Advances in Science, Technology and Engineering Systems Journal Vol. 5, No. 3, 291-298 (2020)

www.astesj.com

Special Issue on Multidisciplinary Innovation in Engineering Science & Technology

ASTESJ
ISSN: 2415-6698

Based on Reconfiguring the Supercomputers Runtime Environment New Security Methods

Andrey Molyakov*

Institute of information technologies and cybersecurity, Russian State University for the Humanities, Moscow, 117534, Russia

A R T I C L E I N F O

Article history:
Received: 09 April, 2020
Accepted: 18 May, 2020
Online: 29 May, 2020

Keywords:
Kripke structures
Sandbox
Stationary and on-board
Supercomputer systems
CTL logic

A B S T R A C T

This paper is an extension of work originally presented in 2019 Third World Conference on Smart Trends in Systems Security and Sustainability (WorldS4) [1]. Author describes two new methods: reactive protection method (without delay after detecting an attack), which consists in virtualizing the execution environment of supercomputers processes if the calculated state descriptor falls into the "risk" zone and based on monitoring requests for allocation of resources in accordance with the rules of the security policy in the form of temporal modal structures CTL logic and method for reconfiguring the runtime environment of the supercomputers taking into account the mobility requirements (built-in computations) based on the application of the trajectories of computing state security descriptors on Kripke structures. The methods develop a number of provisions of the theory of information security, based on the new concept of Information Security of stationary and onboard supercomputer computing systems as a calculated convolution of the states of the execution environment (hardware or virtual) and system software.

1. Introduction

Widely used computers of various classes are essentially systems with von Neumann architecture, and the program execution model is a universal Turing machine with a tape of calculations, left and right shifts.

The performance criterion in the classical Popek - Goldberg theory for Turing machines is ambiguous. In the original work it was formulated twice [1]:

a) "a statistically prevailing subset of virtual processor instructions must be executed directly by the physical processor, without the intervention of a virtual machine monitor";
b) "all harmless instructions are executed directly by the physical processor, without the intervention of a virtual machine monitor".

These conditions are not identical. They are identical only if most of the instructions executed by the virtual machine are harmless, which cannot be guaranteed in the general case. In proving sufficient conditions for constructing a virtual machine monitor, Popek and Goldberg use the second definition.

Since the 70-80s, electronic technology has undergone significant changes. With the advent and development of hardware virtualization successfully implemented [2, 3]:

• *SMEP (Supervisor Mode Execution Prevention*). Prevention of code execution in supervisor mode) is a technology developed by Intel to protect your computer from hacker attacks and other threats that use the so-called "supervisor mode".

• *SMAP (Supervisor Mode Access Prevention)*. It prevents writing to memory and reading code from it that unauthorized uses supervisor mode, while SMEP only prevents the execution of this code.

Supervisor mode is the preferred processor mode used by the kernel of the operating system. This mode is also called kernel mode. The opposite is the user mode in which user applications work [4, 5].

For SMEP to work, in addition to the corresponding processor, a suitable operating system is required. However, SMEP, although significantly complicating the task of hacking the system, still does not guarantee its complete protection. Therefore, later (in Broadwell architecture processors), in order to increase security and protect against vulnerabilities that were not resolved by SMEP, SMAP technology was additionally introduced [6, 7].

Direct execution of guest instructions whose encodings correspond to the privileged instructions of the physical processor on the host is impossible, since they certainly cause a trap to be thrown when executed in the user mode in which virtual machines are running. For example, the encodings of new secure instructions for Intel-64 architecture usually correspond to invalid encodings on processors

*Andrey Molyakov, Moscow, 117534, Kirovogradskaya street, 25/2, Russia. Tel: 8-495-388-0888 & andreimolyakov@mail.ru

of previous generations of this architecture. That is, new instructions are usually privileged on processors of previous generations, because they cause exceptions [8, 9].

Many hardware processors and various research groups are interested in supporting hardware transactional memory [10]. IBM was the first to develop a system with transactional memory hardware support in IBM Blue Gene [11] supercomputers and z-Enterprise EC12 servers [12]. AMD announced the development of the Advanced Synchronization Facility (ASF) extension [13], which is a variant of the hardware transactional memory for the Intel 64 architecture. Sun Microsystems has developed a processor code-named ROCK [14], but this project was closed. The Intel® Transactional Synchronization Extensions (Intel® TSX) instruction set [15], consisting of Intel® Hardware Lock Elision (Intel® HLE) and Intel® Restricted Transactional Memory (Intel® RTM) extensions, has been added to the sixth generation Intel Core processors [16].

