Concept of Cyber Immunity of Industry 4.0

Sergei A. Petrenko
Department of Information Security
Saint Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
S.Petrenko@rambler.ru

Krystina A. Makoveichuk
Department of Informatics and Information Technologies
V.I. Vernadsky Crimean Federal University
Yalta, Russia
Christin2003@yandex.ru

Alexander V. Olifirov
Department of Economics and Finance
V.I. Vernadsky Crimean Federal University
Yalta, Russia
Alex.Olifirov@gmail.com

***Abstract* — The article presents the development of the concept of cyber immunity to protect the Industry 4.0 critical information infrastructure and the theory of self-healing machine computing. The specified theory is based on the results of the scientific-applied sections of biological and cybernetic immunology. In the developed concept it was taken into account that the neutralization of malicious influences should not lead to a denial of service for the entire system and to a loss of the functional semantics of calculations. Using interrelated probative, verification and testing programming, a model for restoring functional program specifications was developed. It was developed with separation by levels: semantic (functional, logical and algebraic models are defined to determine the base of functionally-logic specifications of programs); syntactic (defined models to form automatic machines for the detection and neutralization of malicious influences); semantically syntactic (models are defined for applying the simplest forms of program calculation semantics based on graphical, schematic, and network representations). In order to prove the correctness of functional semantics of the "cleared" calculations, a mathematical apparatus of the similarity theory and calculation dimensions were developed. The π-converter was identified; this operator allows forming the required "passports" of trusted computations in the conditions of disturbances. Calculations with "antibodies" are represented by regular schemes in the system of algorithmic algebras of V. M. Glushkov. A semantically controlled translator based on formal automata with abstract memory was developed for the interpretation of the input program of the trusted computations and type of actions.**

Keywords — cyber immunity; Industry 4.0; self-healing; vulnerabilities; antigens; antibodies; destructive code; neutralization of malicious; critical infrastructure; functional semantics; semantic-syntactic models.

I. Introduction

A fundamental contribution to the formation and development of the theoretical and system programming was made by the outstanding scientists from all over the world: A. Turing, J. Von Neumann, M. Minsky, A. Church, S. Klini, D. Scott, Z. Manna, E. Dijkstra, Ch. Hoare, J. Backus, N. Wirth, D. Knut, N. Khomsky, A. Kolmogorov, A. Ershov, V. Glushkov, A. Markov and others. They had laid the foundations of the mentioned programming, allowing mathematically strict studying the possible computational structures, studying computability properties and modeling computational abstractions of executable actions. These results produced the leading scientific schools, which made a significant contribution to the development of the synthesis methods of model abstractions and specific software solutions. Including Russian scientific schools that have contributed:

- Study of abstract data types and denotation semantics (Y. L. Ershov, Y. V. Sazonov);
- Automatic algebraic synthesis of programs (V. M. Glushkov, E. L. Yushchenko);
- Conceptual programming (E. H. Tyugu, G. E. Minz);
- Automatic synthesis of programs, based on knowledge (D. A. Pospelov);
- Development of a logical and applicative approach to functional programming (W. E. Wolfengagen);
- Methodology of applied verification and testing of programs (V. A. Nepomniashchy, O. M. Ryakin, Y. V. Borzov);
- Methodology of symbolic modeling and intellectual gyromates of cyber security (Y. G. Rostovtsev, A. G. Lomako, D. N. Biriukov).

However, in order to protect the critical information infrastructure of the Industry 4.0 in the face of increasing threats to information security, it was necessary to define the concept of cyber immunity and develop a new theory of self-healing machine computing [13, 14]. This new theory was enriched by the results of the scientific-applied sections of biological (E.

Metchnikoff) (Figure 1, Figure 2), and cybernetic immunology (A. Tarakanov, D. Hunt, D. Dasgupta, P. Andyus).

II. The Basis of Theory of Self-Healing Machine Computing

In the mentioned theory, by analogy with classical immunology (Figure 1, Figure 2), the antigen is understood to be some destructive program code, and the antibody is a synthesized metaprogram of this code neutralization. Model of immune protection of Industry 4.0 describes the causal relationship between "antigens" and "antibodies". That is, between vulnerabilities and program defects (manifested in the form of structural violations), modes of functioning (distorting the properties of programs), security incidents, caused by the destructive program tabs (changing the given standard algorithms of calculations) and metaprograms for their neutralization.

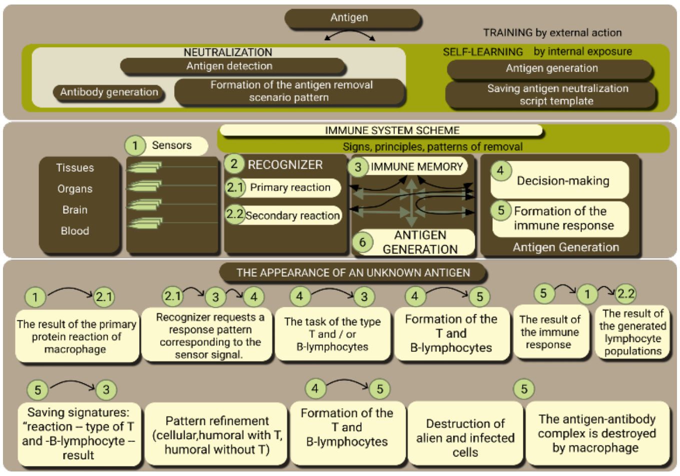
Antigen
TRAINING by external action
NEUTRALIZATION
Antigen detection
SELF-LEARNING by internal exposure
Antigen generation
Antibody generation
Formation of the antigen removal scenario pattern
Saving antigen neutralization script template
IMMUNE SYSTEM SCHEME
Signs, principles, patterns of removal
1 Sensors
Tissues
Organs
Brain
Blood
2 RECOGNIZER
2.1 Primary reaction
2.2 Secondary reaction
3 IMMUNE MEMORY
6 ANTIGEN GENERATION
4 Decision-making
5 Formation of the immune response
Antigen Generation
THE APPEARANCE OF AN UNKNOWN ANTIGEN
1 2.1
The result of the primary protein reaction of macrophage
2.1 3 3
Recognizer requests a response pattern corresponding to the sensor signal.
4 3
The task of the type T and / or B-lymphocytes
4 5
Formation of the T and B-lymphocytes
5 1
The result of the immune response
2.2
The result of the generated lymphocyte populations
5 3
Saving signatures: "reaction – type of T and -B-lymphocyte – result
Pattern refinement (cellular,humoral with T, humoral without T)
4 5
Formation of the T and B-lymphocytes
Destruction of alien and infected cells
5
The antigen-antibody complex is destroyed by macrophage

