Conference Paper

Investigating reliability models as a threat scanner tool in the context of information security

Chekunov A. A. and Norkina A. N.

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe shosse 31, Moscow, 115409, Russia

Abstract

The paper considers a number of features of practical use of the Mills models of software reliability. It investigates current problems related to the detection of threats to software. The paper compares the stages of software security and reliability. It presents the results obtained with the use of the model under consideration. It reflects the dependency of software reliability indicators on the number of testing stages.

Keywords: number of errors, software, testing, software error, reliability model.

Corresponding Author:
Chekunov A. A.
aachekunov@mephi.ru

Received: 11 December 2017
Accepted: 20 January 2018
Published: 13 February 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the FinTech and RegTech: Possibilities, Threats and Risks of Financial Technologies Conference Committee.

1. Modern problems of detecting software threats

During the first half of the year 2017, large companies in their reports on information security point to a rapid evolution of threats and an increase in their scale that can destroy backup copies of systems necessary to restore systems and data after attacks. With the development of the Internet of things (InternetofThings, IoT), more and more operations in various industries are being carried out online, which is expanding the horizon of attacks, increasing their scale and aggravating the consequences [1].

The recent attacks by WannaCry and Nyetya demonstrated the speed and breadth of malware distribution. This software looks like a program of extortionists, but in fact it can cause much more significant destruction.

The Internet of things continues to give attackers new chances. The recent activity of the IoT-botnet gives reason to believe that some attackers are already preparing the ground for a large-scale attack with catastrophic consequences, which can destroy the Internet itself.

Information security is becoming a key factor that determines the stable functioning of the business and the success of digital transformations. Such attacks as WannaCry and Nyetya show that intruders are becoming more inventive. And the complexity

and fragmentation of information security solutions (IS) allow criminals to find new vulnerabilities in defense systems. To counter modern threats, a holistic architectural approach is required that provides protection before, during and after the attack. Effective IB strategy begins with the elimination of the most obvious security gaps [1].

Software error: The program contains an error if its execution does not justify the user's expectations [2].

Testing is the process of executing programs to detect errors. A test is considered good, when it has a high probability of detecting an undetected error. A test is considered successful, when it detects an undetected error.

Application program is a program designed to solve a task or a class of tasks in a specific area of application of the information processing system.

A threat is a possible cause of an undesirable incident that could damage the system or the entire organization.

The threat to the security of information is a combination of conditions and factors that create a potential or actual danger of breach of information security [2].

Vulnerability is the weakness of an asset or management, the exploitation of which will lead to the implementation of one or more threats.

Vulnerability in software is a bug in software that can be directly used by a hacker to gain access to a system or network [3].

Internet of things is the concept of a computer network of physical objects equipped with built-in technologies for interaction with each other or with the external environment.

Botnet is a computer network consisting of a number of hosts, with running standalone software.

Private and state structures are engaged in providing information security. As a toolkit technical solutions can act, for example, it is possible to protect yourself from harmful effects by using a firewall. Firewalls do not allow easy penetration into the internal local area network from outside. Scanning ports allows to identify the launched services, provides an opportunity to exclude unnecessary. However, one needs to support Web servers, e-mail, FTP, SSH, Telnet, database applications. To assess the risks, it is necessary to know the threats and methods of unauthorized access to information and resources of the organization [1].

For these purposes, it is recommended to use a vulnerability scanner.

Vulnerability scanners allow testing various applications in the system for vulnerabilities that can be exploited. A Vulnerability Scanner can also use low-level tools,

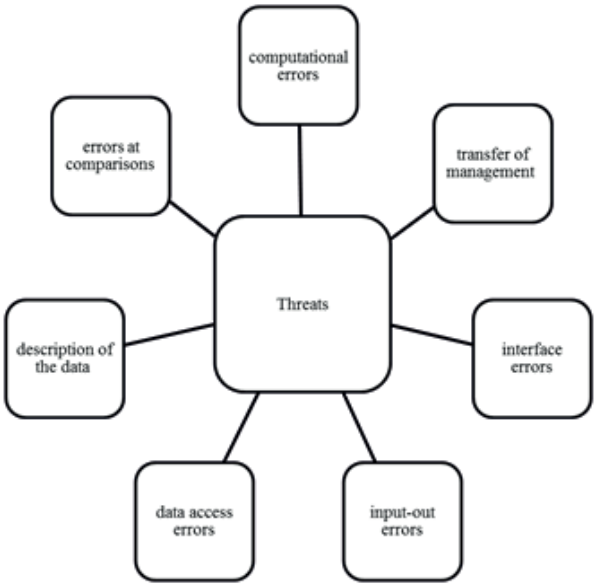

Figure 1: Types of software threats..

