International Journal of Intelligent Information Systems
2022; 11(3): 35-38
http://www.sciencepublishinggroup.com/j/ijiis
doi: 10.11648/j.ijiis.20221103.11
ISSN: 2328-7675 (Print); ISSN: 2328-7683 (Online)

Analysis of the Conditions of Experimental Evaluation Security of Applied Computer Process

Pavlo Khusainov[1, *], Serhii Shtanenko[2]

[1]Department of Cyber Security, Military Institute of Telecommunications and Informatization, Kyiv, Ukraine
[2]Department of Military Training, Military Institute of Telecommunications and Informatization, Kyiv, Ukraine

Email address:

indesys@ukr.net (P. Khusainov), sh_sergei@ukr.net (S. Shtanenko)
[*]Corresponding author

To cite this article:

Pavlo Khusainov, Serhii Shtanenko. Analysis of the Conditions of Experimental Evaluation Security of Applied Computer Process. *International Journal of Intelligent Information Systems*. Vol. 11, No. 3, 2022, pp. 35-38. doi: 10.11648/j.ijiis.20221103.11

Received: April 22, 2022; **Accepted**: May 6, 2022; **Published**: May 26, 2022

Abstract: The need to ensure the effective operation of entities of the National Cyber Security System stipulate the urgency of developing a scientific and methodological apparatus for rapid response to cyber incidents (cyberattacks). The fundamental impossibility of achieving algorithmic and information completeness of cyber defense equipment anticipates the implementation of a process to support the decision-making of the operational staff of cybersecurity. Another factor of decision uncertainty is the lack of a priori data to identify the magnitude of the damage from the effects of the cyber incident. The latter is due to the fact that the description of a cyber incident consists of a set of signs of detection of a possible (potential) cyberattack, but the amount of damage cannot be reliably known instantly. Determining the amount of damage at the moment of detecting a cyberattack can be done using security proof models of information security theory, based on the subject-object representation of the object of cybersecurity. The use of these models requires knowledge of the probability of protection against the imposition of unforeseen execution for applied computing processes of all types in the object of cyber security. The proposition is to evaluate the reliability from influence on the applied computational process in the form of an experiment. The experiment allows obtaining the most complete image of the possibilities and features of the use of typical vulnerabilities of the software implementation of the target computational process. The organization of the experiment is to establish the fact of the execution of an active code from the composition of special code combinations in input data. The result of the experiment: a description of the code combination of input data (subsets of possible combinations) the processing of which led to the execution of the active payload; the average time of the experiment with the target computing process before the transfer of control to the active payload. The article is devoted to the presentation of the results of the analysis of conditions that must be taken into account when organizing an experimental assessment of the possibility of influence on the applied computational process.

Keywords: Computational Process, Critical Data, Method of Imposing Code, Cyberattack, Vulnerability, Experiment

1. Introduction

Responding to global trends and challenges of today since 2017 in the information and legal space of our country appeared many legislative provisions that determine the legal and organizational basis for protecting the vital interests of citizens, society, and national interests of Ukraine in cyberspace, key goals, directions, and principles of the government policy in the cybersecurity area, roles and responsibilities of state bodies, enterprises, institutions, organizations, citizens in this area, basic principles of coordination of their activities to ensure cybersecurity [1, 2].

Following the Law of Ukraine "On Basic Principles of Cyber Security of Ukraine", cybersecurity is achieved and ensured by the quality implementation of comprehensive cybersecurity measures (organizational, legal, technical, cryptographic, and technical protection of information), aimed at preventing cyber incidents, detecting and protecting against cyberattacks, elimination of their consequences, recovery of stability and reliability of functioning cybersecurity objects. Cyber incident – an event or series of adverse events of unintentional nature (natural, technical, technological, erroneous, including due to human factors), and/or those that

have signs of possible (potential) cyberattack that threaten the security of electronic communications systems, control systems of technological processes that create a probability of violation of the regular mode of operation of such systems (including disruption and/or blocking of the system, and/or unauthorized management of its resources), endanger the security of information resources [3].

