Use of modern technologies in assessing the stability and vulnerability analysis of information technology systems

A L Leontyev1,2,3,*, M I Chumak1,4, I V Chumak5
1 Scientific and Technological University “Sirius”, Sochi, Russia
2 Southern Federal University, Rostov-on-Don, Russia
3 Supercomputers and Neurocomputers Research Center, Co. Ltd., Taganrog, Russia
4 Don State Technical University, Rostov-on-Don, Russia
5 Department Institute of Technology of the Don State Technical University, Azov, Russia

*E-mail: Leontyev_Anton@mail.ru

Abstract. The paper is devoted to the study of the possibility of applying the main methods of assessing the reliability of information technology systems, including a review of existing developments in this field and analysis of the basic principles of testing systems. Special attention is paid to methods for evaluating the reliability of software. This paper describes the main steps that should be taken by an attacker in a cyber-attack. Methods of vulnerability detection using graph theory are considered and methods of their elimination are shown. The possibility of using neural networks for traffic analysis in order to detect signatures is considered. The mechanism of DDoS attacks is considered and the main mechanisms of protection against them are analyzed. Mechanisms for changing the topology of information technology systems through the use of graph theory were tested on decentralized systems and showed a significant increase in the system's resistance to unauthorized access. Mechanisms to prevent DDoS attacks through the use of neural networks allowed real-time tracking of unwanted content and reduced the time required to update databases of malicious signatures.

1. Introduction

The development and testing of information systems have been actively developing in recent decades. However, despite progress in this area, the information regularly appears about global data leaks and the failure of expensive equipment. Information technology systems constantly interact with information resources on the Internet, and may also be subject to adverse environmental influences, which under certain conditions can disrupt the stability of the system and as a result leads to loss of control over equipment or data. Here, sustainability is an integral property that includes the reliability of data storage, the survivability of the system, and its security. The development of software tools allows attackers to find vulnerabilities in information systems much faster and with a higher probability of success. The means of introducing malware into information systems are being improved, and decentralization and increased complexity in the connections between individual parts of the systems themselves lead to a decrease in the level of security.

In the works of Matveevsky V.R., the analytical dependencies of the failure rate on time for the distribution of Strett D. U. and Weibull V. are considered [1]. Polovko A.M. and Gurova S.V. investigate reliability indicators of non-redundant recoverable systems [2]. The manual describes the structural and
logical analysis of technical systems in order to assess the impact of each element on the performance of the system as a whole [3]. The paper by Romanyuk S.G. describes the possibility of applying a probabilistic approach to the reliability problem in the analysis of the technical system under study. The work of Graham B., Leroux P.N., and Landry T. is devoted to the static and dynamic analysis of software code for various defects and weaknesses. With such a wide variety of information on this topic, the consensus on methods for assessing the stability of information systems remains unformed.

2. Assessment of the reliability of information technology systems

The complexity of evaluating information technology systems (ITS) consists of two criteria. The first is the reliability of the technical part of the system, i.e. the hardware part. Since this area has been actively developing over a fairly long period of time, most of the system vulnerabilities are eliminated at the design stage. Another factor is the unreliability of the software, which is due to its increased complexity relative to the hardware, as well as its uniqueness and speed of obsolescence. Thus, the estimation of reliability is based on the theory of reliability of technical systems, based on which it is proposed to separate models associated with the probabilistic distribution of errors, the model allows to obtain acceptable results of the assessment, not considering the reliability of the instrument and assessment models FOR considering the complexity. Among the analytical methods for evaluating the reliability of information technology systems, there are continuous and discrete dynamic evaluation models, as well as static models.

2.1. Continuous dynamic models for reliability estimation

In such models, testing is performed without correcting errors. This type of model includes the Jelinski-Moranda model and the Markov transition probability model. The Jelinski-Moranda model assumes that the time between system failures is distributed exponentially, provided that the failure rate of the program is proportional to the number of errors remaining in the program [4-5].

