UDC 004.056.55 **doi: 10.26906/SUNZ.2021.2.070**

Zhang Liqiang[1], Cao Weiling[1], Viacheslav Davydov[2], Veronika Brechko[2]

[1] Neijiang Normal University, Neijiang, China
[2] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine

ANALYSIS AND COMPARATIVE RESEARCH OF THE MAIN APPROACHES TO THE MATHEMATICAL FORMALIZATION OF THE PENETRATION TESTING PROCESS

Abstract. In dynamic models, threats (vulnerabilities) can be viewed as a flow of temporary events. If the intervals of realized cyber threats are recorded, then a continuous log-list of events related to software security can be formed. In some cases and models, only the number of realized cyber threats for an arbitrary time interval can be recorded. In this case, the software response to threats can be represented only at discrete points. In static models, the implementation of cyber threats is not related to time, but the dependence of the number of errors or the number of implemented test cases (models by error area) on the characteristics of the input data (models by data area) is taken into account. The article analyzes the methods of mathematical formalization of the software penetration testing process. This software testing method is one of many approaches to testing the security of computer systems. The article substantiates the importance of the processes of preliminary prototyping and mathematical formalization. The classification is carried out and the advantages and disadvantages of the main approaches of mathematical modeling are highlighted. The list and main characteristics of dynamic and static models are presented. One of the negative factors of formalization is indicated - the neglect of the factors of a priori uncertainty in the safety parameters in static models.

Keywords: information security; vulnerable software; security testing; penetration.

Introduction

A mathematical model constructing process – a formalized description of a complex of factors that significantly affect the state and/or functioning of the object under research, and corresponding to this description of information support - is usually called mathematical modeling. The practical usefulness of mathematical modeling lies in the possibility of obtaining information about the qualitative properties and quantitative characteristics of the object under research without conducting (often complex or expensive) experiments in nature, which may justify the costs of overcoming difficulties arising in the development process or when trying to use mathematical models.

The main difficulty that one has to face in mathematical modeling is to ensure the adequacy of this model to the object under research. The user needs to find out how accurately this model reflects the real situation and how reliable quantitative estimates can be obtained in the process of working with this model. The experience of mathematical modeling of information processes, accumulated over the past few decades, shows that the problem of adequacy in a number of cases can be successfully solved. An example of this is the systems of computer simulation of numerous software and hardware components of computing facilities.

However, on the other hand, attempts to apply mathematical modeling methods to research such a complex and important process as the process of ensuring safety, convincingly demonstrate that, despite the natural desire to take into account in the model all the factors that significantly affect the functioning of the object under research, it is extremely difficult to achieve this.

In cases where the construction of a mathematical model that takes into account with an acceptable degree of accuracy all factors that are essential for the object under research is impossible, one has to abandon the standard methodology for using the model and try to act in other ways based on changing the formulations of the problems being solved and involving the user in the process of finding solutions.

In this situation, it is very important to make a reasoned choice of methods of mathematical formalization of the processes of ensuring the security of computer systems in general and software in particular.

The purpose of this article is to analyze and comparatively research the main approaches to the mathematical formalization of one of the software security testing methods, the software penetration testing process.

Literature analysis [1-10] has shown that currently there are many approaches to the mathematical formalization of the software penetration testing process. They are based on the well-known theories of computer engineering [8], queuing [4], neural networks [1], fuzzy logic [3], graph models and combinatorial calculation methods [9], etc. In addition, the means for solving optimization problems formalized on the basis of these models have been developed. These are, first of all, analytical methods, methods of mathematical programming, heuristic methods, etc. [10]. Let us analyze the approaches of mathematical modeling most often used in practice, which are adapted to varying degrees to modern requirements in the formalization of software security testing processes.

One of the software development leaders is Microsoft. In this situation, it is natural to start the review with an analysis of the modeling approaches offered by this company. In [6], the authors considered the Microsoft Spec Explorer tools. This system is based on the control of post and preconditions, described in special languages as an addition to the program code. The proposed approach is difficult to master, but powerful enough in capable hands, and is intended mainly for academic purposes use or by testing specialists at the level of the development and research department with sufficient knowledge of mathematical formalization methods. Another way of classifying software assessment of methodologies is presented in the works [5]. In these works, it is proposed to divide a number of approaches of mathematical formalization into the following testing groups: dynamic (allowing to assess the indicators of software technological security), static (allowing to obtain estimates of indicators of completeness and complexity of testing) models. At the same time, adapting the presented materials to the topic of software security testing, the following can be noted. In dynamic models, threats (vulnerabilities) can be viewed as a flow of temporary events. If the intervals of realized cyber threats are recorded, then a continuous log-list of events related to software security can be formed. In some cases and models, only the number of realized cyber threats for an arbitrary time interval can be recorded. In this case, the software response to threats can be represented only at discrete points. In static models, the implementation of cyber threats is not related to time, but the dependence of the number of errors or the number of implemented test cases (models by error area) on the characteristics of the input data (models by data area) is taken into account.

