Statistics of software vulnerability detection in certification testing

Statistics of software vulnerability detection in certification testing

A V Barabanov[1], **A S Markov**[1], **V L Tsirlov**[2]

[1] Bauman Moscow State Technical University, 5, Baumanskaya 2-ya St., Moscow, 105005, Russia
[2] NPO Echelon, 24, Eletrozavodskaya St., Moscow, 107023, Russia

E-mail: A.Markov@bmstu.ru

Abstract. The paper discusses practical aspects of introduction of the methods to detect software vulnerability in the day-to-day activities of the accredited testing laboratory. It presents the approval results of the vulnerability detection methods as part of the study of the open source software and the software that is a test object of the certification tests under information security requirements, including software for communication networks. Results of the study showing the allocation of identified vulnerabilities by types of attacks, country of origin, programming languages used in the development, methods for detecting vulnerability, etc. are given. The experience of foreign information security certification systems related to the detection of certified software vulnerabilities is analyzed. The main conclusion based on the study is the need to implement practices for developing secure software in the development life cycle processes. The conclusions and recommendations for the testing laboratories on the implementation of the vulnerability analysis methods are laid down.

1. Introduction

The analysis of software vulnerabilities is currently the major activity performed by experts of testing laboratories (TL) of Russian information security certification systems. This type of work is performed both in the course of certification for compliance with the requirements of the protection profiles, which explicitly include the requirements of the Vulnerability Analysis trust family, as well as in testing for compliance with the requirements of technical specifications or classical guidelines [1]. The vulnerability analysis methods consist of the joint use of the approaches set forth in national standard ISO/IEC 18045 and international standard ISO/IEC TR 20004. In general, the methodology assumes the following steps.

- Identifying known (confirmed) vulnerabilities of the certification object. At this step, TL experts search for known (confirmed) vulnerabilities in publicly available information sources, for example: in the Information Security Threats Database of the FSTEC of Russia or CVE resource.
- Identifying previously unpublished vulnerabilities of the certification object. At this step, TL experts define a list of potential vulnerabilities of the certification object based on the analysis of certification object data (source code, available documentation, information from open

sources) and develop and execute a penetration test for each identified potential vulnerability to make sure that the assumption is correct.

Since the requirements for vulnerability analysis are relatively new for Russian information security certification systems, to date there are practically no methodological documents for TL that could be used to conduct an effective vulnerability analysis. This is a reason for the urgency of developing and improving the methodological support for the vulnerability analysis in conducting certification tests under information security requirements. As part of this study, a combined methodology for the analysis of software vulnerabilities was approved, and recommendations for experts of the accredited TL were formulated.

2. Customized methodology for the software vulnerability analysis

As part of the study, the combined methodology for the analysis of vulnerabilities was customized to meet the NPO Echelon accredited TL peculiarities (Fig. 1).

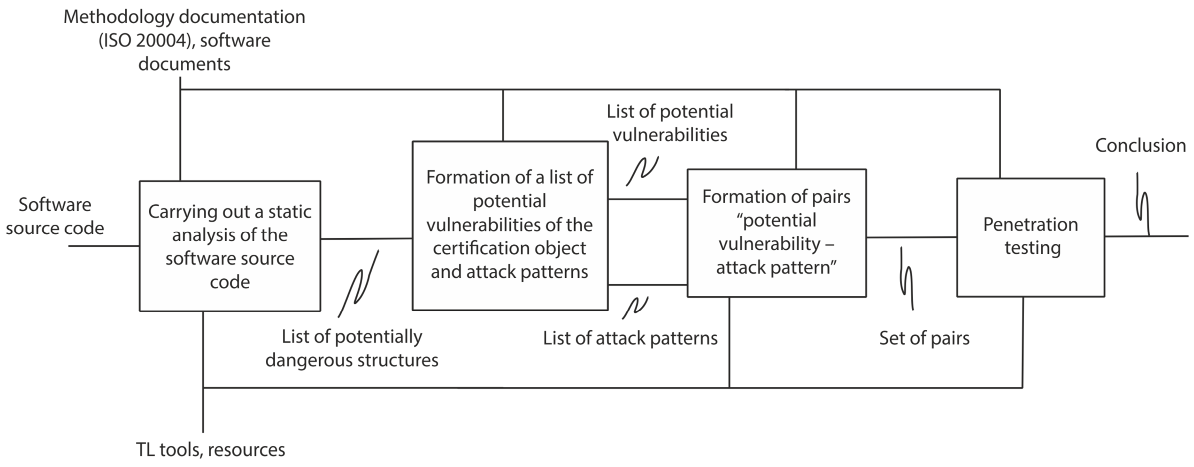
Methodology documentation (ISO 20004), software documents
Software source code
Carrying out a static analysis of the software source code
List of potentially dangerous structures
Formation of a list of potential vulnerabilities of the certification object and attack patterns
List of potential vulnerabilities
List of attack patterns
Formation of pairs "potential vulnerability – attack pattern"
Set of pairs
Penetration testing
Conclusion
TL tools, resources

Figure 1. An IDEF0 diagram of the adapted methodology for the software vulnerability analysis.

