A PROCEDURE FOR ASSESSING SECURITY UPDATES IN INDUSTRIAL SYSTEMS

K. V. Semenkov* and V. G. Promyslov**

*,**Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia

*✉ semenkov@ipu.ru, **✉ vp@ipu.ru

Abstract. This paper is devoted to the problem of applying cybersecurity updates (patches) for the software of instrumentation and control systems (ICS) with a long lifecycle. The problem is considered for the system operation stage. The main focus is on the large number of vulnerabilities found in software, the complexity of analyzing the impact of a vulnerability on system security, and the requirements for testing the compatibility of updates and software certification after changes have been made. Based on the Failure Mode and Effects Analysis (FMEA), a procedure is proposed to simplify the analysis of the impact of a vulnerability on cybersecurity. This procedure considers a smaller set of attack scenarios rather than each vulnerability separately. The analysis of attack scenarios also covers the effect of security measures. The procedure includes simple criteria for applying security updates based on the analysis results. An example of vulnerability analysis using this procedure is provided.

Keywords: vulnerability, patch, risk assessment, instrumentation and control system (ICS), cybersecurity, criterion.

INTRODUCTION

A main way of conducting cyberattacks is to exploit vulnerabilities in software. Despite advances in software development and testing technologies, the complexity of programs makes it impossible to guarantee that software is free of vulnerabilities. Intruders focus their resources on finding and exploiting vulnerabilities, whereas software developers and users are interested in finding and promptly fixing these vulnerabilities, and releasing and installing appropriate software security updates (patches). In general, a patch is often understood as a very wide range of software changes [1–4], which are either characterized by the approach to this patch as a/an process/object or related to the scope or nature of software changes. In informal communication of IT experts, one can also meet other similar terms, e.g., update, bugfix, or hotfix. For the purposes of this paper, let us assume the following.

Definition. A security update (patch) is a modification to installed software intended to eliminate software vulnerabilities without changing other functional characteristics of the software. ♦

A great deal of work has been done globally to accumulate information about vulnerabilities, and there are publicly available and constantly updated databases: CVE (USA) [5], the Data Bank of Information Security Threats (Russia, managed by the Federal Service for Technical and Export Control (FSTEC)) [6], CERT-FR (France) [7], and others. For each vulnerability on the list, the databases necessarily contain its description, impact assessment, and recommendations to eliminate the vulnerability or mitigate its negative impact. Software developers release security updates, which can often be installed automatically.

The experience gained in the world is systematized in international and national standards and methodological documents on the application of patches; for example, see [3, 8, 9]. The guidelines and recommendations of these documents are generally reduced to the following steps:

1) permanently monitor vulnerabilities in the software used;

2) analyze newly discovered vulnerabilities and assess cybersecurity risks;

3) determine further actions depending on the results of the risk assessment: accept the risk, eliminate the risk (apply a patch), etc.;

4) under a positive decision to install an update:

a) develop a plan for applying the update;

b) check the integrity and confidence of the update;
c) test the update;
d) install the update;
e) check the software status and configuration after the installation.

As we believe, however, the issues of practical application of the available data, primarily related to the large volume of analyzed information and its reliability, have not yet been fully settled.

In this paper, the problem of applying security updates [10, 11] in systems with a long lifecycle will be considered on the example of an instrumentation and control system (ICS), and a new solution method will be proposed. Below, a system with a long lifecycle is understood as a system whose operation and support stage lasts for several years or even decades.

According to the guidelines of regulatory and methodological documents [10–13], vulnerability identification, analysis, and assessment should be conducted throughout the entire lifecycle of a protected system. Since cybersecurity resources and the context of viewing the system differ significantly between lifecycle stages, vulnerability elimination problems and methods for addressing them also differ. This paper focuses on the issues of vulnerability assessment during the exploitation stage under the following assumption: at the end of the development stage, the developer has closed known vulnerabilities and applied adequate protection measures to mitigate the risk to an acceptable level. For the development stage, there is a diverse set of recommendations for secure software development [14].

Much attention is paid to the operation stage because, as our experience shows, the problem of vulnerability management most fully manifests itself at the operation stage and becomes more complicated over time. This is primarily due to the integral effect of several factors: the accumulation of detected vulnerabilities in the components used, the end of the developer's support period for some components, and the obsolescence of information security technologies embedded in the system design.

As the object of study, we choose complex software systems, i.e., sets of programs [15] with special system components and third-party components of general application. Assume that the total number of components in the system is sufficiently large: for simple systems, the patching problem seems to be not very serious due to a moderate number of vulnerabilities, which can be promptly monitored and eliminated in the operation process. As shown by the practice, "simple" systems for ICSs consist of at most a single computer; then the number of assets and their links allows describing the emerging security relationships by an access control model, which can be used in risk assessment for assets associated with detectable vulnerabilities.

Risk management in industrial facilities and the installation of updates for digital safety systems are important, both scientifically and practically. The problem of risk assessment for the ICS of nuclear power plants (NPPs) was reviewed in [16]; a comprehensive survey of the recent (2002–2020) publications on patch management was given in [17]. The contribution of this paper is the detailed description of top-level techniques (such as [1, 10]) and a novel, industrial control system-oriented, set of actions for deciding on the installation of updates.

The problem of patch management will be considered in terms of the functions performed by the system rather than the vulnerability of a particular component for which a patch is available. For example, the main function of an ICS is to control an industrial facility. Then the purpose of installing a patch for the operating system (OS) of a computer within the control system is not to protect the OS but to mitigate the risk of the facility's uncontrollability in case of vulnerability exploitation. For the solution, based on *Failure Mode and Effects Analysis* (FMEA) [18], we propose a risk-oriented method with criteria for applying updates. In addition to managing vulnerabilities by their types, the idea is to consider explicitly the impact of protection measures on the realizability of attacks by an intruder with certain capabilities. The approach described below allows comparing the newly discovered vulnerabilities with known ones from some classifier (e.g., the *Common Weakness Enumeration* (CWE) [19]) and answering the following question: Does the system have "an immunity" (a barrier) against a new vulnerability? Hereinafter we will understand a barrier as a certain set of protection measures that guarantee security in a definite attack scenario.

1. THE PECULIARITIES OF USING THE THREAT MODEL IN PATCH INSTALLATION

Most risk-oriented approaches to patch management involve risk assessment techniques formulated in the ISO/IEC 27005 standard [20]. According to these techniques, the analysis of a threat model, including vulnerabilities, threats, and an intruder, mainly influences the decision of risk acceptability or unacceptability and, consequently, the decision to patch the system. There are many methods for describing threat model elements and compiling their taxonomy; below we will discuss the most appropriate ones for risk assessment in complex industrial systems during the operation stage. Let us begin with the individual components of the threat model.

1.1. Vulnerability Analysis

Following the definition of a vulnerability from the ISO/IEC 27000 standard and FSTEC methodological documents [12], vulnerability is "weakness of an asset or control that can be exploited by one or more threats."