such as a port scanner and others, to identify and analyze the possible applications and protocols running on the system.

Modern vulnerability scanners offer more than 2,000 individual vulnerability tests that cover virtually all areas of potential weaknesses in systems. The architecture of many scanners makes it easy to add new tests.

The current range of software threats is quite broad. Figure 1 shows the most common types of software threats.

To identify each type of error, there are certain basic points. The presence of variables with undefined values is the most common program error, it occurs under different circumstances. For each access to a unit of data, it is necessary to informally "prove" that it has been assigned a value at the point under test [4].

In the case of working with arrays of data, it is necessary to check whether the value of each of the indices exceeds the limits defined for the corresponding measurement for all accesses to the array.

It is important to note that non-integer indices are not necessarily an error for all programming languages, but they represent a practical danger.

The presence of reference variables is an error of the type of "cyclic circulation". It occurs in situations where the lifetime of the pointer is greater than the lifetime of the memory to which the pointer is invoked. [5].

There are cases of an explicit or implicit addressing problem, this error occurs if the unit uses less memory allocation units than memory addressing units.

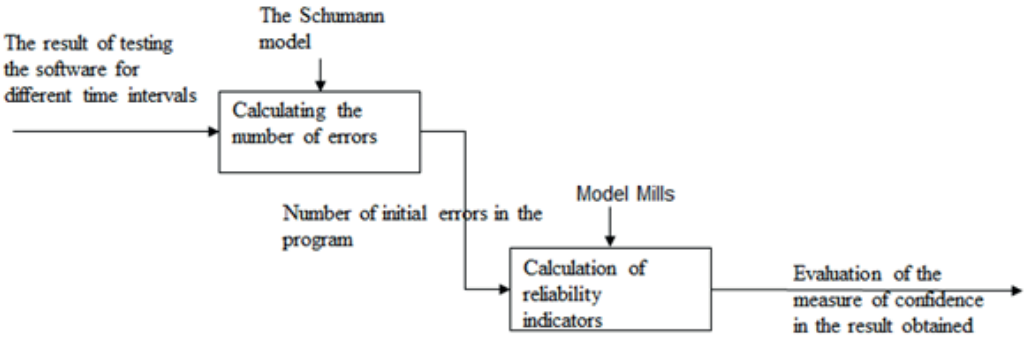

Figure 2: Algorithm for using the vulnerability scanner.

The main task of all types of testing for detecting threats is to increase the reliability of the software being tested, but if there are special indications of reliability in the source document reflecting the program's objectives, special tests can be developed to test it. However, if the program is required to provide a certain time to failure or given an acceptable number of errors, then there are a number of mathematical models to assess the validity of such requirements.

To predict the presence of errors in the software, there are mathematical models of software reliability. Based on the numerical results of testing in a certain period of time, a forecast is obtained for detecting the possible number of errors. These models can be applied at the stages of creating software and its support [2].

It is important to analyze the practical use of the Mills model of software reliability with various initial data.

One of the important parameters that affects the result is the assumption of the number of initial errors contained in the program. This value can be obtained with the help of some other model of reliability, which allows it to be obtained, for example, the Schumann model. The algorithm for using this method is shown in Figure 2.

The Schumann's model refers to discrete analytical models, in which the results, collected during the testing of the automated data processing system during specific or random time intervals, serve as the initial data.

To obtain the initial number of errors present in the program (N), the following formulas are used.