A brief description of the stages and steps of the customized methodology for the software vulnerability analysis is presented below.

Stage 1. Static analysis of the software source code [2, 3].

- Step 1. Identifying a variety of source code that compile the certification object. At this step, TL experts check the software source codes to be tested for completeness and lack of redundancy in order to determine the exact variety of source codes that participate in the software compilation. At this step, TL experts use information generated by the assembly system and various tools (file system monitors, file system audit programs, etc.). The main goal of this step is to register the list of source code files that participate in the compilation of the certification object.
- Step 2. Static signature analysis [4] with respect to the source code variety established at step 1. The static analyzer should be able to search for potentially dangerous structures in the source code and generate this list, assigning a CWE base identifier to each detected potentially dangerous structure.

Stage 2. Making a list of potential vulnerabilities of the certification object and attack patterns.

- Step 3. Processing the obtained list of potentially dangerous structures using the filtering criteria presented in Section 6.1.2.1 of the ISO/IEC TR 20004 standard.
- Step 4. Making a list of attack patterns that are relevant for the test software using the sequence of actions described in Section 6.1 of the ISO/IEC TR 20004 standard. At this step, in addition to the information presented in the source codes of the certification object, TL

experts use documentation (technical, software, operational) provided for testing and the information about the known software vulnerabilities which are similar to the certification object.

Stage 3. Formation of pairs "potential vulnerability – attack pattern."

- Step 5. Processing the list of potential vulnerabilities and attack patterns obtained at Stage 2, using the sequence of actions described in Section 6.1.2.2 of the ISO/IEC TR 20004 standard.

Stage 5. Performing a penetration test.

- Step 6. Development of penetration tests on the basis of the resulting list of potential vulnerabilities and attack patterns.
- Step 7. Installation of a test bench and performance of penetration tests using the developed code.
- Step 8. Determining the actual SW vulnerabilities based on the penetration tests and issuing reports.

3. Experimentation

Experimental studies of the customized methodology for the software vulnerability analysis were conducted for 2 years (2016-2017) at the research base of NPO "Echelon" by experts of the accredited TL. The test object was software that was undergoing case and certification tests in the accredited TL (a total of 76 test objects). Fig.2 shows the distribution of the tested products by types. Depending on the certification criteria determined by the potential scope of the certified product, the TL experts were provided or were not provided with access to the certification object source codes (Fig. 3). For the signature analysis of the source code, TL experts used the static analysis tool such as AppChecker (developed by NPO Echelon). For the penetration tests, the TL experts used recommendations from various issue-related resources (CAPEC, OWASP) [5-12]. The TL experts installed and adjusted the test benches used for penetration testing (step 7) in full compliance with the requirements of operational and technical documentation for test objects.

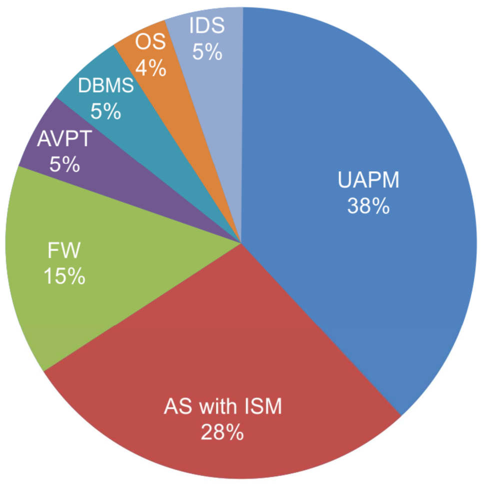

Figure 2. Distribution of the tested products by types: UAPM – unauthorized access protection means; AS with ISM – application software with the embedded information security means; FW – firewall; AVPT – anti-virus protection tool; DBMS – database management systems; OS – operating system; IDS – intrusion detection system.