Thus, under this assumption the probability of software uptime is defined as a function of time \( t \), detections \( i \) errors counted from the moment of detection \((i-1)\) error:

\[
P(t_i) = e^{-\lambda_i},
\]

where \( \lambda_i \) is failure rate,

\[
\lambda_i = C_p(N - (i-1)),
\]

here \( C_p \) is coefficient of proportionality; \( N \) is initial number of errors.

The main advantage of the model is the simplicity of calculations, but if the parameter is incorrectly defined \( N \) the failure rate of the program becomes negative, which leads to meaningless results. A more important factor is that this theory assumes that no new errors are introduced when the detected errors are corrected, which is almost never done during software development.

The Markov model [6-7] is used to estimate the likely number of errors that will be corrected in a given time, considering the preliminary modeling of the intensity of errors that occur \( l \) and the adopted error correction system. \( m \). Under this model it is possible to obtain information about the readiness \( A(t) \) and reliability \( R(t) \) software. So, the system is ready for time \( t > 0 \) defined by the equation:

\[
A(t) = \sum_{k=0}^{\infty} P_{n,k}(t),
\]

here \( P_{n,k} \) is the probability of a system’s transition from state \( n \) to state \( k \), independent of the previous and subsequent states of the system.

The reliability of the system is determined by the expression

\[
R_k(\tau) = e^{-\lambda_k \tau}, \quad 0 \leq \tau \leq T_{k+1}, \quad k = 0,1,2,\ldots
\]

Within this model, there are a number of limitations: all errors are considered random and do not consider the generated consequences, the intensity of occurrence and correction of errors is constant, and the time to change the state of the system is infinitely short. The practice of using this model shows
that the intensity of error correction is delayed in relation to their detection, which makes the process difficult.

On the other hand, the use of this model assumes the presence of pre-accumulated experience, which will allow you to systematize error data and improve the accuracy of analysis when using data from previous modeling.

2.2. Discrete dynamic models for evaluating ITS reliability

In such models, when a failure occurs in the system, all errors are found and eliminated. Classical models of this type are the Schumann and Musa models [8-10].

The Musa model assumes an assessment of software reliability at the operational stage. With the total testing time \( T \) and the number of failures during testing \( n \), this model allows you to determine the average time to failure:

\[
\tau = \tau_0 \left( \frac{CT}{n\tau_0} \right)
\]

here \( \tau_0 \) is average time to failure before testing; \( C \) is a coefficient that reflects the compaction of the test time compared to the actual operating time.

The parameter \( \tau_0 \) can be defined as

\[
\tau_0 = \frac{1}{NKf}
\]

here \( N \) is the initial number of errors in the software (can be obtained from other models); \( K \) is error rate; \( f \) is averaged values of the execution speed of a single program operator.

As the main drawback of this model, it should be noted that it is necessary to perform calculations using a different model (definition \( N \)), which increases the time cost. However, this model does not need to fix failure points, since errors are registered and can be corrected after software testing is completed.

The Schumann model involves step-by-step testing with different sets of test data. While testing is underway, errors are only registered. When the tests are completed, all registered errors are corrected, the test data sets are corrected, and the test is repeated. It is assumed that no new errors are made during the correction, and the intensity of error detection is proportional to the number of remaining errors.

At the \( i \) testing stage, the Schumann reliability function is described as:

\[
R_i(t) = e^{-\lambda_i t}
\]

here \( \lambda_i = (N-n_{i-1})C \); \( N \) is initial number of errors in the software; \( N-n_{i-1} \) is number of errors remaining by the beginning of the \( i \) stage; \( C \) is coefficient of proportionality:

\[
C = \frac{\sum_{r=1}^{l} \frac{m_r}{N-n_{r-1}}}{\sum_{r=1}^{l} \tau_r}
\]

where \( m_r \) is number of errors detected over time \( \tau_r \).

The advantage of this model is that it does not require additional calculations based on third-party models, while the obvious disadvantage is the assumption that there are no new errors when correcting the detected ones.

2.3. Static models of reliability assessment

Among the static models for evaluating the reliability of systems, which differ from others by not considering the time of error occurrence, there are classical models: mills and Nelson, Corcoran and Monte Carlo [11-13].

