesses and consequence properties

E. Risk Calculation (SubSRAP$_4$)

The goal of $SubSRAP_4$ is to calculate the risk value of the whole system according to 7 *Risk Data* and 2 *Risk Relationships,* which can be got from the above steps. The variable meanings of $SubSRAP_4$ in Figure 1 are listed in Table IV.

TABLE IV. SUBSRAP$_4$ VIRABLE DESCRIBTION

OSP$_4$	IM$_4$: receive the weakness information from OM$_1$; IM$_5$: receive asset information from OM$_0$; IM$_6$: receive the consequence property information from OM$_2$; OM$_3$: send risk level;	
ISP$_4$	T$_{300}$: calculate Rt_i; T$_{301}$: calculate $sumRt_k$ T$_{302}$: calculate *Risk*; T$_{303}$: evaluate risk level;	P$_{300}$: consequence property risk degree list for each module; P$_{301}$: consequence property risk degree list for the entire software; P$_{302}$: software risk value;

As we know, the increasing number of software weaknesses leads to the decreasing of its security. Furthermore, software is useless when the number of weaknesses in the system is very huge. So the variation trend of software risk value is similar to an exponential function. We use equation (1) to calculate the consequence property risk degree (Rt_i) for each function module. Let the number of weaknesses be n for one consequence property in one module.

$$Rt_i = \alpha * \exp\{-\sum_{j=1}^{n} dd_j * el_j / \beta\} \quad (1)$$

Where dd_j is *Damage Degree (Input Data 4)*, and el_j is *Exploit Likelihood (Input Data 5)*; α and β are the parameters which influence the variation trend of Rt_i. The values of α, β can be adjusted to change the precision of Rt_i depending on the situation. Next, the consequence property risk degree ($sumRt_k$) is calculated by the weighted average method for the whole system, as formula (2) shows.

$$sumRt_k = \sum_{i=1}^{moduleNum} mwt_i * Rt_i \quad (2)$$

Where *moduleNum* is the number of *Module (Input Data1)*, and mwt_i is *Module Weight* (*Input Data 2*).The risk value of the whole system is calculated by formula (3).

$$Risk = \sum_{k=1}^{conNum} cwt_k * sumRt_k \quad (3)$$

Where *conNum* is the number of *Consequence Property (Input Data 6)*, and cwt_i is *Module Weight* (*Input Data 7*). At last, we required to qualify the risk value to a defined level.

V. CASE STUDY

To verify the feasibility of OOPN-SRAM, we make a risk assessment for an actual software system–Altoro Mutual web-banking system (http://demo.testfire.net) and choose IBM Rational AppScan as the vulnerability scanner to scan the weaknesses in the system. There are 123 weaknesses, 31 weakness types were scanned out. According to the OOPN-SRAM, 5 security specialists divide the system into 12 functional modules and confirm consequence properties as losing confidentiality(C), losing integrity (I), losing availability (A), destroy access control (Ac) and other (O). Table VI shows part of information during the evaluation procedure. At last, the risk value of this system is calculated as 6.24 according to this method.

TABLE V. ALTORO MUTUAL

Module ID	Weakness	Risk Degree(Rt_i)				
		C	I	A	Ac	O
m1	+4,425,+3,531, 79,79,79,79,79, 74,550,301,626, 601,601	3.74	4.19	4.39	3.65	7.94
m2	+2,+3,425,523, 523,615,352	8.74	9.08	9.08	8.54	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮
m12	89,89,209,209,5 50,550,550,550, +2	6.21	7.35	10	7.35	10

VI. RESULT ANALYSIS

A. Rationality

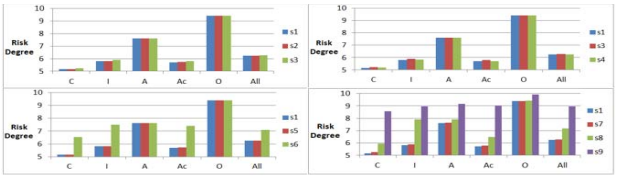

Fig. 2. Histogram of the comparative analysis for risk degree

In order to verify the rationality of OOPN-SRAM, we have improved Altoro Mutual in nine states as shown in Table VII. We make four comparisons with some states to see the variation of risk value as shown in Figure 2. For example, we can see the variation of risk degree when we fix a different weakness in the same module. The variation degrees are consistent with the prediction of specialists.

