KHMELNYTSKYI NATIONAL UNIVERSITY
DEPARTMENT OF THE COMPUTER ENGINEERING AND SYSTEM PROGRAMING

APPROACH FOR THE UNKNOWN METAMORPHIC VIRUS DETECTION

Sergii Lysenko

CriCTecS
KHARKIV, UKRAINE
21 March

Co-funded by the Tempus Programme of the European Union

Local threats

From data of company Kaspersky the metamorphic virus **Sality** leads among local threats in 2015 year

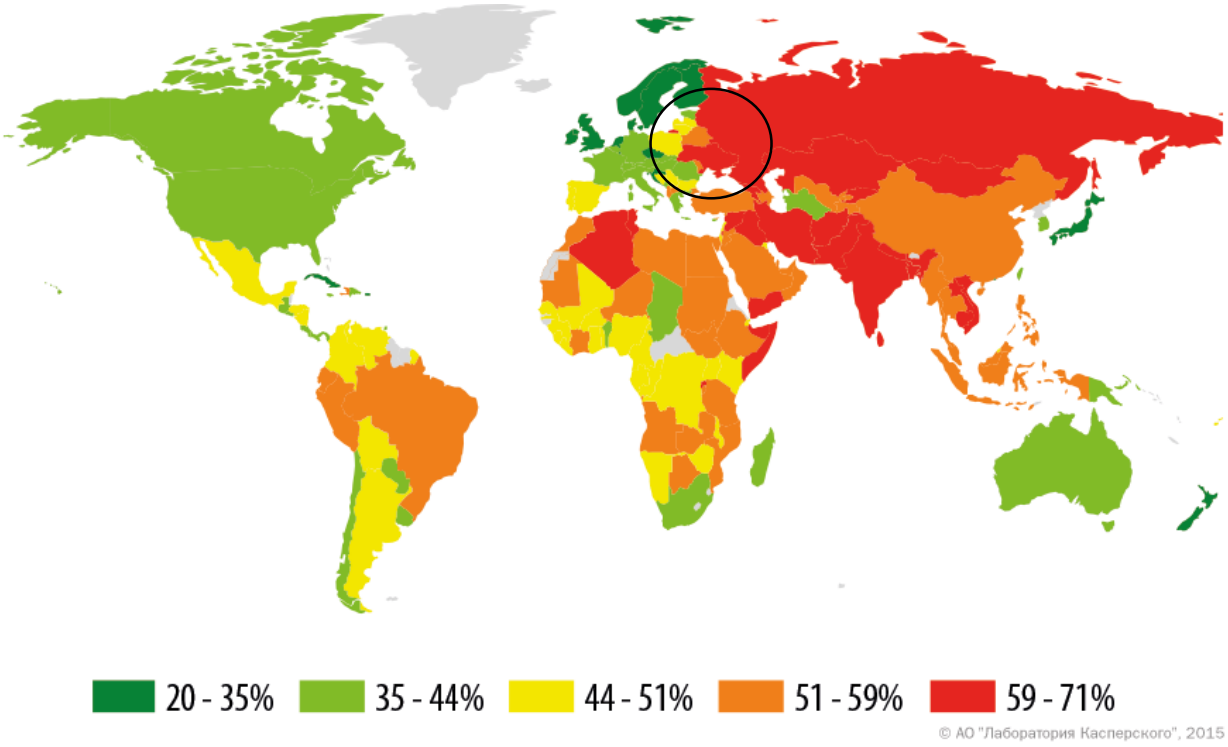
№	Назва	%, інфікування унікальних користувачів
1	Trojan.WinLNK.StartPage.gena	7,19
2	Trojan.Win32.AutoRun.gen	6,29
3	Virus.Win32.Sality.gen	5,53
4	Worm.VBS.Dinihou.r	5,40

https://securelist.ru/analysis/ksb/27543/kaspersky-security-bulletin-2015-osnovnaya-statistika-za-2015-god/

Spreading virus

Ukraine is among in the top twenty countries in the world in the number of infected computers.

60,78% unique users



Obfuscation techniques of a metamorphic virus body

To avoid detection, metamorphic viruses use several different techniques to evolve their code into new generations that look completely different, but have exactly the same functionality:

Paste of “garbage commands"