Vulnerabilities may have different nature. They can be related to the system properties embedded during development (weaknesses in the defense-in-depth architecture or cross-domain communication, implementation errors) or can appear due to incorrect application of protection measures (e.g., passwords). Vulnerability analysis is intended to establish the extent to which vulnerabilities can affect the security of the system and the assessment of confidence in the protection measures implemented [21]. Patches must be applied to a system if the vulnerability analysis reveals an unacceptable level of information security risk to the system (see the guidelines [9], *Fig. 3.1*). Let us demonstrate the problems arising in vulnerability analysis. For this purpose, consider the use of methodological guidelines for analyzing and applying patches in more detail.

The first problem that needs to be highlighted is the scale of the system. As mentioned above, a complex software system includes a large number of heterogeneous components and third-party applications, and cybersecurity requires monitoring a large number of vulnerabilities associated with both special system components and third-party products (e.g., vulnerabilities in the operating system, database management systems, web servers, interpreters, etc).

The number of newly discovered vulnerabilities increases every year. For example, Table 1 presents the corresponding figures for CVE and the FSTEC Data Bank in 2021–2023.

For a complex system, the flow of vulnerabilities can amount to tens or hundreds of vulnerabilities per day, and even the initial analysis of new vulnerabilities can require significant resources and costs for an organization.

Table 1

The number of vulnerabilities added to CVE and FSTEC Data Bank annually, in thousand

Database	2021	2022	2023
FSTEC Data Bank	6.4	7.5	9.1
CVE [22]	20.2	25.0	29.0

The next problem to be emphasized is that in all databases, vulnerabilities are described in a relatively free form, without a generally accepted standard. Descriptions can be either very brief or overly detailed, making their analysis even more complicated. Here are some examples of unsuccessful descriptions (Table 2). The CVE-2018-19932 vulnerability is described with many technical details (may be of interest only to software developers); the CVE-2023-36762 vulnerability has a too general characterization; finally, the CVE-2021-30618 vulnerability is included in the database almost without essential information.

There are works aimed at automating vulnerability description analysis (e.g., see [23, 24]), including those with machine learning algorithms. However, to the best of our knowledge, no available tools completely automate the analysis of real systems in practice.

The description problem is aggravated by the language barrier, which has a complex character. First, most vulnerability descriptions in international databases are written in English, fluently managed by far from all authors of such descriptions. In other words, even at the initial description phase, the essence of a vulnerability may be distorted or incompletely stated. Second, a large amount of information about vulnerabilities is transferred from international open databases to national ones, where the descriptions are usually moderated and translated into the local language. Thus, after such manipulations, national vulnerability databases may contain additional errors and inaccuracies that complicate vulnerability analysis and, vice versa, expanded and clarified descriptions. However, in the latter case, the developer of a component with an open vulnerability often loses feedback from the moderators: all clarifications made are available only in the national language.

Table 2

Examples of unsuccessful vulnerability descriptions

Vulnerability	Description
CVE-2018-19932	An issue was discovered in the Binary File Descriptor (BFD) library (aka libbfd), as distributed in GNU Binutils through 2.31. There is an integer overflow and infinite loop caused by the IS_CONTAINED_BY_LMA macro in elf.c.
CVE-2023-36762	Microsoft Word remote code execution vulnerability.
CVE-2021-30618	Inappropriate implementation in DevTools.

Therefore, an organization needs a sufficiently large staff of experts to monitor and analyze such a volume of vulnerabilities independently; experts will perform a professional analysis of this poorly structured information and assess the risk associated with the vulnerability to the protected system.

A possible way to reduce the amount of information analyzed is to prioritize vulnerabilities and focus on the most critical ones. The *Common Vulnerability Scoring System* (CVSS) [25–27] is one of the most well-known and widespread criticality assessment scales. This scoring system includes three groups of metrics: basic, temporal, and contextual. According to our experience, the last two either do not contain information or the information is specific for a particular scenario of program application. Therefore, we will use only the basic metrics. For CVSS 3.0 [26], they include:

- the attack vector (e.g., a network attack, a local attack, physical access, an attack on a related network protocol);
- the complexity of the attack;
- the necessary access rights to exploit the vulnerability;
- participation of a “normal” user to exploit the vulnerability;
- the possibility that the attack’s consequences will go beyond the system under study;
- the impact on the availability, confidentiality, and integrity of information resources controlled by the program in which the vulnerability is detected.

The basic metrics mainly reflect the properties of a software component, and the factors of the operating environment are considered only conditionally. Their values, calculated via expertise, are given in vulnerability databases.

Therefore, the basic metrics are primarily intended for developers and users of a separate software component and ignore its role in the information system. To perform a complete CVSS vulnerability analysis for further risk assessment, one should calculate the remaining groups of metrics or use other metrics and vulnerability databases that reflect the above aspects [2, 28]. Also, it may be necessary to recalculate the values of basic metrics to consider the specifics of a particular organization or system under study.

The transition from individual vulnerabilities to their classes seems to reduce the labor costs of risk analysis and assessment. Vulnerabilities can be classified in different ways depending on the subject matter and the level of detail of the system. In general, there are the following types of vulnerabilities reflecting the nature of assets [13]:

– hardware vulnerabilities,
– software vulnerabilities,
– network vulnerabilities.

This paper deals only with software vulnerabilities because security updates are mainly focused on them.

To pass from the separate implementations of vulnerabilities (unique in most cases) to their types, we apply the approach [14], linking vulnerabilities to program weaknesses. Let us take CWE [19], one of the most famous classifiers of weaknesses.

This list is a hierarchical, freely augmentable taxonomy of software and hardware flaws that can be used in security analysis tools.

The classifier is a multilevel tree with four levels of weaknesses: Root level, Base level, Class level, and Variant level. To facilitate user work, CWE contains the so-called views, i.e., a set of CWE records intended for specific tasks (e.g., software development, hardware development, and research).

Vulnerabilities in CWE are classified manually by experts; according to practice, the opinions of experts differ in many cases [29]. Also, research into fully automatic classification is ongoing, but without unambiguous results available so far (e.g., see [23]).

The CWE catalog can be used to create a secure development system. As we believe, however, it is of little utility for vulnerability classification to create a protection system due to an implicit relationship between the CWE class, attack methods (e.g., attacks from the CAPEC classifier [30]), and the intruder model. Indeed, the same weakness can be used in different attack scenarios by different intruders and cause different consequences.

1.2. Analyzing Threats and Characteristics of Intruders

In the context of cybersecurity, a threat can be defined as a set of conditions and factors that create a potential or real danger of violating information security [12].

Threat types are usually correlated to the way of exploiting vulnerabilities, each with different implications and prerequisites; as a rule, threat types can be associated with a violation of cybersecurity properties in one of the reference models (e.g., the Confidentiality–Integrity–Availability (CIA) model).

The mapping of cybersecurity properties to system properties is individual for a particular system; depending on the system under analysis, each threat type may affect any system property (reliability, availability, maintainability, and safety). There are several common classifications of threat types:

– The Data Bank of Information Security Threats (FSTEC, Russia) [6],
– MITRE ATT&CK [31],
– Microsoft’s STRIDE [32].