Fig. 1. Structural and functional diagram of the biological system

Immune protection of Industry 4.0 includes three key subsystems: Recognizer, Planner and Executor. Here, the Recognizer is designed to recognize patterns (images) of malicious code by its structural, correlation and invariants features. The scheduler is intended for planning, i.e. creation of corresponding plans and metaprograms of malicious code neutralization. The executor is intended for execution of the specified plans and metaprograms. As a result of these three subsystems operation, the required "purification" and formation of a trusted environment for calculations in the conditions of heterogeneous mass cyber-attacks by malefactors takes place.

Whereas the classical immunology neutralizes antigens by physically destroying them (absorbing them), this is unacceptable in cybernetic immunology. As far as the loss of a part of the functional program code can lead to denial of service and impossibility to continue calculations as a whole. That is why it was demanded that the functional semantics of calculations during the neutralization of malicious influences be invariable (constant) [1-7, 13-18]. Moreover, the critical information infrastructure of Industry 4.0 must be able to recover from both known and previously unknown attackers. Including under conditions of a priori uncertainty and obfuscation of programs (Figure 3).

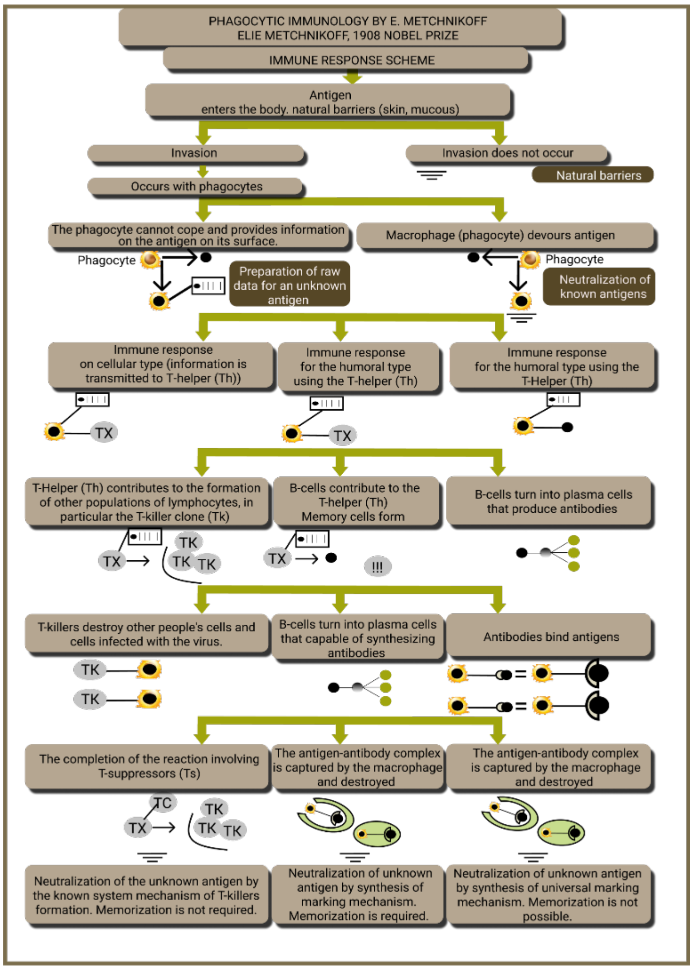
PHAGOCYTIC IMMUNOLOGY BY E. METCHNIKOFF
ELIE METCHNIKOFF, 1908 NOBEL PRIZE
IMMUNE RESPONSE SCHEME
Antigen
enters the body. natural barriers (skin, mucous)
Invasion
Invasion does not occur
Natural barriers
Occurs with phagocytes
The phagocyte cannot cope and provides information on the antigen on its surface.
Macrophage (phagocyte) devours antigen
Phagocyte
Preparation of raw data for an unknown antigen
Phagocyte
Neutralization of known antigens
Immune response on cellular type (information is transmitted to T-helper (Th))
Immune response for the humoral type using the T-helper (Th)
Immune response for the humoral type using the T-Helper (Th)
TX
TX
T-Helper (Th) contributes to the formation of other populations of lymphocytes, in particular the T-killer clone (Tk)
B-cells contribute to the T-helper (Th) Memory cells form
B-cells turn into plasma cells that produce antibodies
TX
TK
TK
TK
TX
!!!
T-killers destroy other people's cells and cells infected with the virus.
B-cells turn into plasma cells that capable of synthesizing antibodies
Antibodies bind antigens
TK
TK
The completion of the reaction involving T-suppressors (Ts)
The antigen-antibody complex is captured by the macrophage and destroyed
The antigen-antibody complex is captured by the macrophage and destroyed
TC
TX
TK
TK
TK
Neutralization of the unknown antigen by the known system mechanism of T-killers formation. Memorization is not required.
Neutralization of unknown antigen by synthesis of marking mechanism. Memorization is required.
Neutralization of unknown antigen by synthesis of universal marking mechanism. Memorization is not possible.

Fig. 2. Phagocytic theory of E. Metchnikoff

Taking into account the requirements set forth above, we present the main goals of the organization of self-recovering trusted computations C_1, C_2 and C_3 by the following display $F_0 \div F_6$ system (Figure 4).