Nowadays, the primary approach for providing information security in automated systems is to organize access delimitation, relying on an access manager concept. Authority for access of the computing process to the objects of the information domain is determined utilizing means of the operating system based on the results of successful authorization of automated system user [4-6]. The most significant imperfection of the concept of access manager is the fundamental assumption of the sustainability of the proper functioning of applied computing processes throughout the operation time interval of the automated system, namely the inability to perform unforeseen actions at any time.

2. Analysis of Recent Publications on the Research Topic

An unprivileged authorized user of an automated system of the cybersecurity object cannot extend (change) its authority, but it can initiate targeted malicious insider activity at any time. At the same time, there are many examples of such information interaction between applied computing processes through system objects of the operating system (within their appropriate information domains), which can cause the imposition of unforeseen execution algorithm [7-9].

Thus, it could be stated, that the development of the scientific and methodological operational response to cyber incidents (cyberattacks) is closely related to the probability of protection applied computing processes in the object of cyber security against the imposition of unforeseen execution based on relevant probations. Based on the above, the article aims to analyze the conditions for the organization of experimental evaluation of the protection of the applied computing processes from the imposition of unforeseen by the developer execution based on modifications of its algorithm.

3. Summary of the Main Material

Modification of the target computational process algorithm by the destructive impact on the content of its RAM could be achieved by adding unforeseen values to critical data. Critical data of the target computational process will be any data located in its RAM, the imposition of unforeseen values on which allows the insider in a favorable way to modify (distort) the algorithm. In this sense, we can say that critical data determines the algorithm. Among the crucial data of the target program, we highlight data for management and other critical data [10-15].

The management data of the target computing process will be called data that directly or indirectly determine the flow of control transmission, more precisely, the value that at some stage of the target program processing will get into the processor counter. Such data may include, for example, function return addresses or pointers (stored stack pointer values) and function pointers. Imposing unpredictable values on the management data leads to unforeseen modification (distortion) of the control transfer flow in the target computing process. As a result, control could be (unforeseen) passed to third-party code entered into the address space of the target operation or, for example, to some privileged piece of code or function that allows running arbitrary external programs. Due to this, one of the regular programs or a third-party program, which was delivered by an insider into the target system previously, could be launched (unforeseen). Imposing unpredictable values on management data allows transferring control of the target computing process.

Other critical data of the target computational process include any data which do not affect the flow of control transfer (or if so, not so much that by their unforeseen modification could be provided an arbitrary control transfer, as in the case of critical data acting as operands of conditional structures). Among such critical data are the following significant groups:

1) commands transmitted by the target program to its environment;
2) authorization data of an individual for access to resources;
3) safety mode switches;
4) identifiers of resources used;
5) resource identifiers that provide access;
6) output state;
7) pointers to the listed data and indexes in their lists.

In order to modify the algorithm of the target computational process, it is necessary to unpredictably change some of its critical data during periods that are limited by the moment of initialization or modification of critical data and by the moment of their use.

Critical data should also be divided into those that directly and indirectly determine the algorithm of the target computational process. We will assume that critical data directly determines the algorithm when they are not pointers to other critical data; such critical data will also be called final critical data. Accordingly, critical data that are pointers to other critical data will be considered as indirectly determining the algorithm of the target computational process.

Modification of the algorithm of the target computing process (by the destructive impact on the content of its RAM) is always based on falsification of certain critical data that directly determine the algorithm of its operation and based on falsification of certain final critical data of the program. Falsification of critical data can be performed directly or indirectly. The direct falsification of critical data is the imposition of unpredictable values on the data, for example, by initializing them with such values or their unforeseen modifying.

For cyber attacks, which are implemented by the method of imposing code (injection-based attacks) involving the

imposition on the target computing process of content with its subsequent activation in the relevant information domain of access provides the following concepts:

1) attack vector;
2) payload;
3) payload activation zone;
4) payload activation impact.

The *attack vector* is a mechanism of payload injection into the environment of the target computing process (including the format of messages that deliver payload and the interaction protocol within which payload is delivered to it).