Use of equivalent instructions

Permutation of commands

Obfuscation techniques of a metamorphic virus

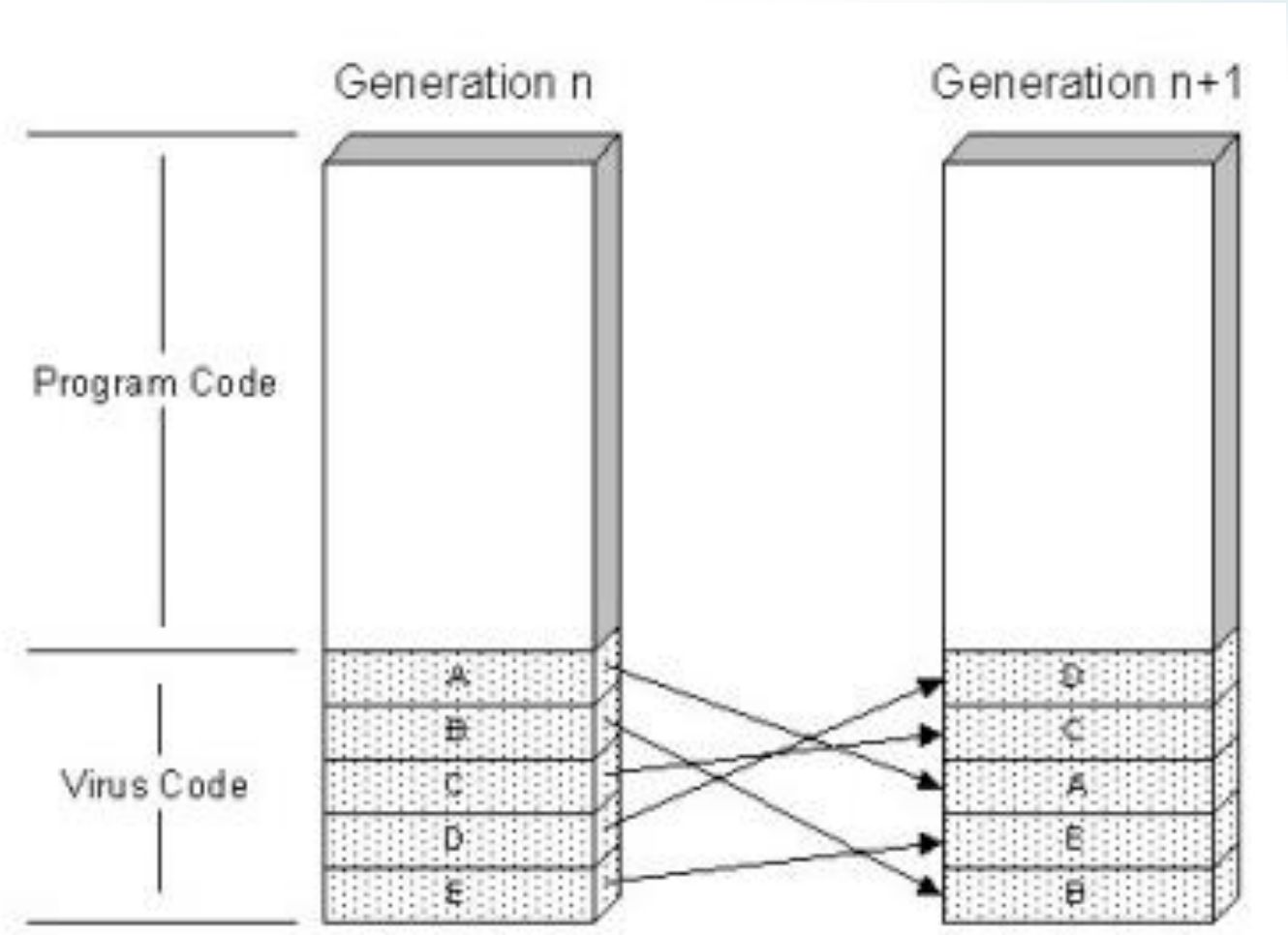
Equivalent instructions

Mov ebx,04h	ebx = 04h
Mov ecx,01h	ecx = 01h
Inc ecx	ecx = 02h
Add ebx ,01	ebx = 05h
Xor eax,eax	eax = 0h
Add ecx, 09h	ecx = Bh
Sub ecx, 06h	ecx = 05h
Add eax, ebx	eax = 05h
Add eax, ecx	eax = 10h ←

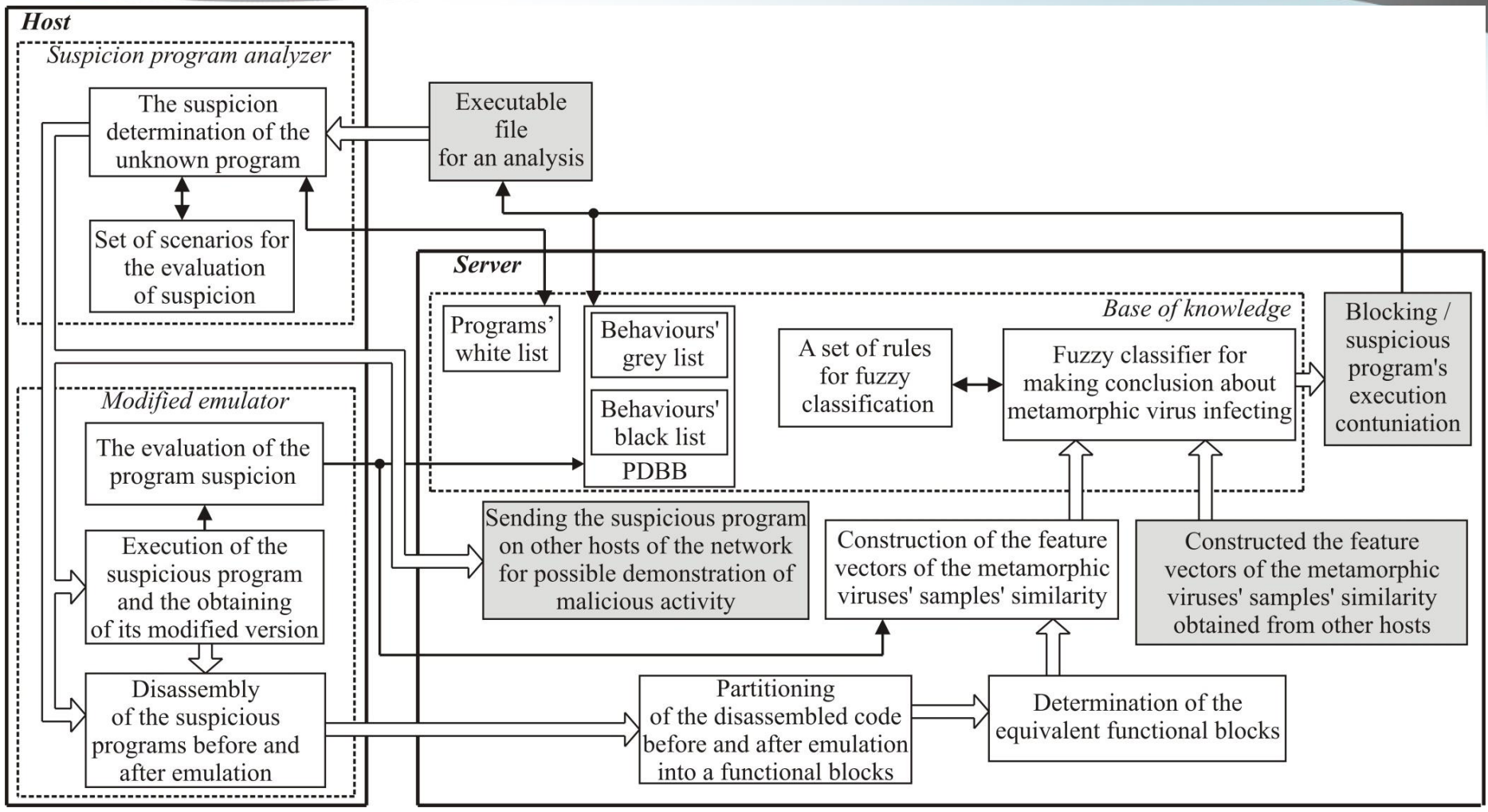
Paste of "garbage commands"