There are also many software implementations of transactional memory ideas that do not require special support from the hardware. However, the study of existing solutions shows their extremely low productivity [17].

Moreover, with the advent of hardware transactional memory, the Intel RTM extension allows you to perform nested transactions, but at the same time, all internal XBEGIN and XEND pairs should not create new save points, that is, rollback of a nested transaction leads to a rollback of the entire transaction chain. The introduction of transactional memory extensions implies that the semantics of all instructions working with memory, as well as instructions affecting the state of the processor that are not stored in savepoints, must be changed in order to support the new execution mode.

In order to avoid critical errors, it is necessary to implement a multi-level “sandbox” in which the agents of the hypervisor function and monitor the execution of processor processing instructions at all levels of the hierarchy.

2. New actual requirements for information security

The Based on the Theorem on a protected hypervisor OS, formulated and proved by Zegzhda, two requirements can be formulated [18]:

a) *Requirement 1 for virtualization of protected resources*: in a supercomputer computing system, all operations of application OS applications on protected resources must be carried out by virtualizing the resource in a virtual application environment;

b) *Access control requirement 2*: in a supercomputer computing system, all operations of application OS applications on protected resources must be controlled by hypervisor protection tools.

Further developing the idea, one can formulate theoretical premises for the solvability of the supervenience problem: all possible operating modes of the processor should be considered - the real hardware mode, the protected hardware mode, the real virtual machine mode, the protected virtual machine mode. This means that the multi-domain protected hypervisor must ensure the correct execution (software using the interpreter or hardware) of all the instructions of the guest system, as well as guarantee isolation of virtual machines. At the same time, all safe instructions common to the guest and host systems are executed directly by the physical processor (privileged modes).

The virtual machine monitor operates in mode 0 (in a separate domain), therefore, any interpretation errors, memory leaks, incorrect pointers, addressing errors of data code segments, program codes, and page addressing errors do not lead to an abnormal termination.

The solution to the problem of providing a modified instruction mnemonics for processors of different generations in order to support hardware transactional memory is to prevent the direct execution of these instructions in a virtual environment - they must be simulated programmatically on virtual processors (unprivileged modes).

Taking into account the development of the element design base and hardware technologies, we can formulate ***new requirements for information protection*** [18]:

a) Protective equipment should control all informational interactions without exception;

b) Security features should be developed independently of application programs and rely on an abstract representation of information interactions;

c) Protective equipment should control information interactions based on clearly defined rules that make up the formal model;

d) A mechanism should be provided to assess the safety of both the current state of the system and predict the safety of future states.

The principle of homogeneity of memory is the main postulate of von Neumann architecture. This means that for supercomputer systems in which record high performance is important, a fundamentally new architecture is required. Sharing the bus for program memory and data memory leads to a bottleneck in von Neumann's architecture, namely limiting the bandwidth between the processor and memory compared to the amount of memory. Due to the fact that program memory and data memory cannot be accessed at the same time, the processor-memory channel bandwidth and memory speed significantly limit the processor speed. Storage of data and commands in different places solves the problem of the “memory wall” on highly loaded computational problems [1].

3. Fundamental scientific problem

Over the past 25 years, OS security tools (for example, SMEP, SMAP, PaX, ExecShield, ASLR, DEP, Flusk, Patchguard and etc.) have come a long way, but failed to provide protection against current threats. The fundamental problem for this field of knowledge (computer science and computer technology) is known as the ***problem of supervenience***: taking into account the high asynchrony and multi-connectedness of processes performed with supercomputer systems, the huge amount of processed data (Petabytes of data) there is no logical correspondence between changes in the programs of the runtime environment of supercomputers and changes hardware components in terms of implementing an isolated environment and monitoring security policy rules. For the chosen class of speakers, the solution to the fundamental problem is the development of the theoretical, scientific, practical and organizational and technical foundations of reactive protection of stationary and onboard supercomputers, which consists in virtualizing the execution environment of the processes of applied and system software, in the interest of increasing their security and reliability, as well as substantiation of the principles for constructing mechanisms for implementing such protection for next generation automated systems.