In order to reach these goals, a number of research objectives were achieved. In particular, the model of restoration of functional program specifications in the ideology of interrelated probative, verification and testing programming has been developed (Figure 4). Here, the probative programming allowed us to study the correctness of computational structures, correctness of computability properties and stability of calculations. These aspects were modelled by denotation, axiomatic and operational formal semantics of programs, respectively.

Thus for an establishment of conformity between their functionally-logic specifications and physical design the methods of annotated programs of N. Wirth, Ch. Hoare and E. Dixtra were involved.

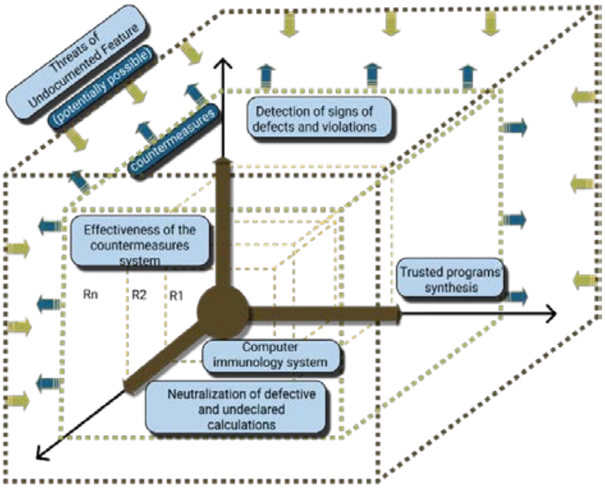
Threats of Undocumented Feature (potentially possible)
countermeasures
Detection of signs of defects and violations
Effectiveness of the countermeasures system
Trusted programs synthesis
Rn
R2
R1
Computer immunology system
Neutralization of defective and undeclared calculations

Fig. 3. State space of the system with "vaccinated" cyber immunity

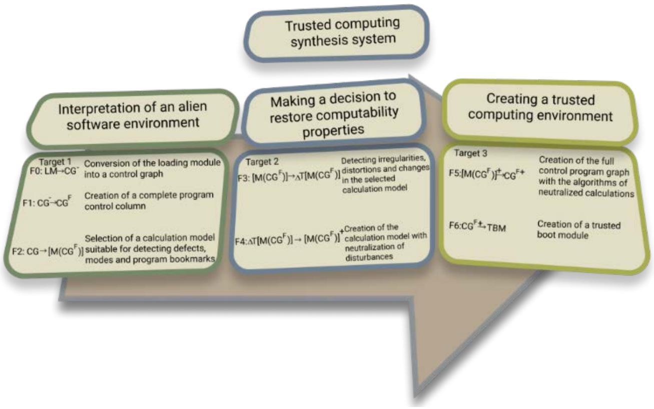
Trusted computing synthesis system
Interpretation of an alien software environment
Making a decision to restore computability properties
Creating a trusted computing environment
Target 1
F0: LM→CG⁻
Conversion of the loading module into a control graph
F1: CG⁻→CG^F
Creation of a complete program control column
F2: CG→[M(CG^F)]
Selection of a calculation model suitable for detecting defects, modes and program bookmarks
Target 2
F3: [M(CG^F)]→ΔT[M(CG^F)]
Detecting irregularities, distortions and changes in the selected calculation model
F4:ΔT[M(CG^F)]→[M(CG^F)]^+
Creation of the calculation model with neutralization of disturbances
Target 3
F5:[M(CG^F)]^+→CG^F+
Creation of the full control program graph with the algorithms of neutralized calculations
F6:CG^F+→TBM
Creation of a trusted boot module

Fig. 4. Goals and objectives of building a trusted synthesis system

In order to solve the specific problems of restoring the functional specifications of programs, a corresponding model basis was developed (Figure 5) [1, 14].

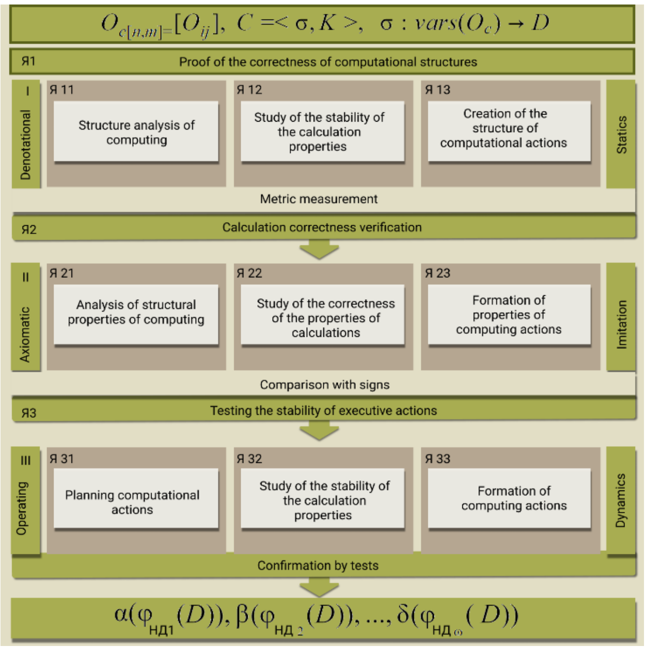
$O_{c[n,m]}=[O_{ij}],\ C=<\sigma,K>,\ \sigma: vars(O_c)\to D$
Я1
Proof of the correctness of computational structures
I
Denotational
Я 11
Structure analysis of computing
Я 12
Study of the stability of the calculation properties
Я 13
Creation of the structure of computational actions
Statics
Metric measurement
Я2
Calculation correctness verification
II
Axiomatic
Я 21
Analysis of structural properties of computing
Я 22
Study of the correctness of the properties of calculations
Я 23
Formation of properties of computing actions
Imitation
Comparison with signs
Я3
Testing the stability of executive actions
III
Operating
Я 31
Planning computational actions
Я 32
Study of the stability of the calculation properties
Я 33
Formation of computing actions
Dynamics
Confirmation by tests
$\alpha(\varphi_{НД1}(D)), \beta(\varphi_{НД2}(D)), ..., \delta(\varphi_{НД\omega}(D))$

Fig. 5. The model of restoration of the functional program specifications

Here, the level classification was made by analogy with the classification of formal languages by N. Khomsky. In the semantic class we chose models suitable for the construction of calculations, in the syntactic class we chose models that allow us to form automatic machines for the detection and neutralization of malicious influences. The class of semantic-syntactic models has allowed operating with the simplest forms of program calculation semantics in an effective basis of models types of graphical, schematic and network representation (Figure 6). At the same time, the control flow graph (CFG), Yanov schemata and Petri nets were chosen to specify the model basis.