Nop
Nop
Push ecx
Pop ecx
Mov eax, 10h ←
Xchg eax, ebx
Xchg ebx, eax

Illustration of module re-ordering



The scheme of the technique for the metamorphic viruses detection



Analysis of the the program suspiciousness and program behavior construction

The basis of program suspiciousness analyzer is the set of the heuristic scenarios, grouped by the suspiciousness levels – *Deep*, *Average deep* and *Low deep*. Each level determines the conditions for the further analysis of the program.

For example, for a given level *High* the scenarios would be:

Deep: *Socket→Connect→ GetSystemDirectory,*

and in the case of the *Low* level:

Low deep: Socket→Connect.

Some features of suspiciousness of the program:

- attempt to get the rights of the system administrator;
- attempt to open or close the system port;
- trying to remove a file;
- creation a file or process;
- interception of data being entered from the keyboard;
- sending messages to the network;
- creation or entry in the registry

Modified emulator functioning

- operating system (OS) type;
- 32-bit or 64bit OS versions;
- MAC address of the guest OS;
- the hiding of the modified emulator process of execution on the host OS;
- disabling of the possibility of the data exchange between the virtual machine and the host OS (Virtual Machine Communication Interface);
- change of a registry key on the host OS HKEY_LOCAL_ MACHINE
- \SYSTEM\ControlSet001\Services\Disk\Enum

Modified emulator functioning

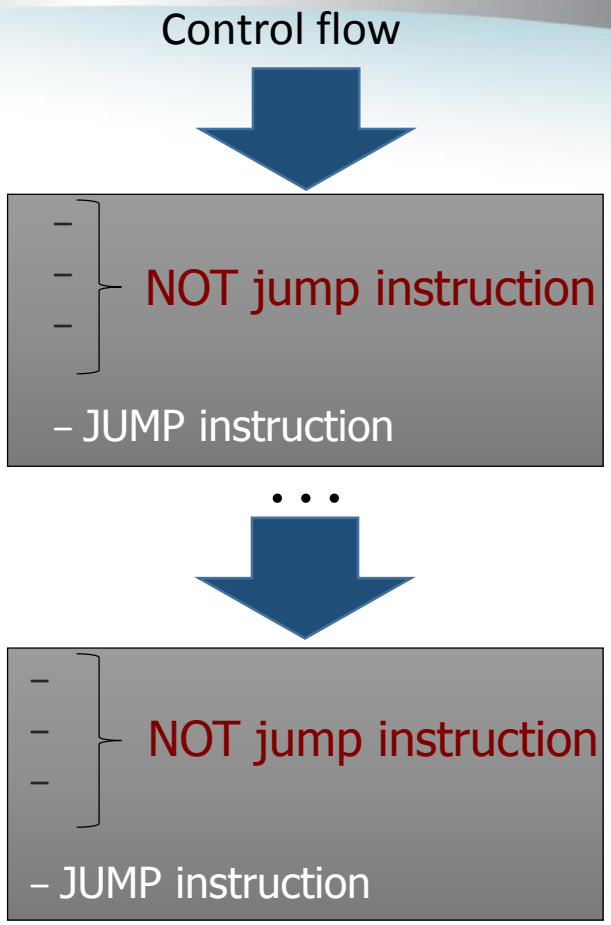
Setting the rules in the virtual machine's configuration file:

№	Rule
1	isolation.tools.getPtrLocation.disable = «TRUE»
2	isolation.tools.setPtrLocation.disable = «TRUE»
3	isolation.tools.setVersion.disable = «TRUE»
4	isolation.tools.getVersion.disable = «TRUE»
5	monitor_control.disable_directexec = «TRUE»
6	monitor_control.disable_chksimd = «TRUE»
7	monitor_control.disable_ntreloc = «TRUE»
8	monitor_control.disable_selfmod = «TRUE»
9	monitor_control.disable_reloc = «TRUE»
10	monitor_control.disable_btinout = «TRUE»
11	monitor_control.disable_btmemspace = «TRUE»
12	monitor_control.disable_btpriv = «TRUE»
13	monitor_control.disable_btseg = «TRUE»