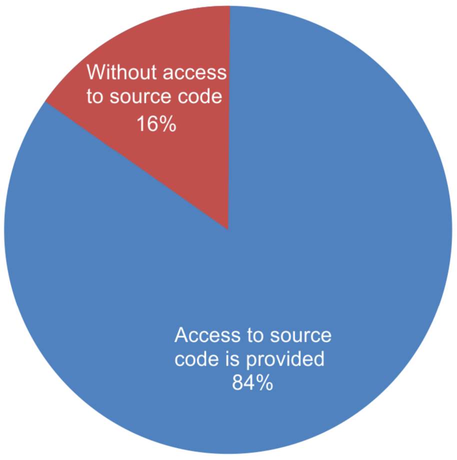

Figure 3. Distribution of the tested products depending on the access to source code.

4. Experimental results

As a result of the tests, 81 vulnerabilities were identified by the TL specialists of NPO “Echelon” (vulnerabilities were found in 26 from 76 products tested). For all detected software vulnerabilities, it was confirmed that they are challenging on the part of the software developer, and software developers took measures to eliminate the identified vulnerabilities.

Fig. 4 shows the distribution of the detected vulnerabilities in terms of their criticality (assessment was performed using the CVSS version 3.0) [13, 14].

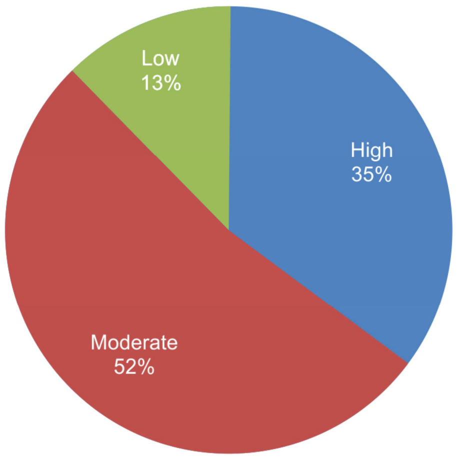

Figure 4. Distribution of the identified vulnerabilities depending on the degree of their criticality.

The most popular type of vulnerable software is application software with the incorporated information security tools (Fig. 5).

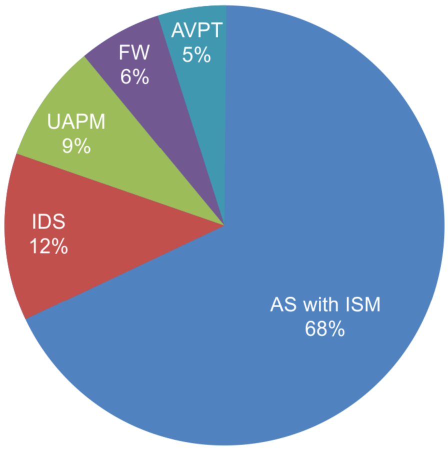

Figure 5. Distribution of the identified vulnerabilities depending on the type of the software tested.

The main types of vectors of successful attacks, which were developed by the TL experts to confirm the relevance of the vulnerability (Fig. 6), are:

- Cross-site scripting (CAPEC-63).
- Cross-site request forgery (CAPEC-62).
- Enhancement of privileges related to circumvention of security functions (CAPEC-233).
- Attacks aimed at denial of service (CAPEC-2).
- Disclosure of critical software information in error messages (CAPEC-54).
- SQL injection (CAPEC-66).

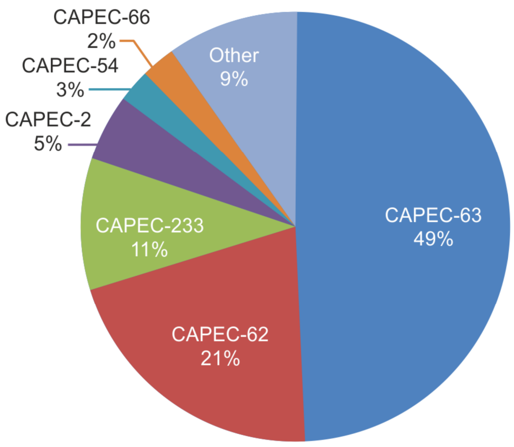

Figure 6. Distribution of the identified vulnerabilities depending on the type of the attack vector.

Types of attack vectors falling in the Other category are as follows: remote execution of operating system commands by data transfer in HTTP requests (CAPEC-76), XML injection (CAPEC-250), session fixation (CAPEC-61), going out of limits of the designated directory (CAPEC-126), Reparse-Point, RegSafe/RegRestore attack types.