An intruder (threat agent, attacker) is an active element (subject) in a system that attempts to exploit a vulnerability. Examples are hackers, computer criminals, terrorists, industrial spies, and insiders [33]. A detailed classification of intruders is provided in the catalogs of FSTEC, CAPEC, MITRE ATT&CK, etc. [12, 30, 31]. Each classifier contains a set of attributes for intruders, e.g., type, the levels of competence and equipment, and the purpose of attack. Within the FSTEC model, intruders are further divided into external and internal. ISO/IEC 27000 and MITRE ATT&CK attribute an intruder by the purpose of attack: obtaining money, undermining reputation, gaining a competitive advantage, etc.

As an example of attribution according to FSTEC documents, we present the types of intruders and their competence levels.

The types of intruders according to FSTEC are:

- special services of foreign countries;
- terrorist and extremist groups;
- criminal groups (criminal structures);
- natural persons (hackers);
- competing organizations;
- developers of software and programmable digital items;
- suppliers of software and programmable digital items for supporting systems;
- providers of communication services and computing services;
- persons engaged for installation, adjustment, testing, commissioning, and other types of work;
- persons ensuring the operation of systems and networks or supporting systems of the operator (administration, security guards, cleaners, etc.);
- authorized users of systems and networks.

The competence levels (H1–H4) of an intruder according to the Russian regulator FSTEC are:

- basic capabilities for realizing information security threats (H1);
- increased capabilities for realizing information security threats (H2);
- medium capabilities for realizing information security threats (H3);
- high capabilities for realizing information security threats (H4).

This list may be supplemented by other intruder types, considering the peculiarities of the field where systems operate and the connection between the system under analysis and its environment.

1.3. Analysis of Threat Model Components: Some Conclusions

According to the aforesaid, clearly, the work on analyzing threat model components and assessing the risk from vulnerability exploitation by intruders requires the regular participation of experts with rich knowledge and skills in programming and information security risk assessment, both at the system level and at the level of separate components.

This work is very time-consuming and goes beyond the functions related to risk analysis for vulnerabilities. If a vulnerability needs to be eliminated by applying a patch, the system owner faces additional work and problems.

2. PATCH MANAGEMENT PROBLEMS

In Section 1, we have described the basic steps for deciding whether to patch or not, as well as the related problems. However, the difficulties do not stop there: having decided to patch, the system owner deals with new problems due to the complexity of this class of systems:

- The security update of a software component within a system is often released not separately but as part of a new version of the component. In this case, the functionality of the new version may require the additional testing of the component within the system. Replacing the existing version of a software component may terminate some ICS functions, causing the need to modify ICS software.
- Software components within a software system are interconnected by a chain of dependencies. Dependencies can be both horizontal (e.g., at the level of application software components) and vertical (at the level of OS components). Replacing any key component may entail replacing the rest and, in the worst case, replacing all components in the dependency chain. For a system with a long lifecycle, some components in a dependency chain may be no longer supported by the developer (no new versions exist for them), and the update in this case cannot be performed by simply passing to a new version.
- ICSs are characterized by a long lifecycle (the operation period may reach decades) and strict requirements for the procedure of their development and testing. Given the high rate of discovering new vulnerabilities, it seems natural to assume that new vulnerabilities will be found in the ICS software environment during the time between the release of the ICS software version and the launch of the system after commissioning. Also note that the suppliers of third-party components may stop supporting outdated versions of their products: in this case, there will be no security updates for new vulnerabilities.
- ICS software is tested and certified to work on certain hardware tools in a given program environment, and depending on the validity conditions of the system certificate, the application of a patch can lead

to a costly and lengthy re-certification procedure for the system.

An important factor is confidence in the source of updates and software developers. As shown by practice, an update may contain deliberately embedded vulnerabilities [2, 34, 35]. Moreover, confidence in the source of updates can change over time, from widespread use of programs to their prohibition, only based on risk assessments related to the social and political circumstances [36, 37].

Let us discuss the testing of updates. Installation and testing of updates may require testing of the entire ICS. In most cases, a hybrid digital twin is a solution to perform full-fledged tests comparable to tests on the real object. In such a twin, some elements of the real ICS equipment are used together with purely digital components [38].

Thus, the practice of sequentially analyzing separate vulnerabilities in components and patching those components can be used on simple systems only. For large and complex systems with a long operation period, it is necessary to find other solutions of the cybersecurity problem.

A list of problems described in the literature was compiled in the review [17]. As we believe, the following are the most important ones:

– The problems of applying patches to ensure information security and current tasks (e.g., continuous business or industrial processes) may be incompatible (see [17], *item 2* of *Table 5*).

– The process of applying patches requires additional resources and expert knowledge, which are not available in the companies operating the software (see [17], *items 5* and *6* of *Table 5*).

– The automatic testing of patches is extremely difficult, most patches are tested manually and the quality of testing is often unsatisfactory (see [17], *items 11* and *12* of *Table 5*).

– Verifying the correctness of an applied patch and eliminating the consequences of deployment errors are a rather difficult problem (see [17], *items 13* and *14* of *Table 5*).

Below we describe a novel risk-oriented approach to simplify vulnerability analysis significantly.

3. THE RISK-ORIENTED APPROACH TO VULNERABILITY ANALYSIS CONSIDERING PROTECTION MEASURES

The approach proposed here is intended for systems with the following characteristics:

– The software environment of a system is treated as a set of separate "black-box" components interacting with each other through a known set of interfaces.

– A system is created using the architectural principles of encapsulation and domain partitioning [39].

We will consider a system in terms of performing definite functions and examine the contribution of each component to the functions.

This approach proceeds from the following main idea: the vulnerability analysis of system software should answer the question of how dangerous a vulnerability is to a particular system function rather than how critical it is to a particular software component.

The vulnerability analysis method proposed below is based on Failure Mode and Effects Analysis (FMEA), a method for identifying weaknesses in the system architecture to improve its reliability and security [18]. FMEA was developed in the 1940s [40] and initially intended to analyze the reliability or security of technical systems and equipment. Later, modifications of this method were proposed to analyze cybersecurity problems for systems with digital components (System Failure Mode and Effects Analysis, SFMEA) [41, 42]. This paper provides general information about FMEA; the interested reader can find details from the extensive literature. Within FMEA, a system must have the following properties:

– definite targets in the form of requirements for its functions,

– established operation conditions,

– definite bounds,

– a hierarchical structure.

Together with the block diagram of the hierarchical structure of elements, FMEA uses block diagrams reflecting the hierarchy of functions performed by system elements and functional relationships between the elements, which makes the functional failures of the system traceable.