The average error rate (λ) in the section τ.

$$\lambda = \frac{\sum_{i=1}^{k} A_i}{\tau}, \quad (1)$$

where A_i is the number of errors at the i-th run

$\varepsilon_1(\tau)$ is the number of errors found during the test time in the calculation for one command in the code.

$$\varepsilon_1(\tau) = \frac{X}{I}, \quad (2)$$

where X is the number of errors found for a certain period of time.

The number of errors initially present in the program (N) is determined under the condition that there are two different testing points (τ) : $\varepsilon(\tau_a)$ and $\varepsilon(\tau_b)$, which are chosen arbitrarily taking into account the following requirement: the number of errors found at the moment *b* is greater than at the moment *a*, $\varepsilon(\tau_b) > \varepsilon(\tau_a)$, and the moment a precedes the moment *b:* $\tau_b > \tau_a$. [7]

In this case, it is possible to apply Formula 3:

$$N = \frac{I * \left[\frac{\lambda_b}{\lambda_a} * \varepsilon(\tau_a) - \varepsilon(\tau_b)\right]}{\frac{\lambda_b}{\lambda_a} - 1} \quad (3)$$

The approbation of the model shows that, given the presence of 1000 (I) operators in the test case (10 test runs were carried out during the research), the data in the graph is generated (see Figure 3).

It can be seen from the graph that if the position is close at the beginning of the testing period, a relatively long time must pass before the result (N stands for the expected number of errors in the program) reaches a fairly steady state.

The further forecast given by this model shows a certain increase in the forecast of the expected number of errors in the program, but requires additional research.

The results obtained in the first step using the Schumann model according to the proposed version of the reliability assessment of the developed (or modifiable) software were used in the Mills model (the Mills model with incompletely found errors introduced).

The results obtained for various parameters of this model are shown in the graph (Fig. 4).

These results are obtained on the basis of the assumption that the number of errors introduced is L = 100, and the number of expected errors is K = 100.

The results are obtained on the assumption that after every stage of testing all the errors found are eliminated. At the same time, new mistakes are not made.

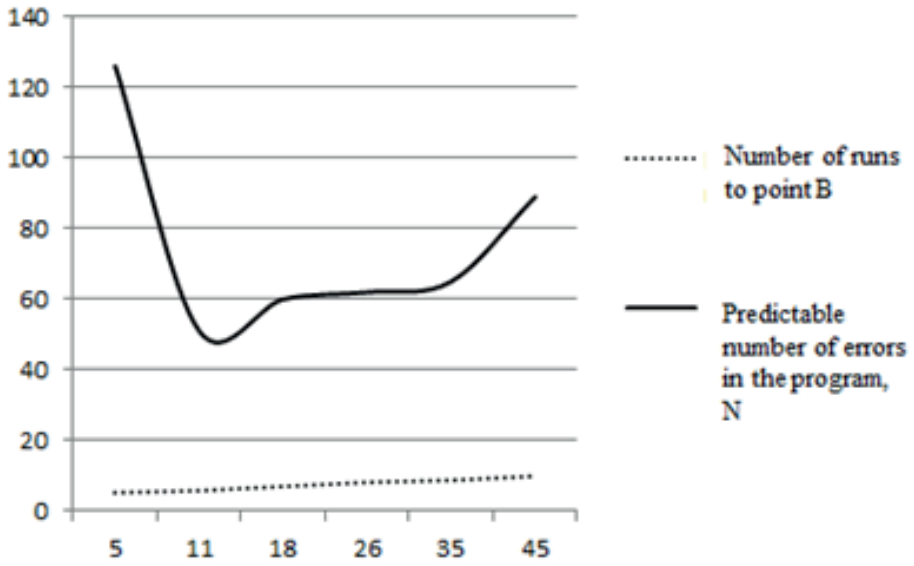

Figure 3: Graphical representation of the program results by the Schumann model.

