Enhancing cryptographic stability of applications based on the principles of dynamic polymorphism

B R Avetisyan¹, E O Pokhomchikova² and I M Daudov³

¹General Prosecutor’s Office of Russian Federation Research Institute, Moscow, Russia.
²Irkutsk National Research Technical University, 83, Lermontov street, Irkutsk, 664074, Russia
³Chechen State University, 17а, Dudaev Boulevard, Groznyy, 364015, Russia

E-mail: kshakalov@mail.ru

Abstract. Today, encryption reflects a high level of reliability and confidentiality when working with data, both within local networks and in the global Internet network. At present, data processing has become an integral part of society. It penetrates practically the whole sphere of human activity. There are negative consequences of the development of this area. Very achievements in the field of high-tech technologies are used everywhere and are very new.

1. Introduction

Information in the form in which it is processed using a personal computer and modern information and telecommunication technologies is one of the crucial factors in the functioning of the information society [1, 5]. It should be noted that the most significant number of 0day attacks target the last application versions, turning to a more detailed consideration of the use of the concept of self-modifying code. After the first “crash” of the program, in the case of dynamically downloading the latest software versions from the server, it will be possible to update the software to an unsupported vulnerability.

2. Areas of application for self-modifying cod-client server applications.

In client-server applications, consideration of such a mechanism as self-modification is possible within the framework of special server programs (for which the presence of a high uptime is critical and straightforward even in a second is unacceptable), malicious programs (updating according to the “classical” scheme can awaken heuristics, UAC may not work according to due to local rights restrictions, or to attract the attention of the user/administrator, microcontroller firmware (updating via standard tools, such as a COM port, in the case, when there are more than a dozen devices – per cent fool costly enough).

Besides, there are such programs that, for example, load modules into third-party applications, sometimes system ones, in such cases, shipment can lead to a system crash, when self-modification can keep the system working even in the conditions of protection against changes.
3. Implementation of the concept of self-modification

In the modern world, countering threats to the security of society and the state is reaching a radically new level. The level where we remotely use technical means to implement such a list of threats, the damage of which is not comparable with the costs of its implementation. The development trend of information technology reflects the buildup of a new level of threats that have to be confronted. Thus, effective confrontation is possible when more and more effective methods and means are introduced into the arsenal of the state that meets threats to the information security of individuals, society and the state.

Not only all the information classified as restricted access but also many operations in the information environment takes place on an electronic level [6].

Turning to a more detailed discussion of the use of the concept of self-modifying code in applications used by departments of the internal affairs bodies, it should be noted that the most significant number of 0day attacks target the latest versions of applications. After the first "crash" of the program, in the case of dynamically downloading the latest software versions from the server, it will be possible to update the software to an unsupported vulnerability. An analysis of the use of the IC concept reflects high efficiency in increasing the cryptographic stability of working applications.

WriteProcessMemory.

```c
int WriteMe(void *addr, int wb)
{
    HANDLE h = OpenProcess(PROCESS_VM_OPERATION |
                           PROCESS_VM_WRITE,
                           true, GetCurrentProcessId());
    return WriteProcessMemory(h, addr, &wb, 1, NULL);
}

int main(int argc, char* argv[])
{
    asm {
        push 0x74 ; JMP >> JZ
        push offset Here
        call WriteMe
        addesp, 8
    }
    printf("Holy Sh^&O
sIX, it worked! #JMP SHORT $2
was changed to JZ $2n");
    return 0;
}

Copy from start function to stack.

void Demo(int (*_printf) (const char *,...))
{
    _printf("Hello, OSIX\n");
    return;
}

int main(int argc, char* argv[])
{
```
char buff[1000];
int (*_printf) (const char *...,);
int (*_main) (int, char **);
void (*_Demo) (int (*) (const char *...,));

_printf=printf;
intfunc_len = (unsigned int) _main (unsigned int) _Demo;

for (int a=0; a<func_len; a++)
    buff[a] = ((char *) _Demo)[a];
_DEmo = (void (*) (int (*) (const char *...,))) &buff[0];
_DEmo(_printf);
return 0;
}

Encryption function Demo.

void _bild()
{
    FILE *f;
    char buff[1000];
    void (*_Demo) (int (*) (const char *...,));
    void (*_Bild) ();

    _Demo=Demo;
    _Bild=_bild;
    intfunc_len = (unsigned int) _Bild (unsigned int) _Demo;
    f=fopen("Demo32.bin", "wb");
    for (int a=0; a<func_len; a++)
        fputc(((int) buff[a]) ^ 0x77, f);
    fclose(f);
}