The following items are analyzed for each system component (Fig. 1):

– the causes of a given failure;

– a function or failure that can occur (the type of failure);

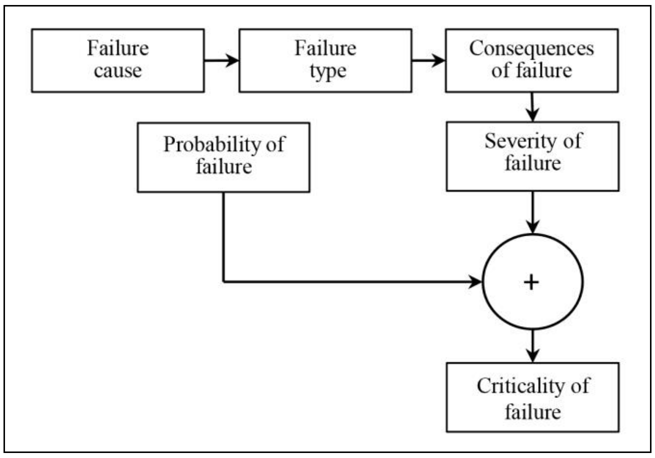

Fig. 1. The flowchart of failure criticality assessment according to the standard [18].

– the peculiarities of possible consequences in case of a failure;

– the severity of a failure (whether the failure is harmless or causes damage);

– the criticality of a failure (how and when the failure can be detected).

When applying FMEA approaches to cybersecurity assessment, it is necessary to describe FMEA stages in cybersecurity terms and relate the cause of a failure to the vulnerability and the intruder's ability to exploit it.

We propose adapting the FMEA methodology in order to assess the impact of a vulnerability, thereby reducing the amount of information under analysis (Fig. 2).

The methodology takes into account that a vulnerability in itself is not the cause of a failure. That is, a vulnerability leads to a failure if an intruder with sufficient competence exploits it to realize a definite threat (see block 1 in Fig. 2). The intruder and his/her competence can be typified according to an accepted scale:

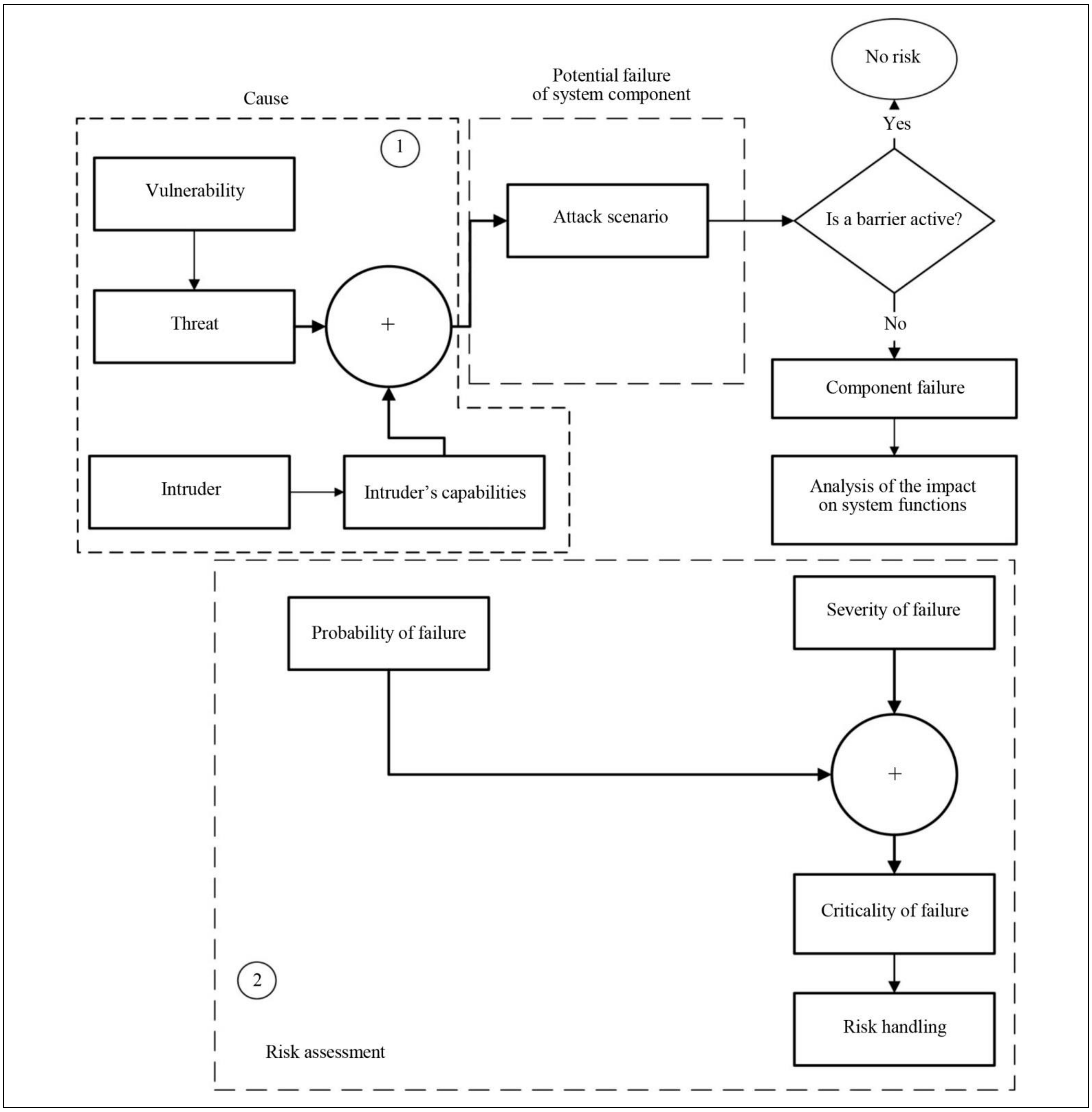

Fig. 2. The flowchart of failure criticality assessment with protection barriers for reliability and cybersecurity problems.

e.g., using the FSTEC classifier [12]. Thus, a failure is the result of a successful attack on the system by an intruder with sufficient competence and capabilities to exploit a vulnerability.

In the methodology proposed, the independence of vulnerabilities is postulated to reduce the analysis space. This is analogous to the assumption on the independence of failures in FMEA ([18], *section 4.1*).

In contrast to the typical consideration of separate technologies underlying a vulnerability (e.g., as in the CWE catalog), the analysis scheme proposed focuses on CVSS attributes to correlate the vulnerability and the associated threat. Additionally, we pass from analyzing separate threats and attackers to analyzing typical attack scenarios associated with threat classes and typical attacker capabilities, thereby reducing the number of threats and attack scenarios under analysis. For ICSs of NPPs, some examples of typical attack scenarios were described in [43].

In this case, a function failure is related not to a particular vulnerability (see Fig. 2) but to the set of conditions and factors that have led to the failure of a component and the intruder's penetration through the protection barrier. Note that a failure is not necessarily directly related to a (cyber)attack and vulnerability exploitation but, e.g., may be the result of resource exhaustion. However, the operation modes of critical facilities are designed so that to exclude resource exhaustion.

The second important supplement to the application of FMEA approaches to cybersecurity problems is to consider the presence of a barrier reflecting the effect of an already implemented set of protection measures. A barrier can block the impact of a component failure on the function of the entire system, thereby nullifying the risks associated with an attack on the system. By assumption, a barrier has a high degree of confidence and can counteract the exploitation of one or more types of vulnerabilities. Thus, when analyzing attack scenarios, a barrier is supposed to be absolute. This model assumption allows significantly reducing the analysis space.