As a result, the following architecture of the neutralization system of malware and malicious software bookmarks was proposed (Figure 7) [3-5, 13, 14].

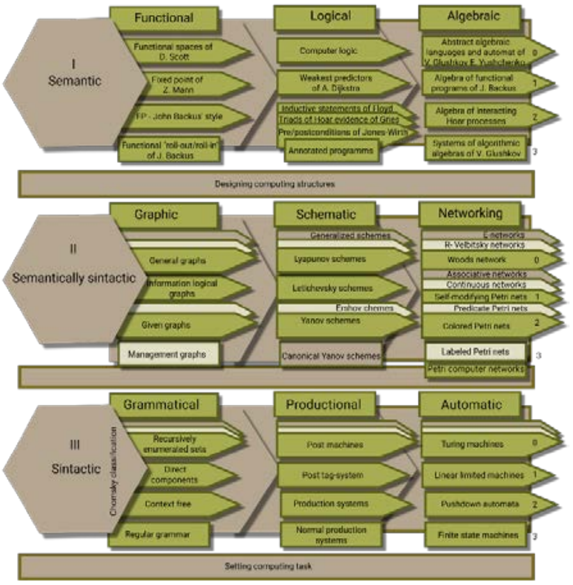

Fig. 6. Stratification of program models for calculation semantics research

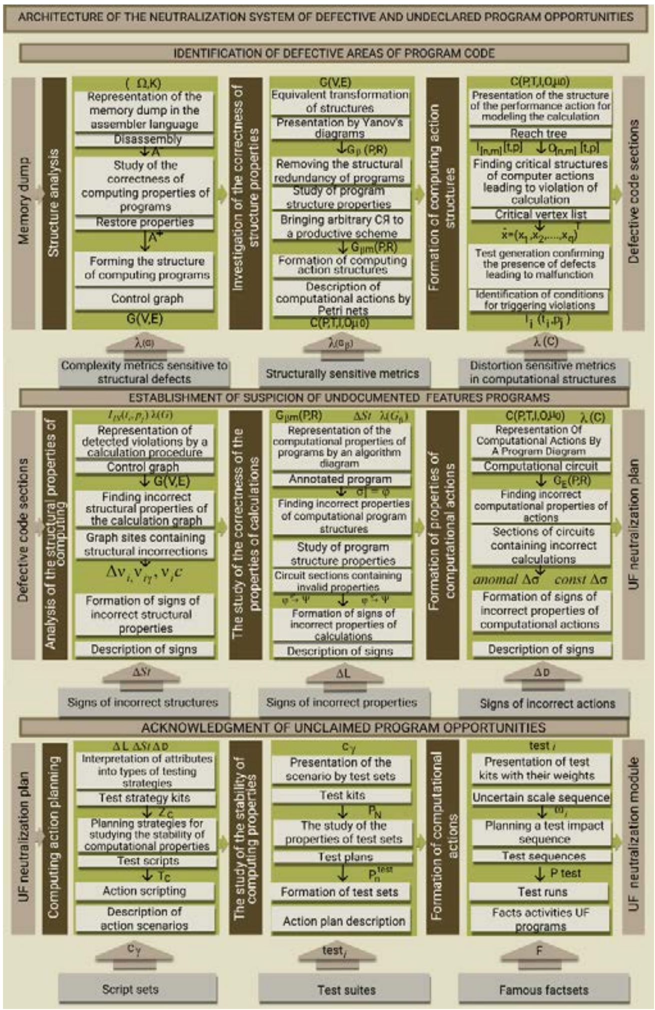
ARCHITECTURE OF THE NEUTRALIZATION SYSTEM OF DEFECTIVE AND UNDECLARED PROGRAM OPPORTUNITIES
IDENTIFICATION OF DEFECTIVE AREAS OF PROGRAM CODE
Memory dump
Structure analysis
(Ω,K)
Representation of the memory dump in the assembler language
Disassembly
A
Study of the correctness of computing properties of programs
Restore properties
A+
Forming the structure of computing programs
Control graph
G(V,E)
Investigation of the correctness of structure properties
G(V,E)
Equivalent transformation of structures
Presentation by Yanov's diagrams
Gβ (P,R)
Removing the structural redundancy of programs
Study of program structure properties
Bringing arbitrary CЯ to a productive scheme
Gμm(P,R)
Formation of computing action structures
Description of computational actions by Petri nets
C(P,T,I,O,μ0)
Formation of computing action structures
C(P,T,I,O,μ0)
Presentation of the structure of the performance action for modeling the calculation
Reach tree
I[n,m] [t,p]
O[n,m] [t,p]
Finding critical structures of computer actions leading to violation of calculation
Critical vertex list
x=(x1,x2,...,xq)
Test generation confirming the presence of defects leading to malfunction
Identification of conditions for triggering violations
Ii (ti,pj)
Defective code sections
λ(G)
λ(Gβ)
λ (C)
Complexity metrics sensitive to structural defects
Structurally sensitive metrics
Distortion sensitive metrics in computational structures
ESTABLISHMENT OF SUSPICION OF UNDOCUMENTED FEATURES PROGRAMS
Defective code sections
Analysis of the structural properties of computing
Ii(ti,pj) λ(G)
Representation of detected violations by a calculation procedure
Control graph
G(V,E)
Finding incorrect structural properties of the calculation graph
Graph sites containing structural incorrections
Δvi, vij, vic
Formation of signs of incorrect structural properties
Description of signs
The study of the correctness of the properties of calculations
Gμm(P,R)
ΔSt λ(Gβ)
Representation of the computational properties of programs by an algorithm diagram
Annotated program
σ = φ
Finding incorrect properties of computational program structures
Study of program structure properties
Circuit sections containing invalid properties
φ ↛ ψ
Formation of signs of incorrect properties of calculations
Description of signs
Formation of properties of computational actions
C(P,T,I,O,μ0) λ (C)
Representation Of Computational Actions By A Program Diagram
Computational circuit
GE(P,R)
Finding incorrect computational properties of actions
Sections of circuits containing incorrect calculations
anomal Δσ const Δσ
Formation of signs of incorrect properties of computational actions
Description of signs
UF neutralization plan
ΔSt
ΔL
ΔD
Signs of incorrect structures
Signs of incorrect properties
Signs of incorrect actions
ACKNOWLEDGMENT OF UNCLAIMED PROGRAM OPPORTUNITIES
UF neutralization plan
Computing action planning
ΔL ΔSt ΔD
Interpretation of attributes into types of testing strategies
Test strategy kits
Zc
Planning strategies for studying the stability of computational properties
Test scripts
Tc
Action scripting
Description of action scenarios
The study of the stability of computing properties
cγ
Presentation of the scenario by test sets
Test kits
PN
The study of the properties of test sets
Test plans
Pn test
Formation of test sets
Action plan description
Formation of computational actions
test i
Presentation of test kits with their weights
Uncertain scale sequence
ωi
Planning a test impact sequence
Test sequences
P test
Test runs
Facts activities UF programs
UF neutralization module
cγ
testi
F
Script sets
Test suites
Famous factsets