The mills model is based on the fact that a number of known errors are previously introduced into the software. At the same time, the testing specialist does not know the number and nature of errors made, which creates equal probability conditions for finding errors.
After testing, when \( n \) real and \( m \) artificial errors are detected, you can determine the initial number of errors in the program \( N \):
\[
N = nM/m,
\]
where \( M \) is number of pre-entered errors.

Then the calculated value is checked \( N \). It is assumed that the program initially had \( K \) errors and the program is tested before all \( M \) errors are detected. The probability that the program initially had \( K \) errors is calculated from the ratio:
\[
p = \begin{cases} 
0, & \text{if } n > K, \\
\frac{M}{(M + K + 1)}, & \text{if } n \leq K.
\end{cases}
\]

If not all previously \( M \) entered errors are detected, but \( m \) only some of them, then the ratio is used:
\[
p = \begin{cases} 
0, & \text{if } n > K, \\
\frac{C_i^{m-1}/C_i^{K+m}}{C_i^{K+m+1}}, & \text{if } n \leq K.
\end{cases}
\]
where \( C_i^n \) – число сочетаний из \( n \) элементов по \( m \) элементам.

This model uses a fairly simple mathematical apparatus, but errors are made in the software, which is a poorly formalized process, and an arbitrary assumption of the \( K \) value is a subjective factor.

The Corcoran model assumes that the software has multiple sources of software failures. The model argument is the number of programs runs, and the software reliability score is expressed as follows:
\[
R(n) = \frac{n^+}{n} + \sum_{i=1}^{n} \delta_i \frac{n^-_i - 1}{n},
\]
here \( n^+ \) is number of successful software runs; \( n^-_i \) is number of detected \( i \) type errors that are likely to be resolved \( p_i \); \( \delta_i \) is the coefficient is determined as:
\[
\delta_i = \begin{cases} 
p_i, & \text{if } n^-_i > 0, \\
0, & \text{if } n^-_i = 0.
\end{cases}
\]

This model considers the existence of several types of errors in the software, which is an extremely positive factor. The disadvantage of the model is the need to determine the probability of sampling data from the required area.

The Nelson model was developed with the basic properties of machine programs in mind. The model is based on software properties, and it allows development by detailed description of other aspects of reliability, and, therefore, can be used to calculate the reliability of software throughout the life cycle of an information technology system.

In this model, it is assumed that the program input data is divided into disjoint areas \( Z_i, i = 1, 2, ..., k \). It is assumed that a set of data from the area \( Z_i \) is likely to be selected \( p_i \). The value \( p_i \) is determined based on statistics of input data in real-world software conditions.

If by the time the reliability of the software was evaluated, \( n_i \) software runs were performed on data sets from the \( Z_i \) area, \( n^-_i \) of which ended in failure, then the software reliability is evaluated using the formula:
\[
R = 1 - \sum_{i=1}^{k} \frac{n^-_i}{n_i} p_i.
\]

The main advantage of this model is its focus on determining the reliability of software without interacting with the theory of hardware reliability, which allows us to apply this model at all stages of the software lifecycle. On the other hand, this model does not provide significant results at the initial stage of software development, as it requires a large number of runs to build an objective assessment.
3. Analysis of the vulnerability of ITS

Vulnerability analysis refers to the process of searching for threats and potential ways to implement them, as well as the models of attackers involved in these processes.

The presence of vulnerabilities has a negative impact on the scope of ITS application and causes rejection by users, among other things, the risks of disclosure of confidential information increase. Threat sources can be intentional or accidental, as well as caused by natural or man-made factors. For each threat, there is a list of vulnerabilities that allow it to be implemented, as well as a variety of methods to identify and combat them.

3.1. The application of graph theory

To detect vulnerabilities in complex decentralized systems, it is best to use graph theory. If you represent system elements as separate vertexes, you can get the topology of ITS or the interaction of several systems (figure 1).