The presence of an intruder with motivation and purpose means that a cybersecurity failure is generally not a random event. Therefore, as a rule, statistical approaches are inapplicable to failure analysis whereas a risk-oriented approach and logical rules can be used.

If the probabilistic nature of the intruder's impact on a system is allowed, the impact of a failure on the entire system and the associated risks (see block 2 in Fig. 2) are assessed using FMEA and the theoretical and probabilistic approaches developed in [18, 44].

The approach described above can be combined into an integrated protection and vulnerability assessment method, called Vulnerability Inspection Control Strategy (VICS); see the flowchart in Fig. 3.

Consider the sequence of actions to analyze vulnerabilities and assess patches for a modern ICS [45] within VICS. The initial data for the analysis are:

– the list of vulnerabilities analyzed;

– an accepted classification system for threats acting on the system (e.g., the CIA model or the FSTEC classifier);

– an accepted intruder's model;

– accepted typical attack scenarios, which allow determining the potential type of a component failure for each "threat class–intruder competence" pair;

– the list of system functions, including known negative consequences of violating them;

– a "function failure–barrier" table, which describes a bigraph defining a barrier for each failure type according to the protection measures implemented;

– the structural diagram of the system, which serves to relate a vulnerability to one or more system components;

– a fault tree, which is intended to trace the impact of a failure of compromised components on system functions and group them by type (see *Fault Tree Analysis* (FTA)).

The analysis comes to the following actions:

1. For each vulnerability, assign a threat class, a typical attack scenario, and a component failure type, considering the intruder's competence level and motivation.

2. Analyze the availability of a protection barrier. If the barrier exists, stop the analysis for this vulnerability.

3. If the barrier is absent or insufficient for this type of attack, select a protection measure or apply a patch that neutralizes the security threats identified or mitigates the risk of their exploitation to an acceptable level, following an accepted patch management methodology (e.g., see the guidelines [9]). If the decision is to implement a new protection measure, add the corresponding attack scenario and barrier into the table describing the bigraph.

Thus, a security update is applied as a vulnerability risk mitigation measure under the following conditions (criteria):

– no barrier against a given attack type;

– an insufficient barrier against a given type of attack;

– the absence of protection measures that are more effective than installing an update.

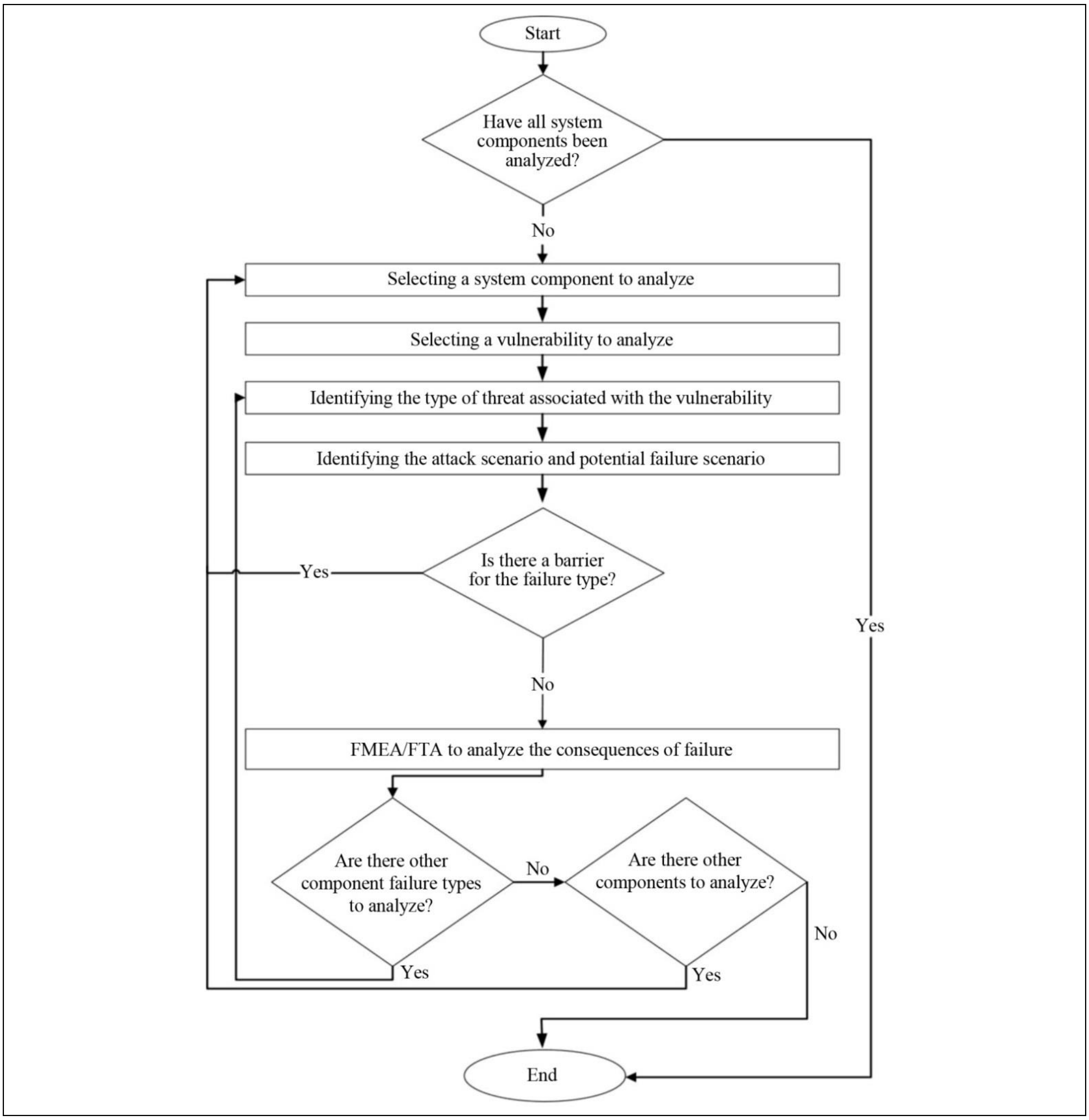
Start
Have all system components been analyzed?
No
Yes
Selecting a system component to analyze
Selecting a vulnerability to analyze
Identifying the type of threat associated with the vulnerability
Identifying the attack scenario and potential failure scenario
Is there a barrier for the failure type?
Yes
No
FMEA/FTA to analyze the consequences of failure
Are there other component failure types to analyze?
No
Are there other components to analyze?
Yes
Yes
No
End

Fig. 3. The flowchart of Vulnerability and Inspection Control Strategy.

We propose the following sequence of actions for information system vulnerabilities:

1. During the development of a protection system, perform a threat analysis and create a catalog of protection barriers and a catalog of typical attack scenarios.

2. Based on the two catalogs, form a bigraph describing the parrying of attacks by protection barriers.

3. Classify each new vulnerability in accordance with the classes of threats, intruders, and attack scenarios (see Section 4). If a vulnerability is assigned a class with an available barrier (a set of protection measures), no patch is required.