Fig. 7. Neutralization system architecture of the destructive software bookmarks

The mentioned architecture includes three main subsystems. The first subsystem is intended for detecting defective program code fragments. The second subsystem sets suspicion that the detected defective code sections belong to destructive program bookmarks and forms a plan (metaprogram) for their neutralization. The third subsystem confirms the presence and forms a module for neutralizing destructive software bookmarks.

In order to prove the correctness of functional semantics of the "cleared" calculations, a mathematical apparatus of the similarity theory and calculation dimensions were developed. In particular, the direct similarity theorem, which allows establishing the general scheme of representation of semantically correct calculations in the invariant (dimensionless) form, is formulated and proved

$$(D_i\left(\frac{x_{1j}}{x_{1j_0}},\frac{x_{2j}}{x_{2j_0}},\dots,\frac{x_{nj}}{x_{nj_0}},\Pi_{1i},\Pi_{2i},\dots,\Pi_{zi-1}\right)=0, \qquad (1)$$

where

$\frac{x_{1j}}{x_{1j_0}},\frac{x_{2j}}{x_{2j_0}},\dots,\frac{x_{nj}}{x_{nj_0}}$ - similarity invariants of calculations.

The direct similarity theorem allowed proving the statements about the necessary and sufficient similarity conditions of semantically correct calculations

$$\begin{cases} \frac{X_{(k+1)j}}{X_{(k+1)j_0}}=\varphi_1(\Pi_{11},\dots,\Pi_{(z_1-1)};\frac{x_{1j}}{x_{1j_0}},\dots,\frac{x_{kj}}{x_{kj_0}}) \\ \frac{X_{(k+2)j}}{X_{(k+2)j_0}}=\varphi_2(\Pi_{12},\dots,\Pi_{(z_2-1)};\frac{x_{1j}}{x_{1j_0}},\dots,\frac{x_{kj}}{x_{kj_0}}) \\ \frac{X_{nj}}{X_{nj_0}}=\varphi_m(\Pi_{1m},\dots,\Pi_{(z_m-1)};\frac{x_{1j}}{x_{1j_0}},\dots,\frac{x_{kj}}{x_{kj_0}}) \end{cases} \qquad (2)$$

where

$\Pi_{1i}=x_{1j}/C_{1j}, \Pi_{2i}=x_{2j}/C_{2j},\dots,\Pi_{zi-1}=x_{nj}/C_{nj}$ - similarity invariants,

C_{ij} - multipliers of similarity ratios transformation,

φ_i - functions of all or some relative data.

Example.

For the assignment operator $\mathrm{A} \coloneqq \mathrm{B} * \mathrm{C} + \frac{\mathrm{D}}{\mathrm{E}} + 1$, the following relations must be performed between the abstract dimensions of the parameters (*A*, *B*, *C*, *D*, *E*, *CONST_1*):

$$\begin{aligned} &(1) * ln[A] + (-1) * ln[B] + (-1) * ln[C] = 0, \\ &(1) * ln[A] + (-1) * ln[D] + (1) * ln[E] = 0, \qquad (3) \\ &(1) * ln[A]^1 + (-1) * ln[CONST_1]^1 = 0. \end{aligned}$$

The received relations allow defining unequivocally the standard (or passport) of semantically correct calculation. The calculation is semantically correct if the corresponding system of dimension equations has at least one component consisting of all non-zero components among the set of vectors-solutions.

Let us suppose that this is not the case, and among these parameters there appeared a parameter identically equal to zero at any values of other parameters. This indicates that the new parameter is dimensionless. However, it is impossible because it contradicts the initial condition of semantic correctness of calculations, which was to be proved.

Also, a п-converter was identified; this operator allows forming the required "passports" of trusted computations in the conditions of disturbances.

Statement 1: Operator F is a п-converter if for each object $O_{r_i} \in M$ and each element $g_v \in G_v$ of the finite abelian subgroup the ratio of

$$F * \left(g_v O_{r_i}\right) = F * \left(O_{r_i}\right) g_v^{-1}, i = 1, 2, \dots, m \qquad (4)$$

is true.

Proof.

Let F is п-converter and F* is corresponding mapping in the subgroup G_v.

Let us assume that the objects to be compared are equivalent.

