International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 2 Issue 5 ǁ May. 2014 ǁ PP.42-44

Secure Programming Language C_s

M.A. Malkov
Russian Research Center for Artificial Intelligence

Abstract: This is a preliminary communication on the computer language C_s, which provides the full security of a computer. We call the security *internal*. The language inherits the syntax of language C++ but with the new rules of declaration of arrays, with an extension of cycle operations, and without pointers, goto, and functions like calloc-free (allocation-deallocation of memory). The language can provide *external* security (from cyber-attacks), if its operation system changes the incoming programs in accordance with standards of C_s. And if its Internet provider is special.

I. Introduction

There are internal and external computer securities. The internal security protects data and programs if programs created by a language are executed. The external security provides protection against spyware, viruses, and the other cyber-attacks.

We offer the programming language C_s for the internal security. In particular, C_s provides protection against wrong addresses and infinite loops.

The language can be used to provide the more deep internal security and the external security, too. For that its operating system must permit only programs created by the language.

This means that the system is closed for programs not created by the language. So it is necessarily to create all existing programs anew. This is a very difficult problem. But we can overcome this by compositing a program which remakes existing programs in accordance with the standards of the language. This program must be used by the operation system to change programs incoming from Internet.

All users of the language must use its operating system and a special processor. The processor provides additional computer security and accelerates execution of programs. The motherboard is special too, since it provides additional security. More detailed information about the processor and the motherboard is out of the communication.

II. Syntax of language

The language inherits the syntax of programming language C++ [1] but pointers and functions like calloc-free (allocation-deallocation of memory) are excluded and rules of declaration of data arrays are changed. Some additional changes must be done, too.

The pointers are excluded since they give unrestricted access to any part of memory. But there are implicit pointers, they are names of data and programs. These pointers are dereferenced at executions. If values of a pointer are pointers, i.e. values of a name are names, then the pointer is dereferenced twice. The programming language Ruby [2] does not contain explicit pointers, too.

The functions of allocation-deallocation lead to random distribution of memory. They must be removed too, since memory for data and programs is automatically allocated if a program begins a block and deallocated if the program leaves the block, or is allocated-deallocated in accordance with the memory classes *static*, *extern* or with modifier *public*, the modifier is used as additional memory class. This allows allocating continuous areas of memory for data and programs. As a result, the control of security of data and programs becomes simpler.

The rules are changed for declarations of data arrays with variable length. The new rules of the declarations demand to indicate not only maximal length of arrays but to indicate the variable containing current length of the first component of arrays. This allows interrupting a program if an address is outside of an array, i.e. this ensures the security of data and programs. Furthermore, arrays can be operated entirely without detailing its structure.

It is enough to specify the length of the first component of arrays since, by rules of the language C++, the length of the remaining components is fixed.

This is an example of declaring a data array: *int* m, M[1000,m] [2]={0,1}. The array is declared such that the current value of the first component equals 1 (*m*=1), the current length of the array equals 1*2 bytes, the maximal length is 1000*2 bytes. If one writes a new value of the array he must use *m* as the first component.

The operators *goto*, *while*, *do-while* are excluded, the operator *for* can replace *while* and *do-while* operators. The operator *for* has the additional fourth field containing the maximal number of cycling. By default the number equals 1 000 000. Hence infinite loops are excluded.

III. Compiler of language

At mathematical point of view, each class of language C++ is an algebra since the class contains a set (data) and operations on this set (programs). A project is an algebra too, and classes are, in fact, subalgebras of the project. The compiler allocates each algebra a part of RAM. This part contains an instruction area and a data area. The instruction area is inaccessible for changing by its instructions and takes a continuous part of memory. The data area is available for changing and also occupies a continuous part of memory. All these will allow controlling the security of instructions and data with minimal expense of time.

The compiler provides relative addresses of instructions and data. The address 0 of instructions is an address of instruction area beginning, the address 0 of data is an address of data area beginning. Programs will be interrupted if an address is out of data area or out of array area.

The compiler gives a relative address of memory. The operation system gives a real address of memory and places data area direct after program area.

The compiler must identify program errors as much as possible. All existing algorithms must be used.

IV. Operational system

The operational system (OS) is constructed only by means of the language. OS is an algebra too, the compiler allocates OS not only relative, but also a real memory. In this case, the compiler performs a simple function of OS.

OS performs the basic functions of operational systems, in particular organizes execution of programs in multitasking mode, allocates real memory for compiled program, works with external devices, provides a library of utilities, and so on. OS does not interfere with inner workings of programs and not collect information about the inner workings. But OS interrupts programs and then removes them if an address is wrong or by a user request.