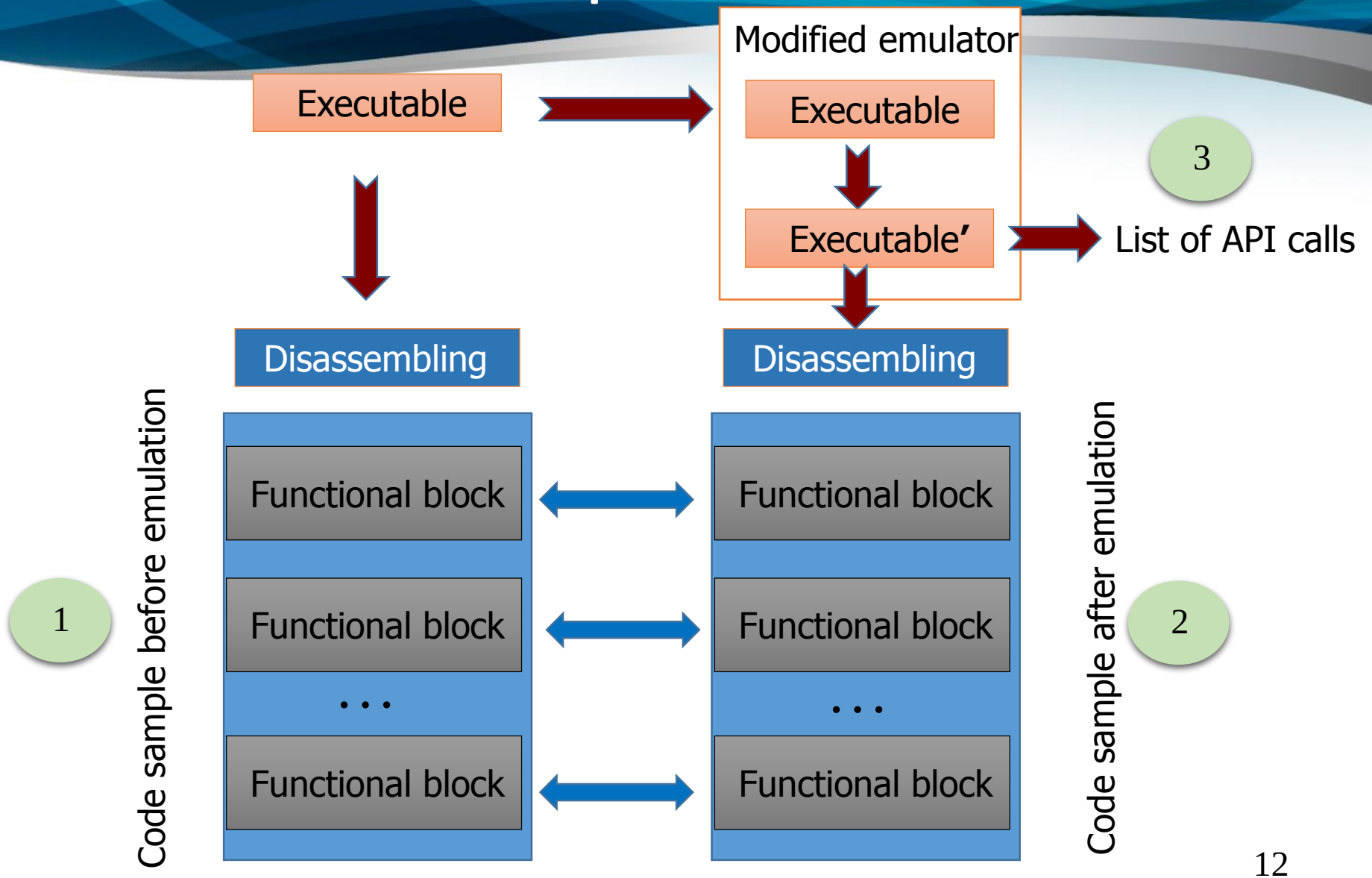
Determination of the equivalent functional blocks between the code samples before and after emulation

Function blocks characterize by:

- the control flow necessarily enters the block through the first instruction;
- inside the block may not be instructions of unconditional or conditional branch (instructions of subroutine call are allowed), all instructions in the block are executed sequentially;
- at the end of the block there is at least one instruction conditional or unconditional branch



Determination of the equivalent functional blocks between the code samples before and after emulation



Determination of the equivalent functional blocks

The procedure of the determination of the equivalent functional blocks consists of two stages

1. Determination of the occurrence of instructions in the block based on the statistical assessment

2. Involvment of the refinement of the EFB choice and the choice of the most relevant block that will be used for the construction of the feature vectors of the metamorphic viruses' samples similarity

Determination of the equivalent functional blocks between the code samples before and after emulation

1 Determination of the equivalent functional blocks

For each functional block for sample before and after emulation TF-iDF metric is used:

$$s_{FB} = \frac{n_i}{\sum_k n_i} * \log\left(\frac{N + 1.0}{n_j}\right)$$

where n_i – the number of occurrences of the i-th opcode in FB;

$k = \overline{1, k_a}$ – number opcodes in FB, where k_a is the total number of the assembler instructions;

N – the total number of FBs, $N_{F_p} \neq N_{F_s}$;

n_j – the number of FBs, which contain i-th opcode.

Determination of the equivalent functional blocks between the code samples before and after emulation

1 Determination of the equivalent functional blocks

As result will be obtain such matrixes:

$$M(FB^{Fp}) = \begin{array}{|c|c|c|c|c|} \hline & i_1 & i_2 & \dots & i_k \\ \hline FB_1^{Fp} & s_{11} & s_{12} & \dots & s_{1k} \\ \hline FB_2^{Fp} & s_{21} & s_{22} & \dots & s_{2k} \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline FB_m^{Fp} & s_{m1} & s_{m2} & \dots & s_{mk} \\ \hline \end{array}$$

$$M(FB^{Fs}) = \begin{array}{|c|c|c|c|c|} \hline & i_1 & i_2 & \dots & i_g \\ \hline FB_1^{Fs} & s_{11} & s_{12} & \dots & s_{1g} \\ \hline FB_2^{Fs} & s_{21} & s_{22} & \dots & s_{2g} \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline FB_n^{Fs} & s_{n1} & s_{n2} & \dots & s_{ng} \\ \hline \end{array}$$

Determination of the equivalent functional blocks between the code samples before and after emulation

1 Determination of the equivalent functional blocks

Compare this matrixes:

$$E\left(FB_i^{Fp}, FB_j^{Fs}\right) = \sum_{i=0,\, j=0}^{k} (s_i - s_j)^2,$$

where s_i – assessment of the opcodes appearance in the i-th block of the program F_P,

s_j – assessment of the opcodes appearance in the j-th FB of the program F_S,

FB_i^{Fp} – i-th FB of the program,

FB_j^{Fs} – the j-th FB of the program F_S.