Then

$$F\left(g_v O_{r_i}\right) = F\left(O_{r_i}\right), i = 1, 2, \dots, m \qquad (5)$$

or in terms of mapping

$$F * \left(g_v O_{r_i}\right)\left(g_v O_{r_i}\right) = F * \left(O_{r_i}\right)\left(O_{r_i}\right), i = 1, 2, \dots, m \quad (6)$$

Apply now to the left and right parts of this equality the

$$F * (O_{r_i})^{-1}, F * (O_{r_i})^{-1} F * \left(g_v O_{r_i}\right)\left(g_v O_{r_i}\right) =$$
$$= F * (O_{r_i})^{-1} F * \left(O_{r_i}\right)\left(O_{r_i}\right) = O_{r_i}, i = 1, 2, \dots, m \qquad (7)$$

based on the property of the existence of the group unit

$$F * (O_{r_i})^{-1} F * \left(g_v O_{r_i}\right) g_v = e, i = 1, 2, \dots, m \qquad (8)$$

multiplying on the left by $F * (O_{r_i})$, get

$$F * \left(g_v O_{r_i}\right) g_v = F * \left(O_{r_i}\right), i = 1, 2, \dots, m \qquad (9)$$

multiplying on the right g_v^{-1}, find the required ratio

$$F * \left(g_v O_{r_i}\right) = F * \left(O_{r_i}\right) g_v^{-1}, i = 1, 2, \dots, m \qquad (10)$$

The converse holds true.

As a result, the following conclusions can be drawn:

- п-converter is a mapping of a reference pair

$$F: M \to M_0 \qquad (11)$$

- Set of standards M_0 represents a set of objects (similarity invariants), which do not change values of their information signs under the action of п-converter F, i.e.

$$F\left(O_{r_i}\right) = O_{r_i}, \qquad (12)$$

- With the help of п-converter F and the corresponding mapping

$$F *: M \to G_v \qquad (13)$$

one can find the transformation g, which connects two equivalent objects O_{r_1} and O_{r_2}so that $g = F(O_{r_2})^{-1} F(O_{r_1})$.

It is essential that a multi-model approach was proposed solving the task of synthesizing programs of trusted computations which allowes describing abstract programs of the trusted computations in structural-functional, logical-semantic and computational-operational aspects [8-12, 14]. Such a multi-model organization of calculations required the introduction of coordination, allowing taking into account the specifics and features of each named functional model of calculations. This has led to the need to build an appropriate knowledge metamodel. As basic models in the knowledge system it was proposed to use formal grammar, production system, automatic converter.

When choosing a meta-modeling apparatus, preference was given to the system of algorithmic algebras (SAA) proposed by academician V. M. Glushkov. This made it possible to create an algorithmic system equivalent in its visual capabilities to such classical algorithmic systems as Turing machines, Post products and Markov algorithms. Besides, the advantage of SAA is the possibility to express structures of abstract programs of trusted calculations in a strict basis of Dijkstra types (sequence, branching, cycle) in the form of corresponding algebraic formulas. This allowed developing a multi-faceted algebraic system of the form $< A, L >$ with a signature of operations Δ, where A is a set of operators; L is a set of logical conditions taking values from a set of $\{\text{true, false, uncertain}\}$. Here, the signature $\Delta = \Delta_1 \cup \Delta_2$consists of a system of Δ_1 logical operations that take on a value in a variety of conditions L and a system of Δ_2 operations that take on values in a variety of operators A.

In SAA $< A, L >$ the system of forming $\coprod$ is fixed. It is the final functionally complete set of operators and logical conditions. With the help of this set and by means of superposition of operations included in Δ, arbitrary operators and logical conditions of the set A and L are generated. The logical operations of the system Δ_1 include generalized Boolean operations of disjunctions, conjunctions, and negation, as well as the operation of left multiplication of the condition by the operator $\beta = A\alpha$ and filtration. The following operations belong to the Δ_2 set: composition of operators $A * L$, sequential execution of operators A and L, α - disjunction of operators, alternative execution of operators A and L, i.e.

$$\begin{aligned} {}_{\alpha}(A \vee L) &= \grave{A}, if\ \alpha = 1; \\ {}_{\alpha}(A \vee L) &= L, if\ \alpha = 0; \qquad (14) \\ {}_{\alpha}(A \vee L) &= J, if\ \alpha = o. \end{aligned}$$

Here, the α-iteration of operator A under the condition $\alpha_{\alpha}\{A\}$ consists in checking the condition α, if this condition is false, then the execution of operator A is performed.

It should be noted that such a representation $< A, L >$ allows developing effective regularization procedures (reduction to a regular scheme *(RS))* $F(\coprod)$ and prove the theorem, which defines the principal possibility of a formal description of an arbitrary and reconstructed algorithm and procedure of trusted calculations in RS.

Thus, it is possible to formally describe the declarative, technological and procedural knowledge of trusted computations in the form of regular schemes.

Statement 2. Calculations with "antibodies" are represented by regular schemes in the system of algorithmic algebras (SAA) of V. M. Glushkov.

The modified technique of a composite programming allowed determining the effective sequence of operations of trusted calculations. For each operator's construction there were given operations and operands that make up the program of trusted calculations. After checking the completeness of this program for compliance with the selected criteria, an executable program of trusted calculations was synthesized (Figure 8).

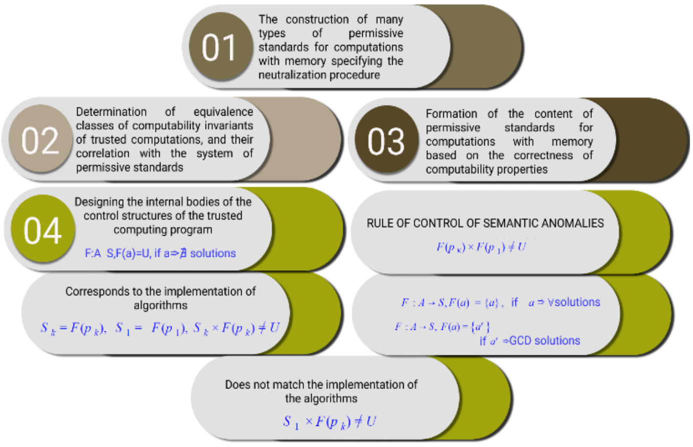
01
The construction of many types of permissive standards for computations with memory specifying the neutralization procedure
02
Determination of equivalence classes of computability invariants of trusted computations, and their correlation with the system of permissive standards
03
Formation of the content of permissive standards for computations with memory based on the correctness of computability properties
04
Designing the internal bodies of the control structures of the trusted computing program
F:A S,F(a)=U, if a⇒∄ solutions
RULE OF CONTROL OF SEMANTIC ANOMALIES
$F(p_k) \times F(p_1) \neq U$
Corresponds to the implementation of algorithms
$S_k = F(p_k),\ S_1 = F(p_1),\ S_k \times F(p_k) \neq U$
$F: A \to S, F(a) = \{a\}$, if $a \Rightarrow$ ∀solutions
$F: A \to S, F(a) = \{a'\}$
if $a' \Rightarrow$GCD solutions
Does not match the implementation of the algorithms
$S_1 \times F(p_k) \neq U$