So, as an example of a dynamic software loader, consider the program algorithm:
The first step is to initialize the underlying virtual heap.
The second step is to allocate memory for the base class.
The third step is the transfer of control to the base class; within it, the following iterations are performed: initialization of the class heap; creating an instance of the system API class; splicing system API; creating a TEB instance; Filling the TEB structure; Application socket initialization decryption of the application configuration; filling in the application configuration; receiving information about the holder.
The fourth step is to load the application into memory; within it, the following iterations are performed: opening a blocking stream and uploading data; calculation of the size of the received data; memory allocation for downloaded data; writing data to dedicated memory and freeing up the stream.
The fifth step is the transfer of control to the startup function from memory; within it, the following iterations are performed: initialization of DOS and NT file headers; file signature verification; initialization of process information structures and startup information; process creation; setting information about process context flags; obtaining the size of the process image and reconciliation with the source; cancel the display of the section; allocate memory in the process to write data; write...
headers into the process; we write each section into the process; recording the correct size of the program image; setting the context of the process; resuming the process.

Semi-automatic optimization of the state-dependent cycle. At runtime, generation of codes, or specialization of an algorithm at runtime or load time (which is accessible, for example, in the field of real-time graphs), such as the general utility of sorting, is the preparation of codes for performing key comparisons described in a specific invocation. Changing from embedded states on the object, or simulating a high-level closure construct. A patch from a subroutine (pointer) the address of the call, as a rule, is executed during loading/initialization of dynamic libraries, or at each call, patching internal links to the subroutine according to its parameters in order to use their real addresses.

Evolutionary computing systems – genetic programming. Hiding code to prevent reverse engineering from preventing detection by scanning antivirus and spyware programs. Filling 100% memory (in some architectures) with a rolling scheme of repeating opcodes – deleting data, records in hardware. Some sets of instructions define the use of self-modification to perform certain functions.

4. Conclusions
It should be understood that the described techniques are applicable not only for circumventing proactive defences, overwriting signatures, receiving fillings on the fly, or executing shellcode. If it is necessary to provide software support in a large organization, there is often a need to maintain the relevance of versions of existing applications. However, MITM attacks, lack of trust in third-party resources in the security policy may require a preliminary check of updates by specialists in the field of reverse engineering. For the timely download of updates, it is more expedient to use a single secure bootloader instead of manually updating or to replace the certificate of a secure connection in the case of updates.

Also, the existing researchers' tools allow, without extensive experience in low-level debugging, reverse-engineer applications, extract replacement tables, encryption keys, and addresses of update servers. It is possible, by resizing the sections, using resource sections, to free up space for writing the stub's "payload" in the form of code that mirrors the data sent to the organization's server, to the attacker's server. When using dynamic loading of the application into memory, it is impossible to append statically stub to unused parts of the program, for applying, the signature method has a high probability of causing the program to crash. Timely encryption of the code used reduces the possibility of detecting dangerous areas of the program, which allows focusing on developing better software, instead of writing fix is of discovered vulnerabilities.

In the modern world, countering threats to the security of society and the state is reaching a radically new level. The level where we remotely use technical means to implement such a list of threats, the damage of which is not comparable with the costs of its implementation. The development trend of information technology reflects the buildup of a new level of threats that have to be confronted. Thus, effective confrontation is possible with the introduction of ever newer effective methods and tools that respond to threats to information security.

Along with this, it is necessary to solve scientifically and applied problems, the solution of which allows making managerial decisions of an organizational and technological nature to increase the efficiency of the functioning of the information society [2-4].

References
[1] Shutaleva A V, Kerimov A A and Tsiplakova Yu V 2019 Humanization of education in digital era Perspect. of Sci. and Ed. 42(6) 31–43 DOI: 10.32744/pse.2019.6.3
[2] Tran V A, Klyuev R V, Bosikov I I and Tsidaev B S 2018 Estimation of the reliability of the ventilation control system at the mining sites Sustainable Development of Mountain Territories 10(1) 117–124
[3] Kozhiev H H, Klyuev R V, Bosikov I I and Youn R B 2017 Analysis of management of mine ventilation networks using simulation models Sustainable Development of Mountain Territories 9(4) 414–418
[4] Kopteva A V, Koptev V Yu, Starshaya V V 2020 Information Support for the Process of Multiphase Flows Transportation Based on the Introduction of a Radioisotope Non-Separation Hydrocarbon Measuring System *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* 1 pp 674-679 DOI: 10.1109/EIConRus49466.2020.9039314

[5] Zhukovskiy Y, Batueva D, Buldysko A, & Shabalov M 2019 Motivation towards energy saving by means of IoT personal energy manager platform. *Journal of Physics: Conference Series* 1333(6) 062033

[6] Zhukovskiy Y, Koteleva N 2017 Method of Data storing, collection and aggregation for definition of life-cycle resources of electromechanical equipment *IOP Conference Series: Earth and Environmental Science* 87(3) 032057