If the value of the similarities evaluation for two FBs are less then specified threshold value δ, then the repeated calculations of the similarity evaluation for the FB of the program FB_i^{Fp} and for the next FB that follows the block FB_j^{Fs}. Mentioned above steps are repeated until the value of the similarity evaluation will be less than or equal to the threshold value.

Determination of the equivalent functional blocks between the code samples before and after emulation

2 The process of refine choice of the equivalent functional blocks

In order to choose the equivalent functional blocks define the probability of the following of opcodes in function block

For each equivalent functional blocks construct a probability matrix opcodes following

Pseudocode of filling cells of probability matrix opcodes following is given below:

for each cells in row begin

if $o_i \rightarrow o_{i+1}$ *then*

$$occur(o_i, o_{i+1}) = occur(o_i, o_{i+1}) + 1;$$

$$M_i^{probabilities} = \frac{occur(o_i, o_{i+1})}{\sum_{i=1}^{row} o_i};$$

end.

Determination of the equivalent functional blocks between the code samples before and after emulation

2 The process of refine choice of equivalent functional blocks

For example,
if functional block specified the following sequence of opcodes:
mov, push, lea, pop, mov, push, push, push, call,mov,

probability matrix opcodes following

	mov	push	lea	pop	Call
mov	0	0	0	1	1
push	2	2	0	0	0
lea	0	1	0	0	0
pop	0	0	1	0	0
call	0	1	0	0	0

	mov	push	lea	pop	call
mov	0	0	0	1/2	1/2
push	2/4	2/4	0	0	0
lea	0	1	0	0	0
pop	0	0	1	0	0
call	0	1	0	0	0

Determination of the equivalent functional blocks between the code samples before and after emulation

2 The process of refine choice of equivalent functional blocks

Comparing probability matrix opcodes following for the program before and after emulation and choice of the minimum similarity, using:

$$R = \frac{1}{N^2}\left(\sum_{i,j=1}^{N-1} | a_{i,j} - b_{i,j} |\right)^2$$

where, $a_{i,j}$ the matrix cell for the functional block FB^{Fp},

$b_{i,j}$ –the matrix cell for the functional block eFB^{Fs},

N – common amount of opcodes for the pairs of blocks.

The construction of the feature vectors of the metamorphic viruses' samples' similarity

$$\overline{V_m} = \left\langle \begin{pmatrix} L(\varepsilon_1), X(\varepsilon_1), D(\varepsilon_1), I(\varepsilon_1), R(\varepsilon_1), M(\varepsilon_1) \\ L(\varepsilon_n), X(\varepsilon_n), D(\varepsilon_n), I(\varepsilon_n), R(\varepsilon_n), M(\varepsilon_n) \end{pmatrix}, B_F \right\rangle$$

where $\varepsilon_1,...,\varepsilon_n$ pairs of the equivalent functional blocks between the program before and after the emulation,

n – a number of the equivalent blocks;

L – the Damerau-Levenshtein distance between the equivalent blocks ε_i of the program before and after emulation;

X – the number of the required opcode exchange operations;

D – the number of the required opcode removal operations;

I – the number of the required opcode insertion operations;

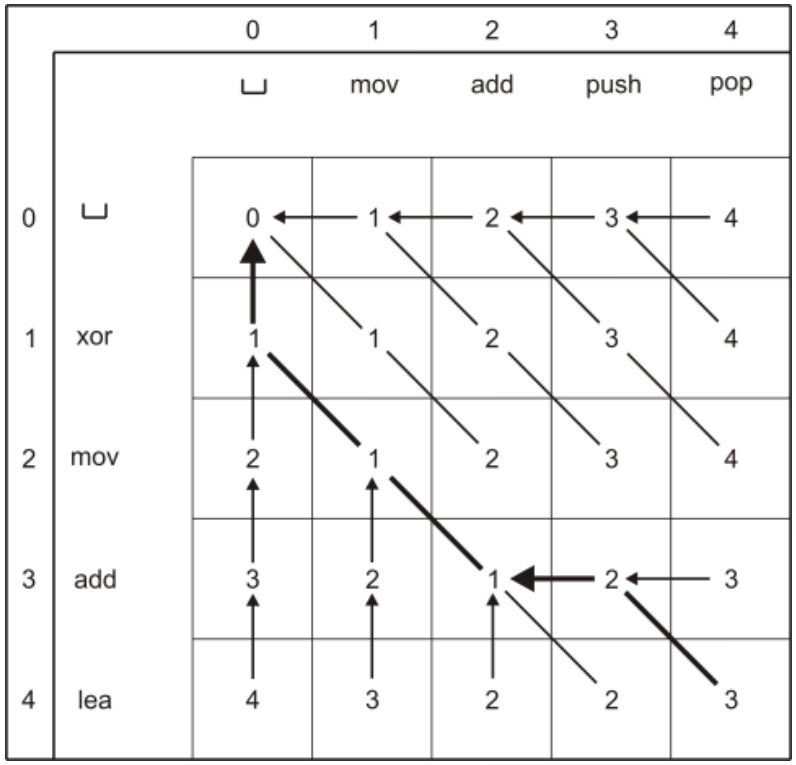
R – the number of the required opcode replacement operations;

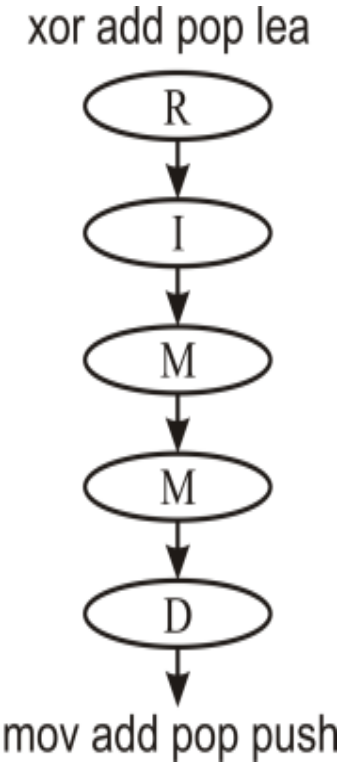
M – the number of matches between opcodes in the equivalent functional blocks of the program before and after emulation;

B_F – the danger degree behavior of the program's behavior.