TABLE VI. IMPROVEMENT STATES OF ATOROL MUTUAL

State	Altoro Mutual Improvement State
s1	No improvement.
s2	Fix one CWE-613 in module m6.
s3	Fix one CWE-89 in module m6.
s4	Fix one CWE-89 in module m5.
s5	Fix all of the CWE-613 in the whole system.
s6	Fix all of the CWE- 89 in the whole system.
s7	Fix all of the weaknesses in module m5.
s8	Fix most of the weaknesses in module m6.
s9	Improve the whole system.

B. Validity

In order to confirm the validity of OOPN-SRAM, we compare the assessment result with the method of integrating work [3] with [4]. The risks of three software, including Altoro Mutual, Acoforum (http://testphp.acunetix.com/) and Crack Me Bank (http://crackme.cenzic.com), are evaluated by the two methods. Figure 3 shows that the risk values calculated by the two methods are close to each other, which proves the validity of OOPN-SRAM. But the result calculated by OOPN-SRAM is a little lower than the method of integrating work [3] with [4]. It means the risk evaluated by OOPN-SRAM is higher. Because the issue that one asset may have many same weaknesses is considered in OOPN-SRAM, but not in work [4]. In addition, it can better express the relationship between weaknesses and risk degrees by the exponential method of OOPN-SRAM than the multiplication method in work [3] and [4]. So the risk value calculated by OOPN-SRAM is more accurate.

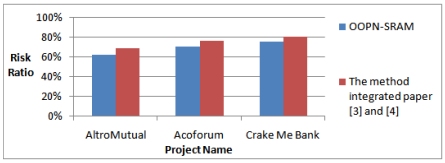

Fig. 3. Comparison result

VII. CONCLUSION

This paper proposes a software risk assessment method based on object-oriented Petri net (OOPN-SRAM). Innovations and advantages of OOPN-SRAM are as follows: 1) OOPN-SRAM is a general software risk assessment framework together with the vulnerability scanners. Evaluators can use different vulnerability scanners to make risk assessment based on this framework. 2) Using OOPN to model the software risk assessment procedure can make the procedure more clear, and assessors can make assessment more correct and efficient than before. 3) The evaluation result can be more objective and accurate by integrating the scan results of application scanner and the weakness information retrieved from CWE. So we can alleviate the specialist subjective judgment. 5) In the risk calculation, this paper uses a weighted average method to calculate the overall risk value by analyzing the *Risk Relationship 1* and *Risk Relationship 2,* which can furthest avoid the double counting of risk factors. 6) The comparison experiments show that the exponential method can better express the relationship between weaknesses and risk degrees than the multiplication method and the addition method.

ACKNOWLEDGMENT

This work has partially been sponsored by the National Science Foundation of China (No. 91118003, 61272106, 61340039), 985 funds of Tianjin University and Tianjin Key Laboratory of Cognitive Computer and Application.

REFERENCES

[1] Barry W. Boehm, "Software risk management: principles and practices."Software, IEEE 8.1 (1991): 32-41.

[2] M. Uzzafer, "A Novel Risk Assessment Model for Software Projects."Computer and Management (CAMAN), 2011 International Conference on. 2011.

[3] Say-Wei Foo, and Arumugam Muruganantham. "Software risk assessment model." Management of Innovation and Technology, 2000. ICMIT 2000. Proceedings of the 2000 IEEE International Conference on. Vol. 2. IEEE, 2000: 536-544.

[4] Heng Liu, Hongbing Wang, Yong Wang, "The overall cloud computing security risk assessment analysis". The 3th Conference on Vulnerability Analysis and Risk Assessment, VARA 2010:75-87.

[5] Ju'an Wang, Hao Wang, Minzhe Guo and Min Xia, "Security metrics for software systems." Proceedings of the 47th Annual Southeast Regional Conference. ACM, 2009: 47.

[6] Wensheng Zhang. "Research on classified protection and risk assessment procedure modeling based on Petri net" MS thesis ChongQing University. 2007