The main contradiction for this class of computing systems is the ratio of energy intensity, speed and security. With the introduction of additional functional elements using the example of a hypervisor and a transactional memory controller, the characteristics of performance and energy consumption change. Concurrently executed queries conflict when they read and modify a certain database element, and the resulting conflict can lead to an erroneous result that could not be obtained if these queries were executed sequentially. Transactional memory provides a lightweight transaction mechanism for control flows running in a shared address space. It guarantees atomicity and isolation of parallel tasks [10, 11].

Atomicity ensures that changes in the state of a program made by code that is executed in a transaction are invisible from the point of view of other transactions executed in parallel [12].

Isolation ensures that concurrent tasks do not affect the outcome of the transaction, so that the transaction produces the same result as if no other task was being performed. Transactions provide the basis for constructing parallel abstractions, which are building blocks that can be combined without knowing their internal details, much like procedures and objects provide suitable abstractions for composing sequential code [13].

Integration with hardware transactional memory requires solving 5 main problems of computer science: limited application, debugging complexity, process synchronization and exclusion of access, resource control in conditions of parallelism and high asynchronous processes, emulation of different types of processors [14]. For multi-domain protected systems, trust is ensured by the following factors: *"Transparency"* – invisibility for the application OS, working directly with equipment, the amount of hypervisor code is small compared to the OS and applications, therefore it is easier to ensure that there are no vulnerabilities, all actions and events inside the virtualized systems are reversible - attacks can be quickly neutralized by rollback or reset.

Monitoring the exchange of a virtualized system with the external environment allows you to abandon a thorough study of the virtualized system itself.

Limitations associated with loss of productivity are extremely low, due to the record high performance characteristics of supercomputers.

4. Research methodology: new concepts and definitions

Information security is based on access control to objects of managing and guest OS, these objects can be attributed to different levels of protection. Examples of objects are directories, files, network sockets, registers, and interrupt handlers and some special-use memory areas by operating systems. The traditional access control approach involves the use of access attributes (rights) in requests to these objects for performing certain operations on them. If the verification of such attributes is successful, then access to the object at its security level is allowed, then the requested operation is performed on it. With this approach, it is technically possible to intercept a request and use its access rights in a substitute request aimed at malicious impact.

New concepts and definitions are introduced in the work: supersecurity, information security of stationary and onboard supercomputer computing systems, a descriptor for assessing the security of system states.

Supersecurity is a property of a supercomputer system based on self-configuration and self-control. By using the high-performance reserves of these supercomputers, you can create a hypervisor that provides reliable protection against attacks associated with high asynchronous processes of the supercomputers and the ability to perform false transactions as a result of destabilizing external influences and accidental hardware failures.

The information security of stationary and onboard supercomputer computing systems is formalized as "a calculated convolution of the states of the execution environment (hardware or virtual) and system software".

The descriptor for assessing the security of system states is a convolution that is "calculated" based on the attributes of ongoing processes that are implemented at the level of microprocessor cores and the interworking environment. All processes have their own descriptors calculated and methods for their processing — get, set, delete and etc. are specified.

5. New Safe Operations Model

The safe operations model includes 3 components: commands, data and timestamps. The space of operations in calculating the descriptors is a variety of Kripke's "worlds". Each world of Kripke is assigned its own digital double and form a knowledge base for machine learning. The class of operations that are characteristic of supercomputers is the operations of multiplication and division in the form of successive shifts in the tagged structures of packets of requests from clients, guest and control OSs.

From the point of view of category theory, we work with the Group object - the category group. Objects are a group in the form of a residue ring, morphisms are mappings preserving the group structure. Category theory studies concepts through how these concepts interact with each other. We forget how these concepts are implemented, and we only look at the properties of connections, abstracting from the type of processor architecture (scalar, vector, MIPS, classic x86_64, tile architecture like Tilera, mass-multithread, hybrid and etc.), processor capacity (32-bit, 64-bit, 128-bit and etc.).

A type refers to a class type when a type provides certain operations with a specific expected behavior. For example, the tau type may belong to the Functor class if it has a specific behavior similar to a collection:

a) The type **tau** is parameterized over another type, which you should consider as the type of the collection element. The type of the complete collection is then similar to Scheme_type = {Int, String, Bool and etc.};
b) If you contain integers, strings, or Booleans, respectively. If the element type is unknown, it is written as a parameter of type a. Examples include lists (zero or more elements),

type “Unknown” (zero or one element of type a), sets of elements of type a, arrays of elements of type a, all kinds of search trees containing values of type a, etc.