Fig. 8. Creation of operator constructions of the trusted computing program

A semantically controlled translator based on formal automata with abstract memory (AAM) was developed for the interpretation of the input program of the trusted computations and type of actions (Figure 9). The AAM consists of four elastic belts (EB), which contain:

- Messages of functional automatons;
- Reports of identified software bookmarks;
- Neutralization and countermeasures scenarios;
- Information messages of the broadcast procedures' completion.

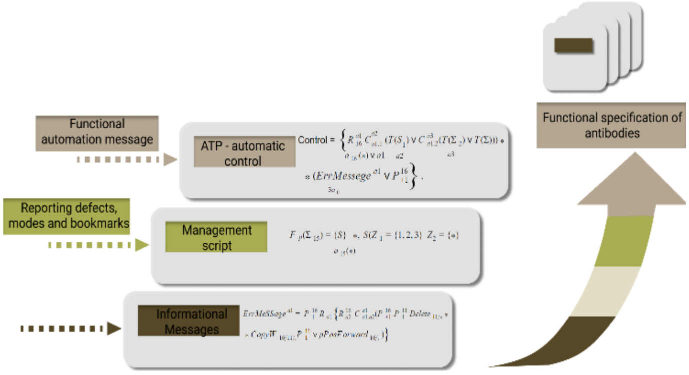
Functional automation message
ATP - automatic control
Control =
Reporting defects, modes and bookmarks
Management script
Informational Messages
Functional specification of antibodies

Fig. 9. Broadcast immunity antibody formation program

III. Conclusion

An overview of the new concept of cyber immunity of Industry 4.0. is presented in Figure 10.

The following significant results have been obtained in the course of the concept development.

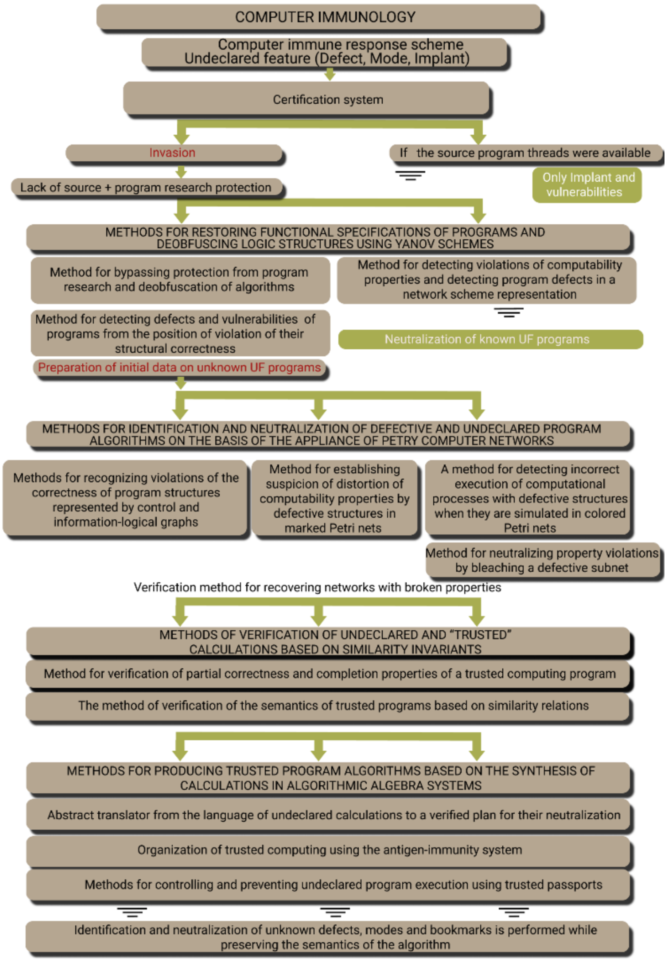
COMPUTER IMMUNOLOGY
Computer immune response scheme
Undeclared feature (Defect, Mode, Implant)
Certification system
Invasion
If the source program threads were available
Lack of source + program research protection
Only Implant and vulnerabilities
METHODS FOR RESTORING FUNCTIONAL SPECIFICATIONS OF PROGRAMS AND DEOBFUSCING LOGIC STRUCTURES USING YANOV SCHEMES
Method for bypassing protection from program research and deobfuscation of algorithms
Method for detecting violations of computability properties and detecting program defects in a network scheme representation
Method for detecting defects and vulnerabilities of programs from the position of violation of their structural correctness
Neutralization of known UF programs
Preparation of initial data on unknown UF programs
METHODS FOR IDENTIFICATION AND NEUTRALIZATION OF DEFECTIVE AND UNDECLARED PROGRAM ALGORITHMS ON THE BASIS OF THE APPLIANCE OF PETRY COMPUTER NETWORKS
Methods for recognizing violations of the correctness of program structures represented by control and information-logical graphs
Method for establishing suspicion of distortion of computability properties by defective structures in marked Petri nets
A method for detecting incorrect execution of computational processes with defective structures when they are simulated in colored Petri nets
Method for neutralizing property violations by bleaching a defective subnet
Verification method for recovering networks with broken properties
METHODS OF VERIFICATION OF UNDECLARED AND "TRUSTED" CALCULATIONS BASED ON SIMILARITY INVARIANTS
Method for verification of partial correctness and completion properties of a trusted computing program
The method of verification of the semantics of trusted programs based on similarity relations
METHODS FOR PRODUCING TRUSTED PROGRAM ALGORITHMS BASED ON THE SYNTHESIS OF CALCULATIONS IN ALGORITHMIC ALGEBRA SYSTEMS
Abstract translator from the language of undeclared calculations to a verified plan for their neutralization
Organization of trusted computing using the antigen-immunity system
Methods for controlling and preventing undeclared program execution using trusted passports
Identification and neutralization of unknown defects, modes and bookmarks is performed while preserving the semantics of the algorithm