Figure 1. Topology of interaction of elements of several ITS

In addition to direct attacks, attackers can resort to indirect attacks, in which case the attack is not the system itself, but a less secure system that interacts with the main target of the attacker. In this case, the goal of the attack is to find the minimum path of traversing the graph before getting access to the files of interest. To implement security tools in this case, you need to solve the problem of finding a click in an undirected graph. In this case, the application of methods of protection against such attacks consists in a preliminary analysis of vulnerabilities, that is, it is necessary to conduct a simulation of the attack. This method is good because it allows you to find possible threats from the first stages of system design.

To minimize the possibilities of exploiting vulnerabilities of topology you can create multiple false peaks that you will pass on useless information. This method requires additional hardware costs, but can be extremely effective, especially if the transmitted data is subject to encryption.

Another, more effective method of countering this threat is to deliberately minimize the path to the intended target of the attacker, while complicating the security system. In this case, it is assumed that when analyzing the topology, the attacker will find the minimum path, but then will not be able to use it either for delivering malicious software or for hacking.

Incorrect topology analysis can make an attack very difficult or impossible. If the topology analysis uses several systems and it is impossible to identify all possible attack paths due to the size of the system, then you can embed additional vertexes-detectors. The essence of these types of vertexes is that at a certain stage, information gets to an independent evaluation, automated or manual, depending on the type of system. In practice, each of these vertexes assumes the definition of a separate type of threat,
which allows you to differentiate the levels of access to information and its forwarding, which in turn leads to minimizing user errors.

Thus, the use of graph theory allows you to: monitor intersystem security; track the shortest ways to bypass the system; fix potentially dangerous data transfer paths within the system; check the system at all stages of development and for all elements of the system, in order to minimize risks; assess the possibility of security violations due to the human factor.

The main disadvantages of this method in modern conditions are the high growth rate of information technology systems and the high complexity of the mathematical apparatus. Based on this, graph theory is most often used in the design of individual elements of the topology of interaction of system elements. Most often, it focuses on creating false vertexes that give the appearance of a highly complex data network.

3.2. Application of neural networks

Currently, additional tools such as antivirus programs, firewalls, and security scanners are usually used to ensure the security of information. Not enough attention is paid to software security. Analytical reports of major companies in the field of software information security show that the number of vulnerabilities detected in software has increased significantly in recent years.

To increase the efficiency of signature detection, it is rational to use neural networks. The activation function can be different and is generally written as:

\[
y = f(s).
\]

The network itself consists of layers of neurons. Usually, the neurons of one layer are not connected to each other, and their outputs are inputs for the neurons of the next layer [14-17]. Starting from the need to identify the specified constructs, to train a neural network, it is necessary to use the method of reverse error propagation, which is aimed at minimizing the difference between the actual and expected outputs of the network by changing the weights of synapses. If there are pairs of input data and output images \(((x_1^p, x_2^p, ..., x_m^p), (y_1^p, y_2^p, ..., y_m^p))\), the value that characterizes the discrepancy between the actual output and the expected one is determined by the expression:

\[
E = \frac{1}{2} \sum_{k=1}^{P} \sum_{p=1}^{P} (t_k^p - y_k^p)^2.
\]

Here \(t_k^p\) is the \(k\) value in the \(P\) output image; \(y_k^p\) is \(k\) the value in the actual output data when the \(P\) input image is fed to the input \((x_1^p, x_2^p, ..., x_m^p)\). Thus, all sets of input signals are first passed through the network, receiving a set of output signals, and then the discrepancy value is calculated.

Improved safety due to faster signature detection could not provide a complete security system. The main part of the largest cyber-attacks is called DDoS attacks. A DDoS attack is a simultaneous attack from a large number of computing devices connected to a single information technology system in order to disable the system. All methods of detecting attacks can be divided into two large categories. These are methods for detecting anomalies and methods for fixing abuse.

Activities aimed at detecting abuse are based on creating templates and finding signs of known attacks. The main advantage of this method is that it practically eliminates false positives. The disadvantage of this method is the inability to detect attacks that are not built into the system. As a result, it is necessary to maintain and regularly update a large database containing each attack and all its possible variants [18].