2. Another property that tau must satisfy is that if you have a function like a –> b (function on elements), then you should be able to use this function and product for a related function over collections. You do this with the fmap operator, which is shared by each type of Functor class. The operator is actually overloaded, so if you have an even function with type Int -> Bool, then

Definition 1. A functor is a kind of collection (a set of configurations for recursively computing a hash), for which, if you are provided with a function on elements, fmap will return the function in the collections.

Definition 2. The context of the operation is a set of tuples consisting of logical variables. A collection of collections is a conjunction of atomic predicates defined on many tuples.

I appeal to the theory of functional programming, we will define a functor for comparing is_equal of two hash values with support for different class type templates :

```
class is_equal

{ private: scheme_type v; public:

is_equal(scheme_type value) : v(value) {}

bool operator () (scheme_type x)

 { return x == this –>v; } };

scheme_type count_zero(const std::vector< scheme_type>& data)

{ return std::count_if(data.begin(), data.end(), is_equal(0)); },
```

scheme_type – data type (int, uint, float, double and etc.).

It is not possible to implement a complete hash function that is valid for all types. You cannot just convert an object to raw memory and hash bytes.

In addition, this idea fails due to padding technology when creating an index for each record. Because of this, it is necessary to take into account the context of a particular operation. To implement a universal convolution calculation algorithm, it is necessary to take into account different processor operating modes and the principle of type conversion and alignment of orders. We need to normalize the presentation of metadata operations in terms of aligning the boundaries of digital structures.

The basic principles of the algebraic structure on which the whole theory and methodology are based: The principle of Soft power, Self-organization, Supersecurity. The descriptors of the state safety assessment function are calculated on the set, which is a multiplicative group of the residue ring modulo 8.

Classical propositional logic is a “black and white” model; utterances are static, unchanged in time. In the ordinary propositional logic, sentences that do not explicitly or implicitly contain properties whose truth changes with time do not adequately formalize. We want to study and verify systems that evolve over time.

The proposed approach to the description of operations is based on the classification of risks of information security breaches and analysis of the context of the implementation of outgoing directives, through which data can be transmitted bypassing the requirements of the adopted security policy, which leads to a violation of the security of computing nodes resources.

The carriers of the analyzed operations are the sets of objects and access subjects to which various security levels (labels) are assigned. To control the security level of operations generating new entities, for example, operations, we will use the sign of immutability of the object generating the access entity.

6. Theoretical calculations of the reactive protection method

6.1. *Algebra of operations with objects processed on supercomputers*

For supercomputers the sign of immutability cannot be a constant indefinitely: simultaneously, a huge number of processes are launched. Temporal logic is needed to describe the states of subjects and access objects. We need a model of threats to the integrity of the execution environment of IC processes, in which instead of the classic link “subject, object, predicate” a new paradigm of writing security policy rules is implemented and new entities are defined - “subject”, “object” and “descriptor for assessing state security”: for each i-th threat, the value of the state security assessment function is calculated, the arguments of which are given in the form of a conjunction of predicates of eight logical variables, the subjects are the guest and control OS, and the objects are the components of the hype Sizer and transactional memory controller.

Temporal logic of branching time consider possible calculations (paths on a tree) - trajectories on a scan of the Kripke structures.

The Kripke structure is a transition system with labeled states and unlabeled transitions. Sweep defines infinite chains of states - possible calculations. Each state can have not one, but many chains - continuations, and is the root of its tree of stories (calculations).

The structure of Kripke M is the five M = (S, S0, R, L, AF).

In our case [19, 20], AF = {context_id | Dom_id | S | Ord | Context_type | TCU | TR}. Let an arbitrary formula Fi of CTL logic and a Kripke structure M. be given. For each subformula ψi of formula Fi, the marking algorithm performs the following steps:

a) We proceed to the construction of the numbering of the programs. Each operator is uniquely characterized by a pair - type (name) and a list of parameters, including operator labels, variables, functional and logical expressions. We only need to encode each of the possible values of the operator parameters. Labels of operators do not need special coding, since they are natural numbers;

b) Since each program calculates a certain function, the introduced Gödel numbering generates some numbering of the functions. On the one hand, this numbering is not one-to-one, since we code the program syntax, and any syntactically different programs have different codes. On the other hand, every computable function is computed by an infinite class of programs.

However, such numbering has fundamental and practically useful properties. Let A be the countable class of functions for assessing the safety of supercomputers states: A = {f1, f2 ..., fn}.