Fig. 10. Possible immune protection methods of Industry 4.0

Theoretical results:

1. Scientific-methodological apparatus of computer immunology of cyber-security based on the mechanisms of "immune response" and "immune memory" of classical immunology.

2. Methodology of self-recovery of trusted machine calculations with the required functional semantics of calculations.

Scientific and practical results:

- Approach to deobfuscation and normalization of logical structures of calculations using a system of equivalent transformations of Janov's schemes.
- Method of combined verification of semantics of calculations on the basis of similarity invariants and provocative load testing.
- Methods of generating trusted program algorithms on the basis of synthesis of calculations in the system of algorithmic algebra and scenarios of permits.
- Computer immunology technology for cybersecurity and private methods of detecting and neutralizing destructive software bookmarks and program vulnerabilities.

REFERENCES

[1] Ashby W.R. (1991) Principles of the Self-Organizing System. In: Facets of Systems Science. International Federation for Systems Research International Series on Systems Science and Engineering, vol 7. Springer, Boston, MA. DOI: https://doi.org/10.1007/978-1-4899-0718-9_38

[2] Barabanov A.V., Markov A.S., Tsirlov V.L. Methodological Framework for Analysis and Synthesis of a Set of Secure Software Development Controls, Journal of Theoretical and Applied Information Technology, 2016, vol. 88, No 1, pp. 77-88.

[3] Biryukov D. N Cognitive-functional memory specification for simulation of purposeful behavior of cyber systems // Proceedings of SPIIRAS. - 2015. - Issue. 3 (40). - C. 55-76. DOI: http://dx.doi.org/10.15622/sp.40.5

[4] Biryukov D.N, Lomako A. G, Petrenko S. A. Generating scenarios for preventing cyber attacks // Protecting information. Inside. - 2017. - No. 4 (76). (In Russian).

[5] Biryukov D.N, Lomako A.G, Rostovtsev Yu.G. The appearance of anticipatory systems to prevent the risks of cyber threat realization // Proceedings of SPIIRAS. - 2015. - Issue. 2 (39). - P. 5-25. DOI: http://dx.doi.org/10.15622/sp.39.1

[6] Biryukov D.N, Lomako A.G. Denotational Semantics of Knowledge Contexts in Ontological Modeling of the Subject Areas of Conflict // Proceedings of SPIIRAS. - 2015. - Issue. 5 (42). - P. 155-179. DOI: http://dx.doi.org/10.15622/sp.42.8

[7] Biryukov D.N, Rostovtsev Yu.G. Approach to constructing a consistent theory of synthesis of scenarios of anticipatory behavior in a conflict // Proceedings of SPIIRAS. - 2015. - Issue. 1 (38). - P. 94-111. DOI: http://dx.doi.org/10.15622/sp.38.6

[8] Gruber T. A translation approach to portable ontology specifications. // Knowledge Acquisition, 1993, V. 5, I. 2, pp. 199 - 220. DOI: 10.1006/knac.1993.1008.

[9] Gruber T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing? // International Journal Human-Computer Studies, 1995, V. 43, I. 5–6, pp. 907 - 928. DOI: 10.1006/ijhc.1995.1081.

[10] Guarino N., Musen M. Applied ontology: The next decade begins // Applied Ontology. - 2015. - V. 10, no. 1, pp. 1-4. DOI: 10.3233/AO-150143.

[11] Kotenko, I.V. Intelligent mechanisms of cybersecurity management // In Risk and security management. Proceedings of the Institute of System Analysis of the Russian Academy of Sciences, 2009, Vol.41, pp.74-103.

[12] Nardi J., Falbo R., Almeida J., Guizzardi G., Pires L., Sinderen M., Guarino N. An Ontological Analysis of Value Propositions. Published in: Enterprise Distributed Object Computing Conference (EDOC), 2017 IEEE 21st International. Quebec City, QC, Canada, 10-13 Oct. 2017, pp. 184 - 193. DOI: 10.1109/EDOC.2017.32.

[13] Petrenko, A.S., Petrenko, S.A., Makoveichuk, K.A., Chetyrbok, P.V. Protection model of PCS of subway from attacks type «wanna cry», «petya» and «bad rabbit» IoT, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 945-949. DOI: 10.1109/EIConRus.2018.8317245

[14] Petrenko, S.A., Makoveichuk, K.A. Ontology of cyber security of self-recovering smart Grid In CEUR Workshop Proceedings, 2017, Vol-2081, pp. 98 – 106. http://ceur-ws.org/Vol-2081/paper21.pdf

[15] Pospelov D. A. The modeling of reasoning. Experience in the analysis of mental acts. - M .: Radio and communication. - 1989. - 184 p. (In Russian).

[16] Leontiev V., Gordeev E. On the Algebraic Immunity of Coding Systems. Voprosy kiberbezopasnosti *[Cybersecurity issues]*, 2019, No 1 (29), pp. 59-68. DOI: 10.21681/2311-3456-2019-1-59-68.

[17] Pospelov D. A. Thinking and automatons. - Moscow: Soviet radio. - 1972. - 224 p. (In Russian).

[18] Sheremet I. A. Augmented Post Systems: The Mathematical Framework for Data and Knowledge Engineering in Network-centric Environment. Berlin, 2013. 395 p.