A more flexible method is the detection of abnormal activity, which allows you to detect unknown attacks, but can often lead to false positives. To detect an attack, you must compare the current values of the activity characteristics with the standard values, and in case of any deviation from the standard behavior, you must consider this situation as a violation. Data averaged over a sufficiently long period of time (from one day to several months) is considered normal [18].
Examples of abnormal behavior include a large number of connections over a short period of time or high CPU usage. It should also be understood that abnormal behavior should not always be considered an attack. Thus, the attack does not consist of a large number of responses to the station activity request from the network management system. A fairly rare update of the database with normal behavior characteristics allows attackers to adapt their behavior to the requirements of the system for detecting abnormal activity, which perceives it as a legal user [18].

The main mechanisms for preventing DDoS attacks are shown in table I. All methods of protection against DDoS attacks are divided into two blocks: methods that precede the start of the attack, aimed at preventing the very fact of the attack, and methods of active counteraction and weakening the results of the attack, which are used immediately after the start of the attack [19]. Methods to prevent attacks include organizational and legal measures.

**Table 1. Comparative analysis of the main protection mechanisms**

| Criterion / method               | State analysis | Neural network | Expert system | Signature-based methods | Statistical technology |
|----------------------------------|----------------|----------------|---------------|-------------------------|-----------------------|
| The observation level            | Network, OS, apps | Network, OS  | Network, OS  | Network, OS, apps       | Network, OS           |
| Anomalies                        | -              | +             | +             | -                       | +                     |
| Corrupt practices                | +              | +             | +             | +                       | -                     |
| Verifiability                    | +              | -             | +             | +                       | -                     |
| Adaptivity                       | -              | +             | +             | -                       | +                     |
| Stability                        | +              | -             | +             | +                       | -                     |
| Computational complexity         | Low            | Average       | High          | Low                     | Average               |

If the attack still took place, then after the start of the attack, you must take active measures to counter the attack as soon as possible. The most important of these measures are increasing resources and filtering traffic.

Resource growth is preceded by a detailed analysis of server and network segment usage to identify bottlenecks. If, for example, the server occupies a significant part of the communication channel during normal operation, it can be assumed that an attacker can completely fill the channel with malicious requests in the event of an attack. In this case, it is advisable to increase the bandwidth of the communication channel in advance [20].

**4. Evaluation of the survivability of ITS**

When forming fault tolerance strategies, you need to define a set of States of the system in which it is necessary to counteract threats to health, as well as set options for distributing functions between functional components of the information technology system in the States of the set $S^{(f)}$. This strategy is aimed at fully compensating for the envisaged functional failures and ensuring the performance of systems in these cases.

Creating survivability strategies for each state of a set of $S^{(f)}$ you need to further develop solutions related to the system functions: whether to narrow the set of functions that together make up the purpose of functioning; how to do it; whether to simplify the algorithm for implementing functions, etc. Опишем варианты решения относительно цели функционирования системы в условиях наличия нежелательных влияний может быть одним из таких:
• many functions of the system cannot be changed, all functions must be used, possibly with less efficiency or with poor quality, that is, in any state of the $S^{(f)}$, the condition must be met:

$$\prod_{i\in I} x(f_i) = 1, x(f_i) = \begin{cases} 1, & \text{if executed } f_i; \\ 0, & \text{if not executed } f_i. \end{cases}$$  \hspace{1cm} (17)$$

• in any state of the $S^{(f)}$, some subset of functions that implement the purpose of the information system functioning must be performed $F^*$

$$\prod_{f\in F^*} x(f) = 1.$$  \hspace{1cm} (18)$$

The set of $F^*$ functions depends on the state of the system and the set conditions for functional survivability. In an arbitrary state from $S^{(f)}$, the system must provide at least one function from the set of $F^*$

$$\sum_{i\in I} x(f_i) \geq 1.$$  \hspace{1cm} (19)$$

The functional viability of an information system depends on the pre-defined purpose of its functioning. The functional viability of different information systems can only be compared if they have the same functioning goals. The assessment of the survivability of the same information system may change if the purpose of operation changes. At the same time, the parameters that determine the conditions of their operability have as significant an impact on the quantitative indicators of the survivability of information systems as the purpose of their functioning.