The main types of software shortcomings that have become the causes of vulnerabilities are as follows (Fig. 7):

- Incorrect use of data obtained from an untrusted source to generate an HTML page (CWE-79).
- Use of authentication data (cookie data) for request authorization (CWE-352).

- Incorrect use of data obtained from an untrusted source when performing security functions (CWE-807).
- Lack of authorization for critical operations (CWE-862).
- Incorrect use of data obtained from an untrusted source to generate a query to the DBMS (CWE-89).
- Incorrect error message generation (CWE-209).

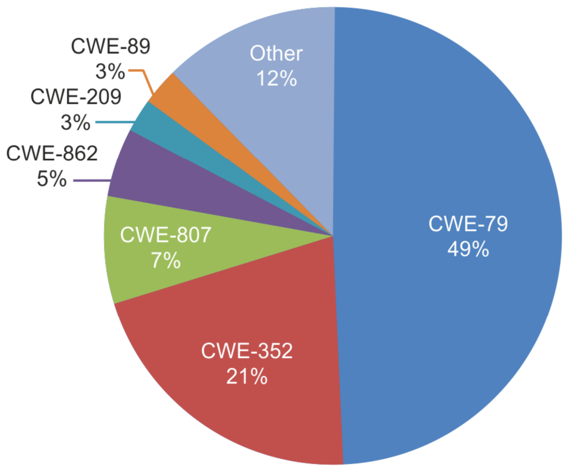

Figure 7. Distribution of the identified vulnerabilities depending on the software error (defect).

Software errors falling in the Other category are as follows: use of the authentication data specified in the code of the program (CWE-798), buffer overflow (CWE-120), errors leading to session fixation (CWE-384), misuse of data obtained from an untrusted source, to generate OS commands (CWE-22), etc. The tests have shown that the software can explicitly include program bookmarks masqueraded as debugging tools, for example, built-in accounts and master passwords.

Speaking about the methods of making a list of potential vulnerabilities, it should be noted that most of the vulnerabilities were discovered through assumptions made on the basis of review of documentation for the certification object and data on vulnerabilities in products similar to the certification object (Fig.8).

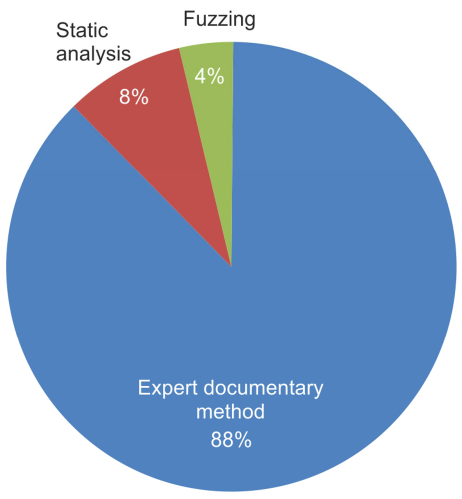

Figure 8. Distribution of the identified vulnerabilities depending on the method of forming a list of potential vulnerabilities.

The average time to fix the vulnerability by the software developer was 3 weeks.

5. The state of the problem in foreign certification systems

It should be remembered that due to innovations in foreign certification systems [15], TL reports that general information about the vulnerability analysis conducted is published in official network resources of certification systems. The TL reports for the period of 2016-2017 were analyzed (33 reports were selected) and published on the website of the NIAP as a regulator of the US certification system. Among the analyzed reports, most of the reports were those prepared by the results of testing for compliance with the requirements for protection profiles of network devices (28 reports). The rest of the reports (5 reports) reflected the results of testing for compliance with the requirements for protection profiles for application programs, operating systems, access control policy tools and mobile device protection tools.

The main analysis results are presented below.

- In the course of every study, foreign TL always searched for information about the known vulnerabilities of the certification object in public databases. Some TL searched for the known vulnerabilities not only by keywords directly related to the certification object (software name and version, software developer's name), but also by identification data related to the imported components.
- Only in half of the studies, the testing laboratories performed additional penetration tests. In most cases, a standard set of tests, applicable to virtually all types of certification objects (for example, scanning network ports), was used. Only one paper provided information on penetration testing based on potential vulnerabilities of the certification object formulated taking into account the analysis of the developer's evidence.
- Fuzz testing was involved in all the activities related to certification by the requirements for protection profiles for network devices. In this case, as a rule, in-house software automation tools were used.
- In their work, the testing laboratories did not use ISO/IEC TR 20004 guidelines on making a list of potential vulnerabilities based on the analysis of CWE and CAPEC databases. The reason is that the requirement to provide access to the source code of the software being certified is not mandatory in foreign certification systems. The analysis is carried out only to the extent required by EP – additional research is performed only by a small number of testing laboratories.