Even though each of the fi functions is computable by some algorithm, this does not guarantee the existence of a single algorithm for computing all functions. We call a class A uniformly enumerable if there exists a two-place computable function F such that the class A consists exactly of functions of the form F (n, x) for some n from the set N.

We call the function a universal function of class A. nF (n, x) is the uniform numbering of class A.

Our main goal is to show that the numbering of the computable functions that we have determined is uniform with respect to other numbers and that by the number of the computable function in the given uniform numbering we can effectively find its number. Using the function for assessing the safety of SC states, we can subsequently accurately calculate the numbering states on the whole variety of Kripke structures, and then identify the classes of safe and dangerous operations.

Lemma 1. If A is an effective set, then for any effective set B: AB is effective, as well as any Cartesian product A1, A2 ... An of effective sets is effective, the set A *, A * = An, of all finite sequences of elements of A is effectively countable.

The proof of the statement is trivial and follows from the main theorem of arithmetic. We formulate and prove a theorem on the enumerability of the class of operations.

Theorem 1. Class A operations on the residue ring modulo m are uniformly enumerable, and the hash value calculation function F is a universal function of class A only if the number of hierarchy (nesting) levels is 8.

Evidence. In fact, operations are performed on data tuples, which are a set of i-th elements of an allergic structure (bit, atomic predicate, predicate conjunction, any set of predicative and functional symbols, vector variable, scalar, real number) of any dimension m (regardless of encodings and positional number system). We work with matrices, where the number of rows is equal to the number of nesting (hierarchies) of the computation space N, and the i-th column is an element of a structure of dimension m in the residue ring Z (m), i = 1 ... n. Using the Kornfeld formula to assess the confidence probability of security, each factor PΛi is a geometric decreasing progression [21].The most suitable values are 0.1 and 0. (1). Criteria of temporal logic (lack of new patterns over time) satisfies only the solution N = 8.

Moreover, nF (n, x) is an effective numbering of class A, since with a change in the dimension of the parameter n the regular relations between the elements of the set do not change, new properties of objects of our algebraic structure do not appear.

The main result of theoretical calculations is that we have proved universality and uniformity for given classes of operations of functional transformations. Regardless of the dimension of the input parameters, the nature of the relationships between the types, the type of operands, the type of processor architecture, the operation scheme and implementation of the algorithms, we have found such a number of hierarchy levels (N = 8) that there is an invariant in the algebraic system. If we evaluate the time parameters, this regularity property is preserved indefinitely (from 0 to + ∞).

6.2. Integration of the algebraic structure with architectural modifications of supercomputers

We proceed directly to the construction of a weighted multigraph-model for performing operations of supercomputers, which is a tree. The starting point is the top of the tree structure, describes a transactional memory controller that interacts directly with the hypervisor verifier module. The controller in conjunction with the verifier implements security mechanisms: an isolated multi-domain address space is represented as non-overlapping memory areas, each of which corresponds to a vertex of the graph lying in the second level of the root structure. The vertices of the third level correspond to the subsets of the components that make up the hypervisor. Since we have 8 levels of the request processing hierarchy, the block is divided into eight subsets of level details of possible states indicated by S8, ... , S1.

The last level of the root structure of a multigraph is represented by agents of the hypervisor (graph leaves) through which communication with the external environment and all possible attacks on the supercomputers occur. Advances to each next level are a higher level of abstraction of the description of request processing, a transition from the lower level of the specification to the upper. Each state can have not one, but many chains - continuations, and is the root of its tree of stories (calculations).

The reactive protection method should include not only deductive algorithms for calculating descriptors for assessing the security of states and identifying threats based on marking the states of fulfillment of supercomputers requests, but also implementing inductive learning based on self-diagnostics and explanatory decision-making mechanisms based on the concept of machine learning. We calculate such impacts that ensure the safe functioning of supercomputers, and block dangerous and suspicious ones.

7. Method for reconfiguring the runtime environment of the supercomputers

The hash is calculated recursively for all processes (subprocesses) and an effective value is obtained for the regular measurement of the query path (based on past experience, pre-training). In the process, the system is being trained. Each new configuration is classified and recognized in the future based on the characteristic set of markers. Coding of identification features - a set or conjunction of simple predicates on which the formula Fi is given [20]. The subformula ψ is a recursive call of the same algorithmic procedure, only with a different set of characteristic features and properties.