The construction of the feature vectors of the metamorphic viruses' samples' similarity

Matrix of Damerau-Levenshtein for two functional blocks opcodes and transformation chain of FB1 into FB2





Feature B_F : the danger degree behavior of the program's behavior

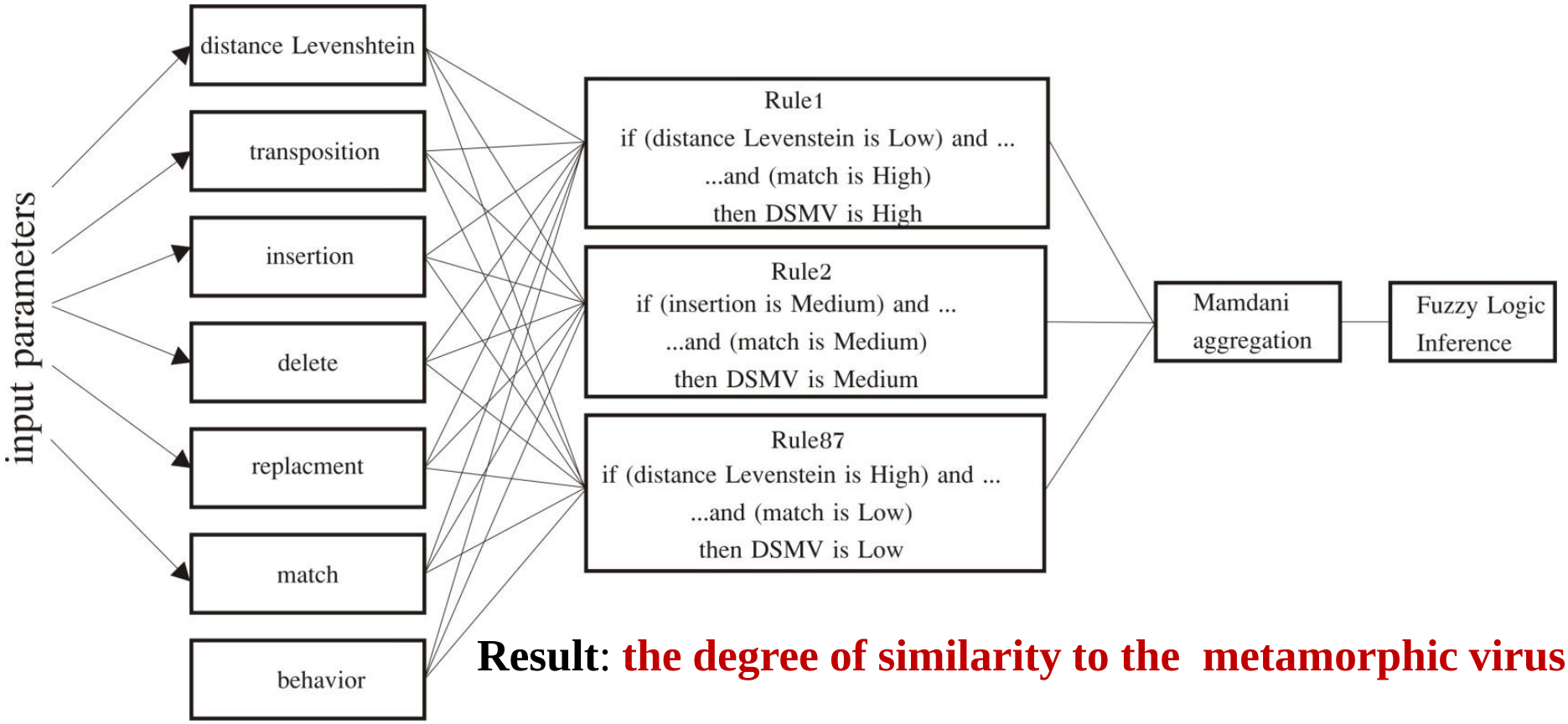
The danger degree behavior of the program's behavior is estimated on the basis of the analysis of API calls that describe the potentially dangerous behavior of the metamorphic virus

Description	Behavior's scenario to determine the High degree of danger
DLL Injection	LoadLibraryA, CreateToolhelp32Snapshot, OpenProcess, VirtualAllocEx, WriteProcessMemory, CreateRemoteThread
Anti-debugging	IsDebuggerPresent, CheckRemoteDebuggerPresent, OutputDebugerStringA, OutputDebugerStringW
File search and injection	FindFirstFileA, FindNextFileA
Finding and changing the system directory	GetWindowsDirectoryA, GetSystemDirectoryA, GetCurrentDirectoryA, SetCurrentDirectoryA
Opening and mapping file	GetFileAttributesA, SetFileAttributesA, CreateFileA, GetFileSizeA, CreateFileMappingA, MapViewOfFile, UnmapViewOfFile

The classification of the feature vectors of the metamorphic viruses' samples' similarity