The list and main characteristics of dynamic and static models are presented in table 1 (1 – Markov exponential models: JM-model, Xui-model, Shanthikumar-model, Bucchianico-model; 2 – Semi-Markov models: SW-model, Hyperbolic model, Sukert-model; 3 – Heterogeneous Markov NHPP models (Duane model, Gompertz model, Goel-Okumoto model, Schneidewind model, Weibull model, Yamada exponential model, S-shaped Rayleigh model, Pareto model, Xie-logarithmic model, parabolic model, structural Nelson model, etc.; 4 – Software complexity models: metric Halstead error model, multivariate complexity model; 5 – Software testing completeness models: Mills model, Lipov model).

Table 1 – **List and main characteristics of dynamic and static models**

No.	**Short description**	**Mathematical formalization technologies background**	**Advantages, disadvantages**
1	**2**	**3**	**4**
		Dynamic	
1	Based on the assumptions that during software testing, the duration of time intervals between the detection of two errors has an exponential distribution with a failure rate proportional to the number of undetected errors.	The distribution density function of the *i-th* error detection time, counted from the moment of detection of the *(i – 1) - th* error, has the form: $p(t_i)=\lambda_i e^{-\lambda_i t_i}$, where – λ_i is the intensity of errors, which is proportional to the number of not yet detected errors in the program (varies depending on the type of model)	Advantages: ease of finding the main characteristics. Disadvantages: large increase in computations due to the inputting of states durations changing; the need to evaluate a large number of new parameters associated with each state.
2	Based on the Rayleigh model of software reliability growth. It is assumed that the error rate is proportional not only to the number of undetected software errors, but also to the debugging time interval:	Security error rate $\lambda_i=\phi\big(N-(i-1)\big)t_i$, where N is the number of errors originally present in the program; ϕ – the proportionality factor, interpreted as the intensity of the error detection, t_i – the interval between *(i-1) -th* and *i-th* security errors. Rayleigh distribution with the following density function: $$p(t_i)=\phi\big(N-(i-1)\big)\times t_i e^{-\phi\left(N-(i-1)\right)t_i^2/2}$$	Advantages: accuracy of simulation results. Disadvantages: the complexity of building a model of semi-Markov processes. Lack of a unified approach to the use of distribution laws when describing individual transient processes.
3	The model assumes that the number of errors that appear per unit of time are independent random variables distributed according to Poisson's law with a flow rate proportional to the expected number of errors remaining in the program at a given time. In contrast to JM – and SW – like convex models, here an additional assumption is made about the S – shaped dependence of the errors number on the testing time.	The error number function is given by the following formula: $m(t)=a\left(1-(1+gt)e^{-gt}\right)$ where a is a coefficient that characterizes the number of expected software errors, g – error detection intensity factor. Accordingly, the intensity of the error is determined as follows: $\lambda(t)=ag^2te^{-gt}$	Advantages: Accuracy of simulation results. A simplicity in finding a formula for the probabilities that a certain number of security errors will be detected and localized (or not) in a given time. Disadvantages: A large increase in the complexity of the model structure with small changes in the input conditions. The complexity of the mathematical representation of distribution laws in the description of individual transient processes.
		Static	
4	The complexity of the program is proposed to consider as a set of intellectual efforts (solving elementary problems by a person before an error occurs) when coding a text in a certain programming language.	Difficulty is estimated as $$E=\tilde{N}\log_2(\mu/L)=\eta_1 N_1 N\log_2\mu/(2\mu_2)$$ where $\tilde{N}=\mu_1\log_2\mu_1+\mu_2\log_2\mu_2$ – theoretical program length, $\mu=\mu_1+\mu_2$ – the number of unique operators and operands of the programming language, $L=\frac{2\mu_2}{\mu_1 N_2}$) – quality of programming, $N=N_1+N_2$ – the number of calls to operators and operands in the software.	Advantages: Software security assessment based on the results of software complexity assessment. Disadvantages: The presence of subjective assessment factors, the possibility of obtaining the effect of the "curse of dimension" with a large number of factors or a non-linear form of the approximating polynomial.