6 configurations of "colors" of the tree of computation histories. We select two colors: Blue - the predicate of the presence of the transient process, Red - the predicate of changing the type of the context of the operations performed.

a) configuration allows you to track changes in local states at the same hierarchy level and identify the launch of child processes as part of the base process with the markers of interest to us;
b) configuration analyzes the entire state tree based on the selected marker (the mode of a full run of calculations in case of restarting the task). The story tree is being rebuilt;
c) analyzes the change in two parameters in the near branches of the tree (local scale, in one domain);
d) configuration to track a single (irregular) parameter change in the far branches of the computation tree;
e) configuration regularly repeated changes of the selected identification parameter along the entire trajectory of movement;
f) configuration studies the branches of the tree; over time, in the future, changes in two parameters may appear on different trajectories. At the same time, processes can be started in different domains.

Sets of configurations adapted to different contexts of operations execution, in the form of matrices of access rules, are stored by agents of the hypervisor monitoring information security events in the memory of the transactional memory controller. In the process of training, the rules of safety rules are updated and adjusted. The system dynamically evolves and modifies sets of modal rules based on CTL temporal logic grammars for responding to signaling events. Identification criteria for threats in the semantic interpretation of a set of atomic predicates.

Thus, the hypervisor in conjunction with the transactional memory controller is a multi-agent system with machine learning.

There are deductive checks that correspond to the principles of classical logic, but we are using inductive algorithms with metadata about the operation of system components at different levels of the hierarchy, where, along with checking the integrity in the form of changing the values of the calculated hash functions for each process, access control is implemented in the form Labeling allowed transients when switching between security domains and thread migrations. The history of various system configurations is kept, new events are recognized due to a variant new set of predicates and the values of their trajectories on Kripke structures.

8. Discussion

8.1. Optimization, normalization of descriptor calculation

For optimization and normalization, it is necessary to specify an exactly safe data volume. The ***axiomatic of our algebraic structure*** are as follows:

a) Multiplication and division operations are specified (cyclic shifts to the right or left);
b) A lot of descriptor calculations is a residue ring modulo m = 8;
c) The ring must be symmetrical with respect to the performance of the operations of multiplication and division;
d) A request for any tap of a processor with a classic von Neumann architecture is a combined tagged structure in which a data segment and a program code segment are stored in the same RAM sections. In this case, the L-operand for the recursive calculation of ψi is stored in the data section, the R-operand for the recursive calculation of ψi is stored in the code section.

The principle of supervenience is the absence of differences of one kind in the absence of differences of another type: the absence of differences in the set of process descriptors in the absence of changes in the configuration of software and hardware agents. There is a one-to-one logical correspondence between a change in the set of a vector variable stored in the hypervisor generative tables, a map of transactional memory states (hardware level), and a hash function value (software level). In the process of passing the request, the key line does not change normally. If it changes, then it means that the route for passing the request by a third-party agent is being modified.

_*A-property* – the identity of the values of the security assessment descriptors for the i-th process at the n-level of the hierarchy (running programs at the OS level, each process is identified by a hash value, formalized as a hash function value).

B-property – configuration of software and hardware components (formalized as a set of matrices in the form of 8 16-bit key lines, based on which the value of the function is calculated).

8.2. The solvability condition for the supervenience problem for supercomputer systems

In this case, one must take into account the upper or lower case of addresses (a segment of a data code or a command code, a combined format for representing the structure in RAM). Then the amount of precisely safe calculations doubles and you need to introduce an additional 2 * 4 = 8 register fields in the form of significant bits of the n-dimensional vector Ψ (n) = 8 to control the addressing and offset boundaries in the pipeline of processed requests and prevent the execution of "false transactions" ". Thus, the dimension of the control parameter k is 8. In this case, the bits (upper and lower registers) of the pointer to the RAM memory cells are not mixed.

If this condition is not fulfilled for the comparison functor, then the boundaries of the data segment and code are violated, the descriptor is recursively calculated: for $N < 8$, empty bits remain (you can write zeros or junk data in them), and normalize the processed data for $N > 8$ (BDC - format and type conversion of operands) is not possible.

To get this type of representation of a data set in the form of tuples of n-logical variables, the minimum set (basis) is 8. When the number of iterations is a multiple of 8, we increase the amount of data of recursive calculations, but the effects and patterns do not change. With a decrease in the number of iterations (less than 8 levels of the query processing hierarchy), automatic alignment of the processed data in the BDC format leads to the fact that individual bits or groups of bits remain uninitialized, which allows dangerous shifts in the supercomputers.