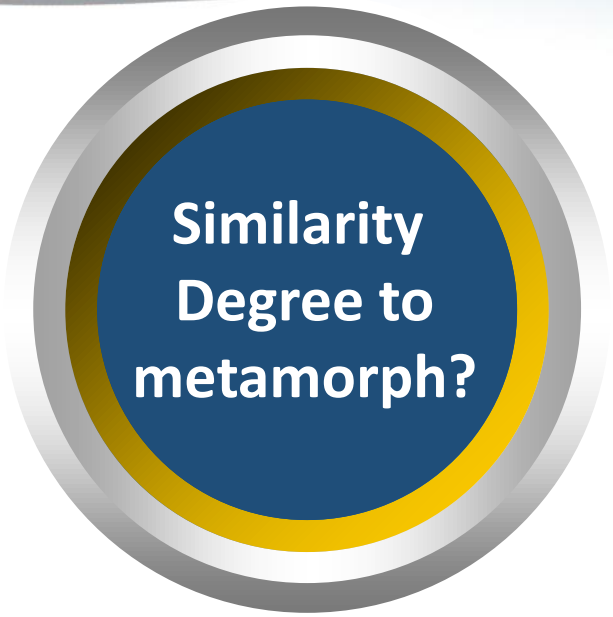
!!!!For example, one of the rules can be presented as:

$$if\ (L\ is\ Medium)\ and\ (X\ is\ High)\ and\ (D\ is\ Medium)\ and\ (I\ is\ High)$$
$$and\ (R\ is\ Low)\ and\ (M\ is\ Medium)\ and\ (B_F\ is\ High)\ then\ DSMV\ is\ High$$



Result: **the degree of similarity to the metamorphic virus**

Processing results



High

the suspicious programs is blocked on the host

Low or Medium

the suspicious programs is sent to other hosts of the network for the purpose of its execution in the modified emulators

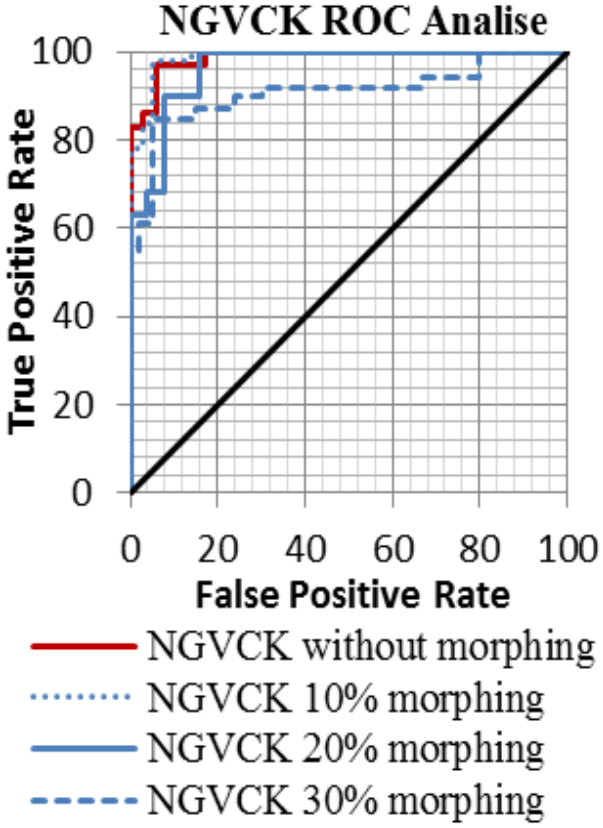
Experiments

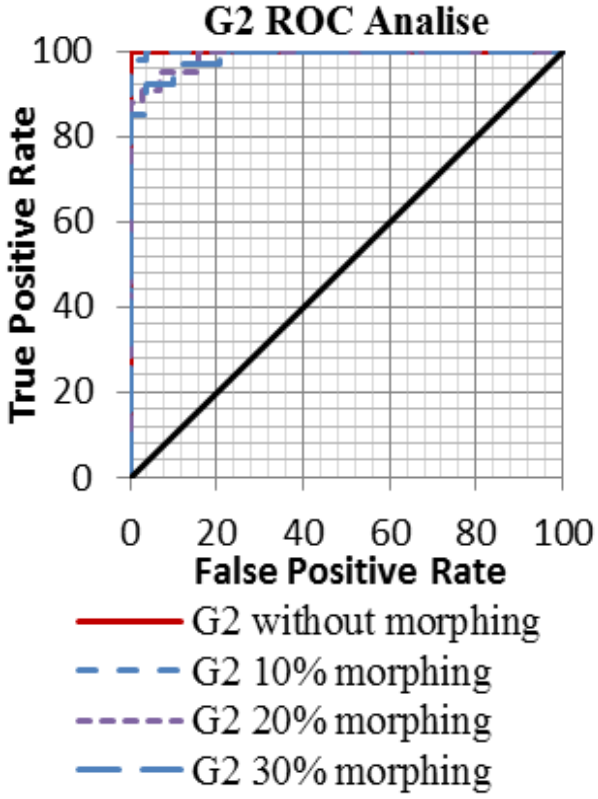
Correctly chosen functional blocks for the program before and after emulation