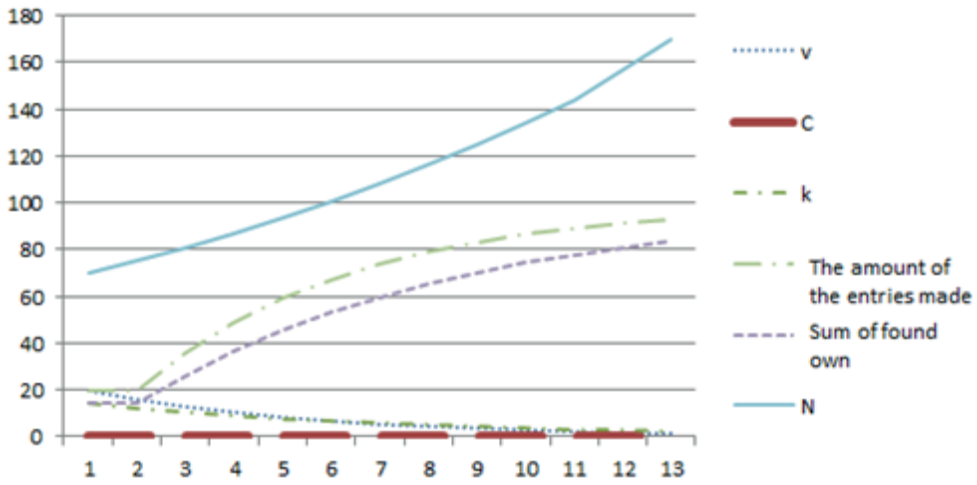

Figure 4: Dependence of reliability indicators on the number of testing stages.

At the next stage of testing, the number of own and introduced errors is reduced. Due to this, the number of detected own and introduced errors decreases.

As shown earlier (Fig. 4), when using dynamic reliability models, the measure of confidence in the obtained values of the total number of errors in the program is extremely small with the number of errors (k) found below the expected number of errors in the program.

When the analysis is carried out using an already debugged program, where the expected number of errors is small, the situation changes significantly.

The main factors are: the manifestation of destructive influences, the impact of the emerging destructive factors [6].

In this case, the following assumptions are usually made: the appearance of each destructive factor does not depend on the manifestation of others; each of the attackers is not dependent on other detected conditions.

To ensure the operation of reliability models, large volumes of raw data are needed, most of which are missing, and the formation is fraught with great difficulties.

The simulation showed that a satisfactory value of the reliability of the result to be obtained is provided merely for the case where the number of own mistakes found happens to be not less than the expected result.

Other acceptable reliability results may be in the case where a small number of errors is expected. These are the cases that are usually cited in the literature to illustrate the use of the Mills model [8, 9].

Acknowledgements

This work was supported by Competitiveness Growth Program of the Federal Autonomous Educational Institution of Higher Education National Research Nuclear University MEPhI (Moscow Engineering Physics Institute).

References

[1] https://www.cisco.com/c/dam/global/es_mx/solutions/security/pdf/cisco-2017-midyear-cybersecurity-report.pdf (circulation date 03.11.17).

[2] Myers G. J. Reliability of software. - Ed. Peace. Edition of the literature on mathematical sciences, 1980 - 354 p.]

[3] The art of testing programs Authors Glenford Meyers, Tom Badgett, Corey Sandler Format, Year, 2012 ISBN 978-5-8459-1796-6, 978-1-118-03196-4 Williams Publishing House

[4] Fred Fox. P. Jr. The Mythical Man-Month: Essays on Software Engineering, Reading, Mass., Addison-Wesley, 1975. Brooks F. P. How software complexes are designed and created. M., Science, 1979.

[5] Craig C. R.etal. Software Reliability Study. - RADC-TR-74-250, TWR Corp., Redundo Beach, Cal., 1974.

[6] Thauer R. H. and Hinton E. S. Software Reliability - A Method that Works. - Proceedings of the 1975 National Computer Conference. Montvaie, N. J., AFIPS Press, 1975.

[7] Lipaev V.V. Designing software: Teaching. manual for universities for special. "Automated System of Information and Control" - Moscow: Higher School of Economics. shk., 1990.

[8] Software reliability [electronic resource] -URL: http://madiasunik.ucoz.ru/Nadejnost/Material_nadejnost.pdf (circulation date 03.11.17).

[9] Pavlovskaya O.O. Static methods of software reliability estimation [electronic resource] URL: http: //dspace.susu.ru/xmlui/bitstream/handle/0001.74/778/7.pdf? Sequence = 1 (circulation date 03.11.17).