The *Payload* is data that can be activated, and the injection of which into the environment of the target computing process and further activation in this environment ensures the success of cyberattacks.

The payload activation zone is the element or stage of execution of the target software, which activates the injected content.

Conditions for the implementation of cyber attacks of this class:

1) software implementation of the boot module of the target computing process is vulnerable to destructive effects on the content of its RAM;
2) the insider (the attacker) has the opportunity to provoke the activation of this vulnerability, to be specific, to transfer data to the target computational process to activate the vulnerability.

Typical vulnerabilities in the boot module of the target computing process:

1) buffer overflow;
2) the overflow of the bit grid of integer variables;
3) imposing format strings.

A *buffer overflow* (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. As a result, the program attempting to write the data to the buffer overwrites adjacent memory locations. In the case of programs with open memory, a sequential buffer overflow in a simple form allows performing unforeseen modification of only the data located directly above the vulnerable buffer (within certain limits) in one of its segments; in a complex form, due to unforeseen modification of pointers located above the vulnerable buffer, which determine the addresses at which further write or read data, allows performing unforeseen modification or unforeseen reading of data located elsewhere in the address space of the vulnerable process; in general – unforeseen modification or unforeseen reading of any area of the address space.

The *overflow of the bit grid of integer variables* occurs when it is possible to provoke the assignment of one or another of its integer variables values that are outside the allowable range according to their type. When the bit grid of an integer variable overflows, the higher bits of the binary representation of the value assigned to that variable, which does not fit in the memory area allocated for that variable, are simply discarded.

Forms (methods of execution) of the overflow of a bit grid of integer variables (types of destructive influence on values of integer variables):

1) the overflow of the bit grid of an integer variable from above;
2) the overflow of the bit grid of the integer variable from below.

The overflow of the bit grid of an integer variable from above occurs when a value of this variable exceeds the maximum allowed.

The overflow of the bit grid of the integer variable from below occurs when a value of this variable is less than the minimum allowed. Overflow of the bit grid can occur not only in the case of explicit assignment of values to integer variables, but also in the transmission of arguments to functions, as well as during placement in memory of functions results. Overflowing the bit grid of integer variables is in itself a destructive effect on the content of the program's RAM. It can directly (without performing additional destructive effects) compromise its security.

Security compromising of the program due to overflow of the bit grid of an integer variable can cause looping or distortion of the algorithm logic. Moreover, it can create conditions for additional destructive effects on the content of the program's RAM.

Format string vulnerability occurs when it is possible to provoke the use of externally provided text strings as format strings. In the case of programs with non-protected memory, imposing format string exploits can be used to crash a program or to execute harmful code. The problem stems from the use of unchecked user input as the format string parameter in certain functions that perform formatting. A malicious user may use the format tokens to print data from the stack or other memory locations. There is the possibility to write arbitrary data to arbitrary locations using the specific format token, which commands functions to write the number of bytes formatted to an address stored on the computing process stack.

General reasons for the vulnerability of the target computing process to the destructive impact on the content of RAM:

1) lack or insufficiency of procedures to ensure the correctness of the input data;
2) incorrect implementation of input data processing procedures.

Often, this class of vulnerabilities is the consequence of incorrect assumptions of developers that input data may not be incorrect.

Typical attack vectors of this class, namely methods of entering destructive data into the RAM of the target computing process (generalized):

1) through data files intended for processing;
2) via command line arguments;
3) through environment variables;
4) through configuration files;
5) through files with additional data;
6) through responses to requests for the input of extra data;
7) via messages received through communication channels.

An insider-controlled computing process can be located both: in the same computing environment as the target

computing process and remotely relative to it in the same computing environment of another element of the cybersecurity object.

4. Conclusion

Attack vector choosing is the main element of experimental security evaluation of the computational process from the imposition of unforeseen algorithm execution. The proposal is to define the network interface as the attack vector. Payload will be included in the message and transmit to the network interface, designed to impose a bytecode to open a session via a network with a command line interface. The payload activation zone is the moment of execution of the computational process from a vulnerable function. Content activation zone - the moment of execution harmful code and exploitation of the computational process.