4. If a vulnerability leads to a new attack scenario, either install a patch or implement other risk mitigation measures. In particular, they may include the introduction of new protection barriers minimizing the impact of this attack scenario.

5. Analyze periodically the correctness of the protection measures that form barriers. If protection

measures are considered insufficient based on the analysis results, other or additional ones (including patch installation) must be developed and implemented.

Note that VICS does not cover vulnerabilities in software protection barriers and the correctness of the operating environment. However, the systems under consideration are usually built with high confidence in the correctness of protection measures implemented and the quality of system operation; as a result, the number of vulnerabilities in the barriers is usually much smaller than the total number of vulnerabilities in the system. Therefore, the method covers the vast majority of vulnerabilities. Existing guidelines can be used to manage vulnerabilities in protection barriers (e.g., see [3, 8, 9]).

In the next section, we demonstrate the practical application of VICS on the example of building a protection system for an upper-level system of an instrumentation and control system (ULS ICS).

4. PRACTICAL APPLICATION OF THE METHOD

Let us demonstrate the practical application of VICS on the example of ULS ICS. Here is a brief description of its main functions and properties. (For a detailed consideration of such systems, we refer, e.g., to [45].)

The upper-level system of an ICS is intended:

– to implement information, control, and auxiliary functions;

– to send operator's control commands for industrial processes and equipment;

– to monitor the ICS state;

– to integrate information from ICS subsystems.

The ULS ICS under consideration consists of servers, which collect and archive information from related systems and operator's commands, and workstations, where information about the ICS state is displayed and control commands are entered. All ULS elements are redundant and connected by a redundant network. The ICS has no access to the Internet. Unidirectional data flows from the ICS to the outside (e.g., via a data diode) are allowed.

We adopt the CIA model to describe threats, associating with it the main threat types of violating confidentiality, integrity, and availability. This approach is undoubtedly a simplification, but it follows the practice reflected in many widespread security classifiers (e.g., CWE and CVSS). We take the intruder's model with a low or medium level of privileges (no administrator rights) and medium capabilities to realize information security threats (intruder's competence level H3 according to FSTEC). Suppose that the intruder undertakes a local attack to violate the integrity of software or data (including an unauthorized execution of commands) and implement thereby a control command that will cause physical damage to the industrial facility.

Let a set of protection measures (a barrier) be included in the ICS during the design stage to prevent access of an unprivileged user with a medium level of competence in the system software. For the ICS, the protection measures can be selected from the list [46]. Assume also that the system is not compromised at the time of the attack. For the attack scenario and barrier, a fragment of the bigraph is shown in Fig. 4.

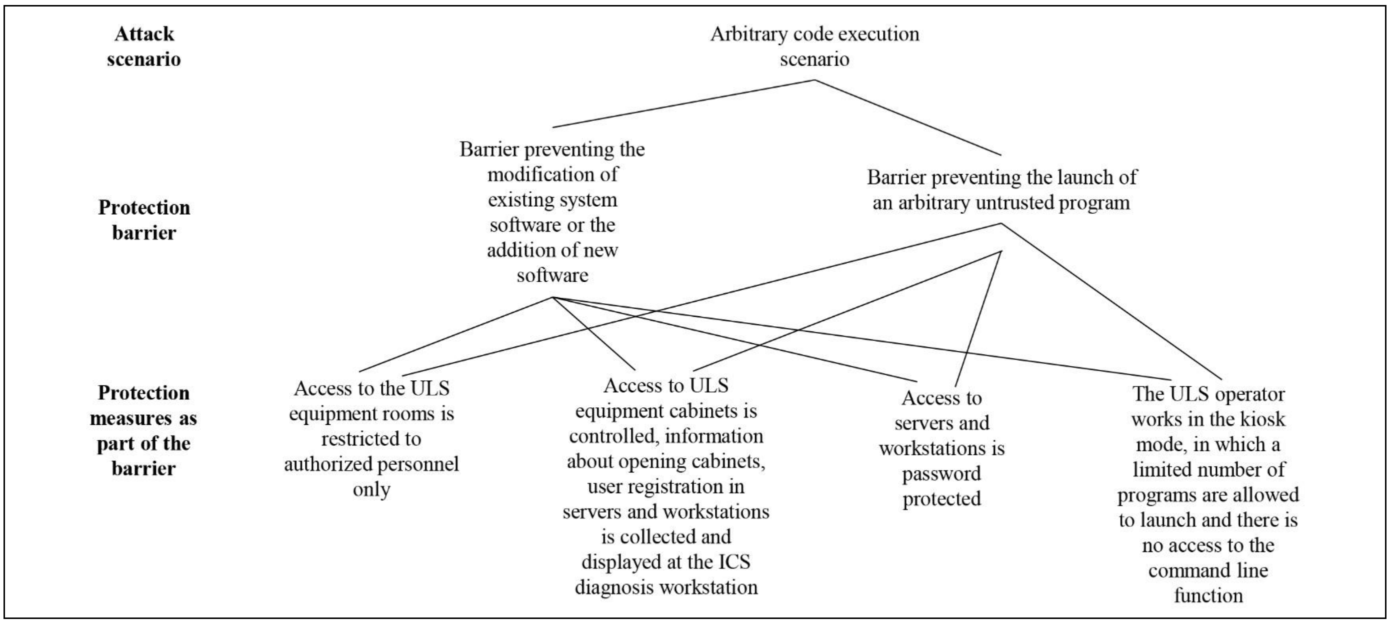
Attack scenario
Arbitrary code execution scenario
Protection barrier
Barrier preventing the modification of existing system software or the addition of new software
Barrier preventing the launch of an arbitrary untrusted program
Protection measures as part of the barrier
Access to the ULS equipment rooms is restricted to authorized personnel only
Access to ULS equipment cabinets is controlled, information about opening cabinets, user registration in servers and workstations is collected and displayed at the ICS diagnosis workstation
Access to servers and workstations is password protected
The ULS operator works in the kiosk mode, in which a limited number of programs are allowed to launch and there is no access to the command line function

Fig. 4. The bigraph describing the "attack scenario–protection barrier" relationship.

Now we pass to the vulnerability analysis. From the vulnerability database it is necessary to unload the descriptions of vulnerabilities related to the software used in the ULS ICS, considering the versions of all components. To ease the work, one can use a vulnerability scanner.

For the sake of definiteness, let the ULS ICS be implemented in Linux. Consider a vulnerability in the glibc base component (CVE-2020-1752), which has a high degree of danger according to the CVSS 3.0 classifier. Exploiting this vulnerability, a local intruder can execute an arbitrary code by passing a special file path to a program and thus violate software integrity. All these properties of the vulnerability are reflected in the CVSS 3.0 vector.

In the accepted security model (see the table), such a vulnerability corresponds to an attack scenario allowing the execution of an arbitrary code without privilege escalation; also, a barrier acts against this attack to prevent vulnerability exploitation by an intruder in the given attack scenario.