6. Conclusions

The main conclusions obtained from the study are summarized below.

- The portion of vulnerabilities detected in Russian software is much larger than the portion of vulnerabilities detected in foreign software, even versus a significant difference between the quantities of Russian and foreign products studied. This is due to the significant differences in the maturity levels of life cycle processes of the secure software development [16, 17], introduced by foreign and Russian software developers. However, it should be noted that during the study of foreign software, the developers in most cases did not provide access to the source code of the test objects, which made it impossible in principle to make a list of potential vulnerabilities based on the study of the program source code.
- Most of the vulnerabilities identified in the study could be detected by software developers at the early stages of software development using software architecture analysis in terms of implementing information security threats and a static analysis of software source code.
- In order to reduce the number of vulnerabilities, software developers are recommended to implement in the life cycle processes the main activities aimed at developing secure software (GOST R 56939 [18]) – software architecture analysis in terms of implementing information security threats, static analysis of source code and security testing. The introduction of such procedures into the practice of Russian software developers will increase, in the authors'

opinion, the level of protection of the software being created and, as a consequence, significantly reduce the number of information security incidents.

References

[1] Barabanov A and Markov A 2015 Modern trends in the regulatory framework of the information security compliance assessment in Russia based on common criteria *The 8th international conference on security of information and networks (SIN '15)* 30-33
[2] Kozachok A, Bochkov M, Lai M T and Kochetkov E 2017 First order logic for program code functional requirements description *Voprosy kiberbezopasnosti* **3(21)** 2-7
[3] Reber G, Malmquist K and Shcherbakov A 2014 Mapping the application security terrain *Voprosy kiberbezopasnosti* **1(2)** 36-39
[4] Markov A S, Fadin A A and Tsirlov V L 2016 Multilevel metamodel for heuristic search of vulnerabilities in the software source code *International Journal of Control Theory and Applications* **9 30** 313-320
[5] Iskhakov S Yu, Shelupanov A A and Meshcheryakov R V 2017 Simulation modelling as a tool to diagnose the complex networks of security systems *Journal of Physics: Conference Series* **803 1** 012057
[6] Kinash N A, Trufanov A I, Tikhomirov A A, Berestneva O G and Fisochenko O N 2016 Network vulnerability in two-phase evolution *International siberian conference on control and communications (SIBCON 2016)* 16090550
[7] Lekies S, Kotowicz K, Groß S, Nava E A V and Johns M 2017 *The 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)* 1709-1723
[8] Mangathayaru N, Kumar G R and Narsimha G 2016 *The International Conference on Engineering and MIS (ICEMIS 2016)* 7745351
[9] Pellegrino G, Johns M, Koch S, Backes M and Rossow C 2017 *The 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)* 1757-1771
[10] Ryck P D, Desmet L, Piessens F and Joosen W 2015 *The 30th annual ACM symposium on applied computing (SAC '15)* 2171-2176
[11] Selim H, Tayeb S, Kim Y, Zhan and Pirouz M 2016 *The 3rd multidisciplinary international social networks conference on socialinformatics 2016 (MISNC, SI, DS 2016)* 45
[12] Vorobiev E G, Petrenko S A, Kovaleva I V and Abrosimov I K 2017 Organization of the entrusted calculations in crucial objects of informatization under uncertainty *The 20th IEEE International Conference on Soft Computing and Measurements (SCM 2017)* 299-300
[13] Allodi L and Massacci F 2014 *ACM Trans. Inf. Syst. Secur.* **17 1** 20
[14] Alsaleh M N and Al-Shaer E 2014 *The 2014 workshop on cyber security analytics, intelligence and automation (SAFECONFIG '14*) 25-28
[15] Biró M and Molnár B 2007 *LNCS* **4764** 31-45
[16] McGraw G 2015 *J. Comput. Sci. Coll.* **30 3** 7-8
[17] De Win B, Scandariato R, Buyens K, Grégoire J and Joosen W 2009 *Information and Software Technology* **51 7** 1152-71
[18] Barabanov A V, Markov A S and Tsirlov V L 2016 Methodological framework for analysis and synthesis of a set of secure software development controls *Journal of Theoretical and Applied Information Technology* **88 1** 77-88