The optimization of descriptor calculations is due to eight interval constraints in the processing cycle to obtain a productive sample, using only eight significant bits (rather than infinitely large numbers) for storing and processing the results. Based on the foregoing, we formulated the solvability condition for the fundamental problem of establishing logical correspondence (supervenience) of the program execution process on the supercomputer, by changing the state of the hypervisor components and supercomputers hardware: the number of levels of the request processing hierarchy by the hypervisor N in interaction with the transactional memory controller should be eight.

Solvability condition. The number of levels of the request processing hierarchy by the hypervisor N in interaction with the transactional memory controller should be eight.

A change in the states of processes at the upper level leads to a coordinated change in states at the lower level; there is a similarity relation. When using two rings of protection (N = 2) and in the absence of physically separated storage of programs and data, the process can gain access to the “alien” segment and increase privilege levels.

9. Results

Development, taking into accounts the specifics of supercomputers, of a fundamentally new technological solution creating an isolated program execution environment in the form of an 8-level sandbox with the implementation of control mechanisms both at the level of hypervisors and at the level of transactional memory controllers. Thus, an important scientific and technical problem has been solved in the field of creating information security tools for a new class of systems - stationary and on-board supercomputers. All the means of protecting information that existed today were the hardware and software components of the protected system itself, without integration with hardware transactional memory and the introduction of multi-level control, it was impossible to solve the problem of detecting and identifying various types of threats at all levels of the hierarchy of query execution. In the course of experimental studies, new scientific results were obtained that confirm the efficiency and minimal performance loss of applying hardware virtualization technology in the form of a multi-level “sandbox” for promising supercomputers compared to using traditional clusters.

Since it is impossible to control the operation of all equipment, but only to monitor the execution of requests at the level of the components of the hypervisor and the controller of transactional memory, during the research, the maximum level of functioning of information protection agents was found - S8. With this configuration, when the number of hierarchy levels is N = 8, the execution of context-sensitive operations becomes quasi-determined with a confidence probability of approximately 0.9.

The threshold for performance loss when using the verification complex is less than 6-7%. The effectiveness of the developed security system was assessed based on the use of various security tools (used to protect clusters and mainframes) and analysis of the number of successful recognitions and errors.

The following results of experimental studies are obtained:

a) with the number of training samples n > = 86, stable detection with critically minimal errors of the 1st and 2nd kind is ensured;
b) only 8 clusters are enough for effective detection of malicious code, which significantly reduces the cost of implementing a software and hardware solution. Each cluster is bound to a security domain. Their number is also 8.

Thus, an unambiguous correspondence has been achieved between the number of hierarchy levels and the number of protection domains and the number of countable clusters tied to a certain “sandbox” level. To ensure the requirements of the international standard ISO 5725 for the convergence and reproducibility of measurement data, the experiments were performed in a series of 5 repeated experiments for 14 days until the observed data were stabilized**.**

10. Conclusion

a) The theorem on the uniform enumerability of functions for assessing the safety of states is proved. In this case, the means of protection of stationary and onboard supercomputers are considered as an object of research for the first time;
b) The solvability condition is formulated, as a consequence of the theorem on the uniform enumerability of functions, the problems of establishing the logical correspondence (supervenience) of the program execution process on a supercomputer, by changing the states of the components of the hypervisor and equipment;
c) A reactive protection method has been developed (without delay after detecting an attack), which consists in virtualizing the execution environment of supercomputers processes if the calculated state descriptor falls into the “risk” zone and based on monitoring requests for allocation of resources in accordance with the rules of the security policy in the form of temporal modal structures CTL logic;
d) A method has been developed for reconfiguring the runtime environment of supercomputers taking into account the mobility requirements (built-in computations) based on the application of the trajectories of computing state security descriptors on Kripke structures;
e) It is proposed to use a model of threats to the integrity of the execution environment of supercomputers processes, in which instead of the classic link “subject, object, predicate” a new paradigm of writing security policy rules is implemented and new entities are defined - “subject”, “object” and “descriptor for assessing state security” : for each i-th threat the value of the state security assessment function is calculated, the arguments of which are given in the form of a conjunction of predicates of eight logical variables, the subjects are the guest and operating systems, the objects are components of the hypervisor and transactional memory controller;

f) A safe operation model has been developed that considers, from the standpoint of "integrity" of the runtime environment, both supercomputers hardware at the microprocessor level and system software, describing its processes in the form of decomposition into an 8-level hierarchical structure: each process privilege level uniquely corresponds to a domain number protection and operation mode of the microprocessor.