Thus, in view of high confidence in the barrier's capability to counteract uncontrolled access to system software, we consider the risk of vulnerability minimal and no patches need to be installed for the CVE-2020-1752 vulnerability.

Obviously, for vulnerabilities with a similar attack scenario, VICS will yield the same results.

CONCLUSIONS

The problems of installing security updates are quite acute due to the increasing focus on cybersecurity in all computer applications. Recommendations on patching have been developed over the years [3, 8, 9], but their use encounters difficulties in practice.

The first difficulty, which concerns information systems of any purpose, is the effect of scale. Currently, thousands of vulnerabilities are discovered every year, making it inefficient to analyze each vulnerability "manually." The difficulties of analyzing each vulnerability separately can lead to a "patch everything" strategy. However, this strategy is fraught with compatibility validation problems for software and hardware components and disruption of continuous business processes under protection.

Instrumentation and control systems (ICSs) are additionally characterized by a long lifecycle (decades), low variability of hardware during operation, a rather lengthy development stage, and (often) the need for software certification. Therefore, even if all known vulnerabilities are eliminated in the ICS software at the time of its acceptance for operation, by the time of complete commissioning the software will certainly contain newly discovered vulnerabilities that cannot be promptly eliminated. Due to invariable hardware, after several years of operation, many software components will be impossible to update without violating the hardware-software compatibility of the system, as new versions of third-party software may no longer support obsolete hardware. Industry regulations may require recertification after updates have been installed, resulting in additional time and costs. The need for recertification after each software change is essential and is recognized not only by developers but also by regulating authorities. In particular, at the international level, the possibility of classifying changes (patches) according to the degree of their impact on the functionality of software and, depending on this, changing the requirements for recertification is being considered.

Obviously, some method is needed to analyze a large number of vulnerabilities for a particular system and provide recommendations on patching without a full and detailed analysis of each vulnerability.

This paper has proposed an FMEA-based approach considering not a separate vulnerability but its impact as part of an attack scenario (within an accepted model of threats and intruder) on the functions of the entire system with existing barriers (sets of protection measures effective against a definite attack scenario).

In accordance with secure design principles, definite sets of protection measures are embedded into the system to form guaranteed barriers for definite classes of vulnerabilities. These barriers are activated during the operation stage.

The "attack scenario– barrier" relationships form a bigraph, and risk analysis in case of new vulnerabilities is reduced to analyzing this bigraph.

Suppose that a new vulnerability is discovered, leading to an attack scenario without an installed (or ineffective) protection barrier. In this case, the new risk source requires additional measures in the form of installing a security update or applying other compensating measures. This is the decision criterion for installing security updates.

Barriers are considered in the context of protection against classes of vulnerabilities rather than a separate vulnerability. Therefore, the approach not only counteracts open (known) vulnerabilities but also provides protection against yet-to-be-discovered vulnerabilities, which are always present in complex software products.

Note finally that in a complex system, manual vulnerability analysis, even with a small number of attack scenarios, types of intruders and threats, is extremely difficult and makes a mass problem. As we believe, automation of vulnerability analysis using formal problem-oriented languages for describing vulnerability and attack scenarios will make this approach applicable to real control systems, and research in this direction is ongoing.

REFERENCES

1. *IEC TR 62443-2-3. Technical Report. Security for Industrial Automation and Control Systems. Part 2-3: Patch Management in the IACS Environment,* Geneva: International Electrotechnical Commission, 2015.
2. *The Methodology for Testing Security Updates of Software and Programmable Digital Items*. Approved by the Federal Service for Technical and Export Control (FSTEC) of Russia on October 28, 2022. (In Russian.)
3. Souppaya, M. and Scarfone, K., Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology, *Special Publication (NIST SP) no. 800-40r4*, Gaithersburg, MD: National Institute of Standards and Technology, 2022. DOI: 10.6028/NIST.SP.800-40r4
4. National Information Assurance (IA) Glossary, *CNSS Instruction no. 4009*, Fort Meade: Committee on National Security Systems Instruction, 2015.
5. *CVE*. URL: https://cve.mitre.org. (Accessed October 2, 2024.)
6. *The Data Bank of Information Security Threats*. URL: https://bdu.fstec.ru/vul. (Accessed October 2, 2024.) (In Russian.)
7. *CERT-FR avis*. URL: https://www.cert.ssi.gouv.fr/avis/. (Accessed February 2, 2024.)
8. *ISO/IEC TS 9569:2023. Technical Specification. Information Security, Cybersecurity and Privacy Protection – Evaluation Criteria for IT Security – Patch Management Extension for the ISO/IEC 15408 series and ISO/IEC 18045*, Geneva: International Standard Organization/International Electrotechnical Commission, 2023.
9. *Vulnerability Management Guidelines for an Authority (Organization)*. Approved by the Federal Service for Technical and Export Control (FSTEC) of Russia on May 17, 2023. (In Russian.)
10. *The Methodology for Assessing the Criticality of Vulnerabilities in Software and Programmable Digital Items*. Approved by the Federal Service for Technical and Export Control (FSTEC) of Russia on October 28, 2022. (In Russian.)
11. Scarfone, K., Souppaya, M., and Dodson, D., Secure Software Development Framework (SSDF). Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities, *Special Publication (NIST SP) no. 800-218*, Gaithersburg, MD: National Institute of Standards and Technology, 2022. DOI: 10.6028/NIST.SP.800-218
12. *The Methodology for Assessing Information Security Threats*. Approved by the Federal Service for Technical and Export Control (FSTEC) of Russia on February 5, 2021. (In Russian.)
13. *IEC/TS 62443-1-1:2009. Industrial Communication Networks. – Network and System Security. – Part 1-1: Terminology, Concepts and Models (IDT)*, Geneva: International Electrotechnical Commission, 2009.
14. *GOST* (State Standard) *R 56939-2016: Information Security. Development of Secure Software. General Requirements*, Moscow: Standartinform, 2016.
15. *GOST* (State Standard): *The Unified System of Program Documentation. The Types of Programs and Program Documents*, Moscow: Standartinform, 2010. (In Russian.)
16. Promyslov, V.G. and Zharko, E.F., Approaches to Risk Assessment in Cybersecurity of A-plant Process Control Systems, *Automation in Industry*, 2022, no. 11, pp. 28–33. (In Russian.)
17. Dissanayake, N., Jayatilaka, A., Zahedi, M., and AliBabar, M., Software Security Patch Management - A Systematic Literature Review of Challenges, Approaches, Tools and Practices, *Information and Software Technology*, 2022, vol. 144, art. no. 106771.
18. *IEC 60812:2006. Analysis Techniques for System Reliability – Procedure for Failure Mode and Effects Analysis (FMEA)*, Geneva: International Electrotechnical Commission, 2006.
19. *About CWE*. URL: https://cwe.mitre.org/about/index.html. (Accessed April 22, 2024.)
20. *GOST* (State Standard) *R ISO/MEK 27005-2010*: *Information Technology. Methods and Means of Ensuring Security. Risk Management in Information Security*. Moscow: Standartinform, 2011. (In Russian.)
21. *GOST* (State Standard) *R ISO/MEK 15408-3-2008: Information Technology. Methods and Means of Ensuring Security. Criteria for Assessing the Security of Information Technology. Part 3. Security Confidence Components*, Moscow: Standartinform, 2008. (In Russian.)
22. *CVE Metrics*. URL: https://www.cve.org/About/Metrics. (Accessed April 22, 2024.)
23. Haddad, A., Aaraj, N., Nakov P., et al., Automated Mapping of CVE Vulnerability Records to MITRE CWE Weaknesses, *arXiv:2304.11130*, 2023. DOI: 10.48550/arXiv.2304.11130
24. Lin, Y.-Z., Mamun, M., Chowdhury, V.A., et al., HW-V2W-Map: Hardware Vulnerability to Weakness Mapping Framework for Root Cause Analysis with GPT-assisted Mitigation Suggestion, *arXiv:2312.13530*, 2023. DOI: 10.48550/arXiv.2312.13530
25. Mell, P., Scarfone, K., and Romanosky, S., A Complete Guide to the Common Vulnerability Scoring System Version 2.0, 2007. URL: https://www.first.org/cvss/v2/guide. (Accessed April 1, 2024.)
26. *Common Vulnerability Scoring System v3.0: Specification Document*, URL: https://www.first.org/cvss/v3.0/specification-document. (Accessed September 22, 2024.)
27. *CVSS V3 Calculator*. URL: https://bdu.fstec.ru/calc3. (Accessed September 22, 2024). (In Russian.)
28. Kekül, H., Ergen, B., and Arslan, H., Comparison and Analysis of Software Vulnerability Databases, *Int. J. Eng. Manuf.*, 2022, vol. 12, no. 4, pp. 1–14.
29. *How We Assess Acceptance Levels*. National Vulnerability Database. URL: https://nvd.nist.gov/vuln/cvmap/How-We-Assess-Acceptance-Levels. (Accessed April 26, 2024.)
30. *CAPEC List*. URL: https://capec.mitre.org/data/index.html. (Accessed April 22, 2024.)
31. *MITRE ATT&CK*. URL: https://attack.mitre.org/. (Accessed April 22, 2024.)
32. *Microsoft Threat Modeling Tool*. URL: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats. (Accessed May 31, 2024.)
33. Pietre-Cambacedes, L. and Chaudet, C., Disentangling the Relations between Safety and Security, *Proceedings of the 9th WSEAS International Conference on Applied Informatics and Communications*, Stevens Point, Wisconsin, 2009, pp. 156–161.
34. Boehs, E., Everything I Know About the XZ Backdoor. URL: https://boehs.org/node/everything-i-know-about-the-xz-back door. (Accessed April 10, 2024.)
35. *CVE-2022-23812*. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23812. (Accessed April 10, 2024.)