End of Table 1

1	2	3	4
5	Models for assessing the completeness of software testing are based on methods of independent introduction and detection of test errors and methods of conducting independent examinations. Errors are introduced randomly and recorded in the man-made error log.	The reliability of the statement that there are k errors in the program is as follows: $$R(k,S)=\begin{cases}1, & if\ n>k\\ S/(S+k+1), & if\ n\le k\end{cases}$$ where S – number of errors introduced, k – number of real errors found.	Advantages: Software security assessment based on the results of software completeness assessment. Disadvantages: A simplified look at the error detection process using a different number of tests. Neglecting factors of a priori uncertainty in software security parameters.

Among the intelligent approaches of the mathematical formalization of processes associated with the software life cycle, the direction based on neural networks is worth mentioning [1]. This is connected with specifics of the functioning of computer systems, which are human-machine systems.

Neural networks are an alternative to the statistical analysis components of anomaly detection systems. Neural networks make it possible to identify common test indicators, identify statistically significant deviations from the requirements for software quality and security. They are applicable as a statistical error detection system because of their self-learning ability. Moreover, a neural network can be configured so that it will further train on its own, constantly reacting to the slightest changes in the software. Neural networks require less intervention from penetration testers.

The studies carried out showed that at present, when modeling, software is presented in the form of a recurrent system that solves a number of specific tasks. In this case, the energy function acts as a system quality indicator:

$$E=-\frac{1}{2}\sum_{i}\sum_{j\neq i}w_{ij}x_i x_j\ ,$$

where w_{ij} is a weight coefficient between the *i*-th and *j*-th neurons; $x_i\ u\ x_j$ are *X* vector components (input data of the system),

and an expression of the form is used as a criterion for solving the problem of access distribution and data protection in a computer network:

$$E=\min(-\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1,k\neq i}^{N}\sum_{j=1}^{N_p(i)}\sum_{\ell=1}^{N_p(k)}\left|P_{ij}\cap P_{k\ell}\right|x_i^j x_k^\ell)\ ,$$

where P_{ij} is j-th resource between source and destination; $\left|P_{ij}\cap P_{k\ell}\right|$ – the number of nodes on a computer network which share routes P_{ij}, and $P_{k\ell}$; $x_i^j=1$, if chosen P_{ij}, and $x_i^j=0$ otherwise; $N_p(i)$ – number of access directions variants defined between the source and the destination.

Analysis of works in the field of data protection [1, 3, 8, 10] showed that this direction of modeling has a number of advantages associated with taking into account the specifics of external influences and the possibility of self-learning.

However, the studies of the software testing process models, presented in the form of neural networks [1], along with their advantages, have also shown disadvantages associated with the inevitable time spent on the learning process when building a model and, as a consequence, "conservatism" in relation to dynamic changes in the process of managing the system software development. Therefore, it is advisable to use these models when modeling individual components or structural elements of intelligent decision-making systems or to use them as the basis for the process of developing practical recommendations for managers.

Studies of models based on the apparatus of controlled random processes made it possible to determine two specific areas of software testing implemented by this approach: software technological security and debugging models which allow assessing the indicators of software technological security depending on runs on specified input data areas and subsequent software modifications. [2].

Among the advantages of this modeling approach, one should highlight the possibility of its implementation in the form of a link monitor and an audit system. In addition, in complex, multifactorial systems, the use of this modeling approach makes it possible to analyze individual components without the danger of reducing the accuracy of the modeling results as a whole.

At the same time, mathematical models of the software development process, taking into account the peculiarities of the modern SCRUM methodology and the factors of increasing security requirements, need a number of improvements.

In the automaton control model, the software penetration testing technology is represented as a deterministic automaton, the input of which receives a sequence of user commands. The main elements of the automaton model are: a set of system states $\{V\}$, many users $\{U\}$, multiple access matrices $\{M\}$, many user commands changing the access matrix $\{CC\}$, many user commands changing state $\{VC\}$, set of output values $\{Out\}$.

Among the advantages of this modeling approach, the variety of testing approaches should be mentioned. They determine not only the rules for the distribution of tasks, but also the configuration, the order of interaction between objects and subjects of the software security testing process.

Among the disadvantages of automatic models, we note the complexity of their practical implementation in the case of taking into account the whole variety of stages, methods and tools for software security testing. In addition, the problem how this direction of modeling should take into consideration the security factor is also not solved. Thus, as a result of the analysis and comparative studies of the existing approaches to the mathematical formalization of the software penetration testing process, a number of characteristic features, advantages and disadvantages of the existing areas of analysis and synthesis of these processes were identified.

The research of the main modeling approaches showed that in most models associated with the implementation of software security testing technology (especially dynamic models) there is no unified approach to the use of distribution laws when describing individual transient processes.

Moreover, the neglect of the factors of a priori uncertainty in the security parameters in static models is also a common negative factor for formalization. In addition, the lack of consideration in models of dynamic changes during software development (SCRUM features) requires appropriate research and development.