References

[1] Molyakov A. S., "New security descriptor computing algorithm of Supercomputers" / 2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4), IEEE Xplorer Digital Library. https://doi.org/10.1109/WorldS4.2019.8903965

[2] Popek G. J., Goldberg R. P., "Formal Requirements for Virtualizable Third Generation Architectures" / Communications of the ACM, 1974.

[3] F. Leung, G. Neiger, D. Rodgers et al, "Intel® Virtualization Technology" : Intel Technology Journal., 2006, 10.

[4] AMD Corporation. — AMD64 Architecture Programmer's Manual Volume 2: System Programming, 2013.

[5] Stephan Diestelhorst, Martin Pohlack, Michael Hohmuth et al., "Implementing AMD's Advanced Synchronization Facility in an out-of-order x86 core" / 5th ACMSIGPLAN Workshop on Transactional Computing, 2010.

[6] Intel® Software Development Emulator, 2012. https://software.intel.com/en-us/articles/intel-software-development-emulator .

[7] Rechistov Grigory, Plotkin Arnold, "Implementation of Intel Restricted Transactional Memory ISA Extension in Simics" / Procedia Computer Science, 2013, 18 , 1804 – 1813.

[8] Herlihy Maurice, Moss J. Eliot B., "Transactional memory: architectural support for lock-free data structures" / Proceedings of the 20th annual international symposium on computer architecture, ISCA '93,New York,, USA : ACM, 1993, 289–300.

[9] Amy Wang, Matthew Gaudet, Peng Wu et al., "Evaluation of Blue Gene/Q Hardware Support for Transactional Memories" / Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, PACT '12, 2012, 127–136.

[10] Jacobi Christian, Slegel Timothy, Greiner Dan, "Transactional Memory Architecture and Implementation for IBM System Z" / Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, 2012., 12, 25–36.

[11] Jaewoong Chung, Stephan Diestelhorst, Martin Pohlack et al., "ASF: AMD64 Extension for Lock-free Data Structures and Transactional Memory" / Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, 2010.

[12] Moir Mark, Moore Kevin, Nussbaum Dan, "The adaptive transactional memory test platform: a tool for experimenting with transactional code for ROCK" / Proceedings of the twentieth annual symposium on Parallelism in algoithms and architectures, SPAA '08, New York, USA : ACM, 2008, 362–362.

[13] Maurice Herlihy, Victor Luchangco, Mark Moir, William N. Scherer , "Software Transactional Memory for Dynamic-sized Data Structures" / Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing, PODC '03, New York, USA : ACM, 2003, 92–101.

[14] Dice Dave, Shalev Ori, Shavit Nir, "Transactional Locking II" / Proceedings of the 20th International Conference on Distributed Computing, DISC'06, Berlin, Heidelberg : Springer-Verlag, 2006, 194–208.

[15] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset" / SIGARCH Comput. Archit. News, 2005, Vol. 33, no. 4, 92–99.

[16] Yi Liu, Yangming Su, Cui Zhang et al..,Efficient Transaction Nesting in Hardware Transactional Memory / Architecture of Computing Systems - ARCS 2010, Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, 138–149.

[17] Moore Kevin E, "LogTM: Log-Based Transactional Memory" / In Proceedings of the Twelfth IEEE Symposium on High-Performance Computer Architecture, 2006, 258–269.

[18] .Zegzhda D..P., "Building secure operating systems based on virtualization technology" [presentation], St. Petersburg, 2018, 33 p. https://docplayer.ru/storage/77/76383820/1589898846/qi9gw3tca9toeeJG7VgqWQ/76383820.pdf

[19] Molyakov A. S., "A Prototype Computer with Non-von Neumann Architecture Based on Strategic Domestic J7 Microprocessor" / Automatic Control and Computer Sciences., 2016, 50(8), 682 -686.

[20] Molyakov A. S., "Token Scanning as a New Scientific Approach in the Creation of Protected Systems: A New Generation OS MICROTEK" / Automatic Control and Computer Sciences, 2016, 50(8), 687-692.

[21] Molyakov A. S., "Threat model and theoretical foundations of the reactive protection method of supercomputers" / Natural and technical sciences, Company Sputnik +, 2019, 7, 197–201.