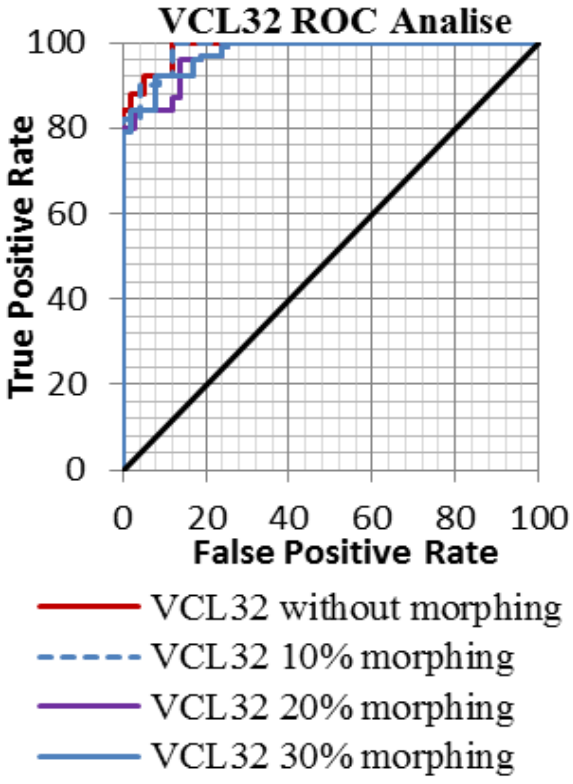
Metamorphic viruse's class	Number of correctly chosen FB, % (previuos approach)	Number of correctly chosen FB, % (new approach)
NGVCK	85	96
VCL32	88	100
G2	91	100

Experiments

ROC curves for metamorphic versions without and with obfuscation and with different values of the obfuscation degrees







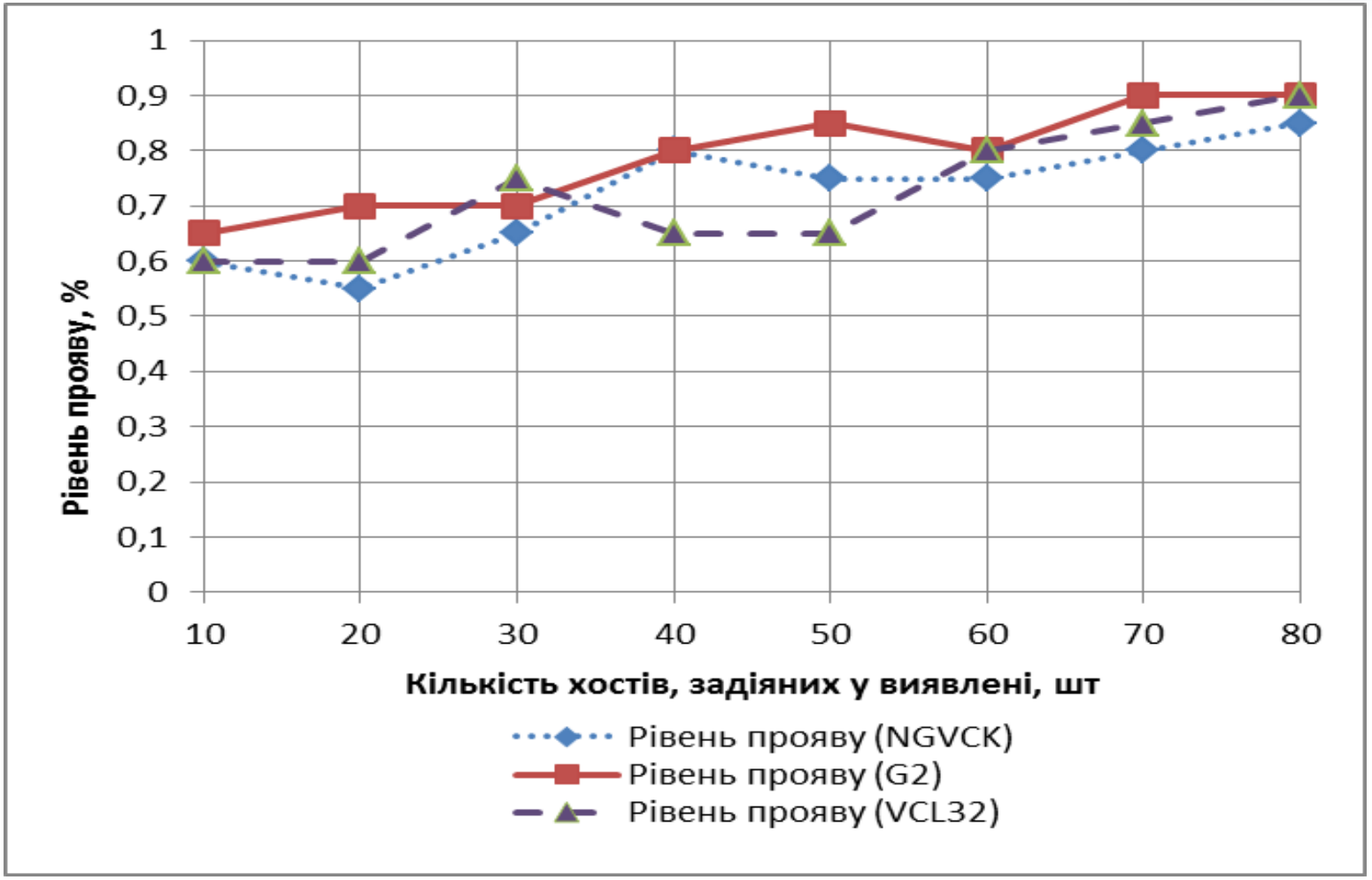
Experiments

Dependency of the accuracy of the detection on the values of the similarity threshold for two functional blocks

	Number of metamorphic viruses	Threshold value δ for determination of equivalent function blocks	Detection Rate	False positives rate
NGVCK	50	0,5	0.9242	0.1947
		0,6	**0.8671**	**0.0641**
		0,7	0.8102	0.0812
VCL32	50	0,5	0.9214	0.0789
		0,6	**0.8905**	**0.0587**
		0,7	0.8454	0.0546
G2	50	0,5	0.9987	0.0112
		0,6	**0.9752**	**0.0094**
		0,7	0.9341	0.0088

Experiments

!!!!!!The dependency of the level of metamorphic viruses' demonstrations on the hosts' number, involved in the experiment



Thank You !

Questions?

sirogyk@ukr.net, sprlysenko@gmail.com