Conclusions

The main directions and approaches of mathematical modeling are analyzed, promising directions of mathematical formalization of software security testing processes are highlighted.

The expediency of improving the existing methods of software penetration testing by synthesizing a new software testing method taking into account increased security requirements is indicated.

References

1. Adetunji Adebiyi A Neural Network Based Security Tool for Analyzing Software // Adetunji Adebiyi, Johnnes Arreymbi, Chris Imafidon / Technological Innovation for the Internet of Things 4th IFIP WG 5.5/SOCOLNET Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS 2013, Costa de Caparica, Portugal, April 15-17, 2013. Proceedings
2. Daniel Dalalana Bertoglio Overview and open issues on penetration test // Daniel Dalalana Bertoglio, Avelino Francisco Zorzo / Journal of the Brazilian Computer Society (2017) 23:2 DOI 10.1186/s13173-017-0051-1
3. Kostadinov Dimitar Introduction: Intelligence Gathering & Its Relationship to the Penetration Testing Process [Electronic resource]. URL: https://resources.infosecinstitute.com/penetration-testing-intelligence-gathering
4. Mukhin, V., Kuchuk, N., Kosenko, N., Kuchuk, H. and Kosenko, V. Decomposition Method for Synthesizing the Computer System Architecture, Advances in Intelligent Systems and Computing, AISC, vol. 938, pp 289-300, DOI: https://doi.org/10.1007/978-3-030-16621-2_27
5. Markov A.S. Models for evaluating and planning software tests for safety requirements information // Bulletin of MSTU im. N.E. Bauman. Ser. "Instrument Engineering", 2011. Special issue "Technical means and systems of information protection ". S. 90-103.
6. Model-based Testing with SpecExplorer [Electronic resource]. URL:https://www.microsoft.com/en-us/research/project/model-based-testing-with-specexplorer/
7. Nickerson C. and other. The Penetration Testing Execution Standard / Chris Nickerson, Dave Kennedy,Chris John Riley, Eric Smith, Iftach Ian Amit, Andrew Rabie, Stefan Friedli, Justin Searle, BrandonKnight, Chris Gates, Joe McCray, Carlos Perez,John Strand, Steve Tornio, Nick Percoco, DaveShackelford, Val Smith, Robin Wood, Wim Remes,Rick Hayes. 30.04.2012 [Electronic resource]. URL: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines
8. Sanchez, M.A. Computer Science and Engineering—Theory and Applications / Sanchez, M.A., Aguilar, L., Castañón-Puga, M., Rodríguez Díaz, A. 2018. – 101 p.
9. Semenov, S., Sira, O., Kuchuk, N. Development of graphicanalytical models for the software security testing algorithm / Eastern-European Journal of Enterprise Technologies, Vol 2, No 4 (92), pp. 39-46, DOI: https://doi.org/10.15587/1729-4061.2018.127210
10. Study A Penetration Testing Model / Germany, Bonn. 111 p. [Electronic resource]. – URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile

Received (Надійшла) 21.01.2021
Accepted for publication (Прийнята до друку) 07.04.2021

Аналіз і порівняльне дослідження основних підходів математичної формалізації процесу тестування на проникнення

Zhang Liqiang, Cao Weiling, В. В. Давидов, В. О. Бречко

Анотація. У динамічних моделях загрози (уразливості) Software можна розглядати як потік тимчасових подій. Якщо фіксуються інтервали реалізованих кіберзагроз, то може сформуватися безперервний log-лист подій, відносящіхся до безпеки Software. У ряді випадків і моделей може фіксуватися тільки число реалізованих кібе-ругроз за довільний інтервал часу. У цьому випадку реакція Software на загрози може бути представлена тільки в дискретних точках. У статичних моделях реалізацію кіберзагроз не пов'язують з часом, при цьому враховують зави-ності кількості помилок або число реалізованих тест-кейсів (моделі по області помилок) від характеристики вхідних даних (моделі по області даних).У статті проаналізовано методи математичної формалізації процесу тестування на проникнення програмного забезпечення. Цей метод тестування програмного забезпечення є одним із багатьох підходів до перевірки безпеки комп’ютерних систем. У статті обгрунтовано важливість процесів попереднього прототипування та математичної формалізації. Проведено класифікацію та висвітлено переваги та недоліки основних підходів математичного моделювання. Представлено перелік та основні характеристики динамічних та статичних моделей. Вказується один із негативних факторів формалізації - нехтування факторами апріорної невизначеності параметрів безпеки в статичних моделях.

Ключові слова: інформаційна безпека; вразливе програмне забезпечення; тестування безпеки, проникнення.