OS increases a data area on request of an algebra. If the increase is not possible OS performs garbage collection by moving algebras in memory.

OS allocates memory for the algebra of the system designer. Only the site of the designer has access to this algebra.

V. Site of designer

The designer is a very big collective of programmers since the designer must create immense volume of soft. This site is a server for all users of the language system.

The site provides users of the system with new programs and with corrections of existing programs. All programs are tested for safety and put into the library by a special program.

The site is an Internet provider. It provides users with unlimited Internet connection. The site performs only protection against cyber-attacks such as DDOS. For that the site records all users of the security system and their subscribes. The site blocks all unregistered subscribers during a cyber-attack. The blocking is partial since a cyber-attack can be organized by using only subscribers of users of the security system. In this case, the site does not block users (registered and unregistered) identified by a random selection.

VI. Other properties of language

The other properties are used to simplify programs. Then some properties of C++ become unnecessary but they can be used by one if he likes them.

The operator *if* is unnecessary, it is replaced by the operation ?: . The second and third operands of the operation can contain any operators, in particular the operator *for*. The third operand can be absent.

The language C++ is functional. Every operator has a value, the value can be *void*. If a sequence of operators is closed by parenthesis then a value of the sequence is a value of the last operator. But a sequence of operators for functions has values for every operator. The property is used by every two-placed operation in C_s. So there are a sequence of values for every two place operation $\circ$: $(X_1,...,Xn)\circ(Y_1,...,Yn)=(X_1\circ Y_1,...,Xn\circ Yn)$. But there is only one value if a symbol $\circ$ stands before a sequence: $\circ(Z_1,...,Zn)=Z_1\circ\ldots\circ Zn$. For example, $\&\&((X_1,X_2,X_3)||(Y_1,Y_2,Y_3)) =(X_1||Y_1)\&\&(X_2||Y_2)\&\&(X_3||Y_3)$.

Dots are used to construct a sequence of values with some step. If the step equals 1 then one use the construction a,…,b for the sequence a,a+1,…,b if $b\le a$, and the sequence is empty if b>a. For example, *int* m,n=1000,a[m,n]={b,…,c}; . If a step does not equal 1, then one use the construction a,b,…,c for the sequence a,a+(b-a),a+2*(b-a),…,a+m*(b-a) where $a+m*(b-a)\le c<a+(m+1)(b-a)$.

Labels in C_s have from 1 to 8 any symbols except : . The labels can be used as commentaries before blocks or operators. Then they have no reference. But there are labels that have references. In addition to C++, the labels can be used in the operators *continue, break* and *switch*. Then the operator *switch* is used like *continue* if *continue* will be out of a cycle. For example, *switch*(a+b), *break*(?). And there are arrays of labels, they are used by the same operators.

In C_s definitions of data can be added by pointing out data length. For example, *int* 5bit X, 1000byte Y, 100 Z; float 1000.10 F;, here *X* has length of 5 bit, *Y* has length of 1000 bytes, *Z* has length of 100 decimal digits, F has mantissa of 1000 decimal digits and exponent of 10 decimal digits, so $10^{-10}<F<10^{10}$.

New types are introduced only by using specifier *typedef*. In particular the definition: *typedef struct* X \{*int* x; *float* f;\};, introduces the new type, the definition: *struct* X, Y {*int* x; *float* f;}; , does not introduce a type.

Definitions of classes are replaced by definitions of structures, components of the structures can be programs. The definitions do not differ from definitions of classes.

The priorities (ranges) of operations &, | and ^ are changed. They have the same priority, which follows after the priority of operations << and >>. For example, one can use the expression a | b>>c!=d instead of the expression (a | b>>c) != d.

An arithmetic expression can not be Boolean. This allows the compiler to find more mistakes in programs. So one must use the expression x==0 instead of !x.

Non-recursive functions are executed more simple and without any additional time of .their execution. These functions can be used instead of *inline functions* without additional memory requested by *inline functions*.

The type *char* is used only as an array of characters without the symbol \0 in the end. The type cannot be used as an array of integers of length 8. Instead the type *int* 1byte must be used. The new type *string* is used as an array of characters with the symbol \0 in the end. The operation + is used to concatenate data of types *char* and *string*.

If one inputs data he can point out to execute some operators without waiting end of the imputation. The sequence of the operators must be ended by the operator *wait*; .

References

[1] B. Stroustrup, *The C++ programming language*, fourth edition, Addison-Wesley, 2013.

[2] David Thomas, Chad Fowler, Andrew Hunt, *Programming Ruby: The Pragmatic Programmer's Guide*, second edition, Addison-Wesley, 2004.