36.Kapranov, O. and Gureeva, Yu., Employees of the Ministry of Education Have Been Prohibited from Using Apple Devices, *Rossiiskaya Gazeta*, July 7, 2023. URL: https://rg.ru/2023/07/19/sotrudnikam-minprosveshcheniia-zapretili-polzovatsia-tehnikoj-apple.html. (Accessed October 22, 2024; in Russian.)

37.*Commerce Department Prohibits Russian Kaspersky Software for U.S. Customers*, Bureau of Industry and Security, June 20, 2024. URL: https://www.bis.gov/press-release/commerce-department-prohibits-russian-kaspersky-software-us-customers. (Accessed December 20, 2024.)

38.Semenkov, K., Promyslov, V., Poletykin, A., et al., Validation of Complex Control Systems with Heterogeneous Digital Models in Industry 4.0 Framework, *Machines*, 2021, vol. 9, no. 3, art. no. 62.

39.*GOST* (State Standard) *ISO/IEC TS 19249-2021: Information Technologies. Methods and Means of Ensuring Security. The Catalog of Architecture and Design Principles for Secure Products, Systems, and Applications*, Moscow: Standartinform, 2021. (In Russian.)

40.*MIL-P 1629: USA Military Standard, Procedure for Performing a Failure Mode, Effects and Criticality Analysis*, Washington, DC: Department of Defense, 1980.

41.Schmittner, C., Gruber, T., Puschner, P., and Schoitsch, E., Security Application of Failure Mode and Effect Analysis (FMEA), in *Lecture Notes in Computer Science*, 2014, vol. 8666, pp. 310–325. DOI: https://doi.org/10.1007/978-3-319-10506-2_21

42. Talwar, P. Software Failure Mode and Effects Analysis, Advances in Intelligent Systems and Computing, 2020, vol. 1131, pp. 86–91.

43. Busquim e Silva, R.A., Piqueira, J.R.C., Cruz, J.J., Marques R.P. Cybersecurity Assessment Framework for Digital Interface Between Safety and Security at Nuclear Power Plants, International Journal of Critical Infrastructure Protection, 2021, vol. 34, art. no. 100453. DOI: 10.1016/j.ijcip.2021.100453

44. Kalashnikov, A.O., Bugajskij, K.A., Birin, D.S., et al., Application of the Logical-Probabilistic Method in Information Security (Part 1), *Cybersecurity Issues*, 2023, no. 4 (56), pp. 23–32. (In Russian.)

45. Mengazetdinov, N.E., Poletykin, A.G., Promyslov, V.G., et al., *Kompleks rabot po sozdaniyu pervoi upravlyayushchei sistemy verkhnego blochnogo urovnya ASU TP dlya AES “Busher” na osnove otechestvennykh informatsionnykh tekhnologii* (The Complex of Works on Creating the First Control System of the Upper Block Level of the ICS for the Bushehr NPP Based on Russian Information Technology), Moscow: Trapeznikov Institute of Control Sciences RAS, 2013. (In Russian.)

46.*Requirements for Ensuring the Security of Significant Critical Information Infrastructure Objects of the Russian Federation*. Approved by the Federal Service for Technical and Export Control (FSTEC) of Russia on December 25, 2017. (In Russian.)

This paper was recommended for publication by R. V. Meshcheryakov, a member of the Editorial Board.

Received June 6, 2024, and revised March 18, 2025.
Accepted April 28, 2025

Author information

Semenkov, Kirill Valer’evich. Cand. Sci. (Phys.–Math.), Trapeznikov Institute of Control Sciences, Russian Academy of Science, Moscow, Russia
✉ semenkov@ipu.ru
ORCID iD: https://orcid.org/0000-0003-0865-9072

Promyslov, Vitaly Georgievich. Cand. Sci. (Phys.–Math.), Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ vp@ipu.ru
ORCID iD: https://orcid.org/0000-0003-1919-8718

Cite this paper

Semenkov, K.V., and Promyslov, V.G., A Procedure for Assessing Security Updates in Industrial Systems. *Control Sciences* **2,** 49–61 (2025).

Original Russian Text © Semenkov, K.V., Promyslov, V.G., 2025, published in *Problemy Upravleniya*, 2025, no. 2, pp. 58–73.

Translated into English by *Alexander Yu. Mazurov*,
Cand. Sci. (Phys.–Math.),
Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ alexander.mazurov08@gmail.com