5. Conclusion

The article discusses methods for assessing the reliability of information technology systems, describes various types of vulnerabilities that the information technology system is exposed to. The main steps that should be taken by an attacker during a cyber-attack are shown. Methods of vulnerability detection using graph theory are considered and methods of their elimination are shown.

The necessity of using neural networks in traffic analysis in order to detect signatures is justified. The mechanism of DDoS attacks is analyzed and methods of their prevention are shown. The analysis of the main mechanisms of protection against DDoS attacks is carried out.

Acknowledgments

The reported study was funded by RFBR, project number 19-31-51017.

References

[1] Matveevsky V R 2002 Reliability of technical systems: textbook (Moscow: Moscow State Institute of Electronics and Mathematics)

[2] Polovko A M and Gurov S V 2006 Fundamentals of reliability theory: workshop (St. Petersburg: BHV-Petersburg)

[3] Vikharev S M and Barinov A A 2007 Reliability of automated systems: textbook (Kostroma: KSTU publishing House)

[4] Yamada S. 2002 Software Reliability Models Stochastic Models in Reliability and Maintenance pp 253–280.

[5] Peña E.A., Hollander M. 2004 Models for Recurrent Events in Reliability and Survival Analysis. Mathematical Reliability: An Expository Perspective. International Series in Operations Research & Management Science vol 67 pp 105–123.

[6] Afanasyeva L.G., Tkachenko A. 2016 On the Convergence Rate for Queueing and Reliability Models Described by Regenerative Processes Journal of Mathematical Sciences vol 218 pp 119–136.
[7] Rahmoune F., Aïssani D. 2014 Quantitative Stability Estimates in Queues with Server Vacation. *Journal of Mathematical Sciences* vol 200 pp 480–485.

[8] Polovko A M and Gurov S V 2006 *Fundamentals of reliability theory* (St. Petersburg: BHV-Petersburg)

[9] Mazzola G. et al. 2020 Software Tools and Hardware Options *The Future of Music* pp 35–80.

[10] Ceschia S., Schaerf A. 2013 Local search for a multi-drop multi-container loading problem. *Journal of Heuristics* vol 19 pp 275–294.

[11] Vasilenko N V and Makarov V A 2004 Model of software compaction. *Bulletin of the Novgorod state University* Vol 28 p 126–132.

[12] Murgu A., Postlethwaite I., Gu D., Edwards C. 2010 Topology Information Control in Feedback Based Reconfiguration Processes. *Dynamics of Information Systems* pp 101–124.

[13] Zeng Z., Wen M., Kang R. 2013 Belief reliability: a new metrics for products’ reliability. *Fuzzy Optimization and Decision Making* vol 12 pp 15–27.

[14] Kalan R 2001 *Basic concepts of neural networks* (Moscow: Williams)

[15] Nalini D.N., Singh K.J., De T. 2018 ICMP-DDoS Attack Detection Using Clustering-Based Neural Network Techniques. *Operations Research and Optimization* vol 225 pp 211-225.

[16] Ahmed M., Naser M.A., Hu J. 2016 A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* vol 60 pp 19–31.

[17] Sun Y., Guo L., Li Y., Xu L., Wang Y. 2020 Semi-supervised Deep Learning for Network Anomaly Detection. *Algorithms and Architectures for Parallel Processing* vol 11945 pp 383–390.

[18] Lukatsky A V 2003 *Detection of attacks* (St. Petersburg: BHV-Petersburg)

[19] Ternovoy O S 2016 Methods and means of early detection and counteraction to threats of information security violations in DDOS attacks: (dissertation for the degree of candidate of technical Sciences) (Barnaul)

[20] Tarasov Ya V 2014 Method for detecting low-intensity DDOS attacks based on a hybrid neural network, infrastructure. *News of the southern Federal University. Technical Sciences* Vol 8 p 47–87.