The further research direction is developing a method of experimental estimation of the average time for imposing on the target computational process the number of bytes (code) needed to open a session through a network with a command-line interface.

References

[1] L. Slipachuk, S. Toliupa and V. Nakonechnyi, "The Process of the Critical Infrastructure Cyber Security Management using the Integrated System of the National Cyber Security Sector Management in Ukraine", 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), 2019.

[2] Toliupa, S., Parkhomenko, I., & Shvedova, H. Security and regulatory aspects of the critical infrastructure objects functioning and cyberpower level assesment. In 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019 – Proceedings (pp. 463–468).

[3] Zakon Ukrayiny № 2163 VIII ot 05.10.2017 «Pro osnovni zasady zabezpechennya kiberbezpeky Ukrayiny».

[4] Khusainov P. V. Osnovy pobudovy operatsiy-nykh system, kompleksiv ta zasobiv avtomatyzatsiyi upravlinnya viys′kamy: Navchal′nyy posibnyk/ P. V. Khusainov, I. YU. Subach, O. V. Silko, S. V. Lyu-bars′kyy. – K.: VITI, 2016. – 220 s.

[5] Azarenko Ye. V. Proyektirovaniye avtomati-zirovannykh sistem upravleniya na komp'yuternykh setyakh: Monografiya/ Ye. V. Azarenko, B. M. Gerasimov, B. P. Shokhin. – Sevastopol': Gos. Okeanarium, 2007. – 272 s.

[6] Terminolohiya v haluzi zakhystu informatsiyi v komp″yuternykh systemakh vid nesanktsionovanoho dostupu: ND TZI 1.1–003–99. – Kyiv: DST·SZI SB Ukraine, 1999. – 26 s.

[7] D. C. Wardell, R. F. Mills, G. L. Peterson, M. E. Oxley, A method for revealing and addressing security vulnerabilities in cyber-physical systems by modeling malicious agent interactions with formal verification, Procedia Comput. Sci. 95 (2016) 24–31.

[8] Y. Albrekht and A. Pysarenko, "Multimodular Cyberphysical Systems: Challenges and Existing Solutions,"2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT), 2020, pp. 376-379, doi: 10.1109/ATIT50783.2020.9349291.

[9] M. Elnour, N. Meskin, K. Khan, R. Jain Application of data-driven attack detection framework for secure operation in smart buildings Sustain. Citie. Soc., 69 (2021), p. 102816, 10.1016/j.scs.2021.102816.

[10] Gerasimov B. M., Divizinyuk M. M., Subach I. YU. Sistemy podderzhki prinyatiya resheniy: proyektirovaniye, primeneniye, otsenka effektivnosti. Sevastopol'.: SNIYAEiP, 2004. – 319 s.

[11] Gerasimov B. M., Kamyshyn V. V. Orhanizatsiyna erhonomika: metody i alhorytmy doslidzhennya ta proektuvannya. – K., Infosystem, 2009. – 212 s.

[12] Antonyuk A. O. Teoretychni osnovy modelyuvannya ta analizu system zakhystu informatsiyi: [monohrafiya]. – Irpin′: Natsional′nyy universytet DPS Ukrayiny, 2010–310.

[13] The Handbook of Artificial Intelligence, Vol. 1 Avron Barr and Edward A. Feigenbaum (Eds.) William Kaufman, Inc., Los Altos, Calif., 1981, 409 pp, ISBN 0-86576-005-5.

[14] Matematicheskaya Entsiklopediya. T. 1 (A - G). Red. kollegiya: I. M. Vinogradov (glav red) [i dr.] – M., «Sovetskaya Entsiklopediya», 1977, 1152 stb.

[15] Nabi, F., Yong, J., Tao, X., Farhan, M. & Naseem, N. (2021). Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems. Journal of Computer Science, 17 (11), 1046-1058. https://doi.org/10.3844/jcssp.2021.1046.105