# GENETIC PROGRAMMING IN VIRUS

I wanna comment here some ideas i have had. They are only ideas...these ideas seems very beautiful however this seems fiction more than reality.

## Evolutionary Systems

We can find in the nature some biological systems that can response to a stimulus, learn from enviroment and training,... however we can find in the nature systems that improve their capabilities with evolution of specie.
Evolution works in this manner: descendants are generated from their progenitors. Descendants are similar to their progenitors, however they have some modifications.Using a natural selection process only descendants that was addapted better to environment survive. They will have more descendants and again it will be used a natural selection process.
With these characteristics (natural selection and progenitors generating more descendants) is anough to generate descendants more poweful with generations.
We can see a analogy to understand this fact better:

We have a environment with valleys,mountains and plains. We have some individuals. A individual is more powerful than others if he is over a location with more high than others individual's locations. Descendants will be generated using characteristics of two individuals,their progenitors. Descendant's location will be in the middle of the line that links both progenitors's locations.If this middle point has more high than progenitors's locations we have a descendant more poweful than progenitors.A selection process will eliminate individuals with less high.With evolution,we will have individuals placed in higher locations.This method could be used to find summits for example.

## Genetic Programming

Genetic programming is based in these terms. We have a set of operations(elements of code considered as a unit). With these operations we have a first generation of programs. We use a selection process to select programs that adapts better. We search that programs that solve our problem more nearly posible.For evaluating a program we execute it over a set of training inputs,and we see results,and we see what program is nearer to the best solution.Then,we eliminate the worse programs. With that programs that survived, we generate a new generation of programs with two process,mutation and crossing. With mutation we generates descendants from a progenitor, changing a piece of code by other piece of the set of operations. Crossing is more effective. With crossing we generates descendants crossing two progenitors: we select randomly a operation of a progenitor and we change it by other randomly selected operation of the other progenitor.Usually,its used crossing more frequently and sometimes is used mutation,less frenquently,both combined.

Well,you can search a lot of articles about genetic programming in internet.I have commented a few what is this for now speaking how genetic programming could be used in the virus world.

## Virus

Since i read something about genetic programming i have been thinking how it could be used in the virus programming. Virus has some similaritys with genetic programming. When a virus infects a file it is generating a descendant. However, descendant is identical to progenitor. Yes, a polimorphic or metamorphic virus is different to progenitor, however it works in the same manner of its progenitor.
To use genetic programming in virus world we must see a virus as a set of operations.This operations arent xor,call,jmp... Im refering to more high level operations. For example, we could have a operational element that changes current directory or other element that infects .exe files or .hlp or others. A worm element. A payload element,....
These elements of code must works by itself, in any direction of memory. A program will be composed by these elements.We could name these elements blocks. I think this blocks could be similar to pieces of code defined in siilex article in 29a #6. Pieces of code that could be chained to do a complete program.
For example,we have a block with ability of changing current directory. We have too a block that infects .exe files with itself and others blocks in its current infected file. And too, we have a block with antidebbuging trickz and a block with a payload.We will generate randomly programs with 6 blocks.We could have a lot of posibilities:

1º ChangeDir-Infector-ChangeDir-ChangeDir-Infector-Payload
2º Antidebug-ChangeDir-Infector-ChangeDir-Infector-Payload

```
   ...
nº Antidebug-Antidebug-ChangeDir-ChangeDir-Payload-ChangeDir
```

Well,with these blocks we have generate n programs. If we see previous showed programs, its easy to say that 2º program will be more effective than 1º and nº. If these three programs are in the wild, second program will be more time in the wild.In fact,nº program will not infect others programs.nº "will die" first, and 2º will survive more time.

I said viruses have some similaritys with genetic programming. In the virus world, does it exist some selection process? Of course. However, this process is not a evaluation function or similar. A virus must,literally,survive. In virus world, antivirus, intelligent users,... are the selection process. Before, i said that obviously 2º program was better than 1º and nº.
A vxer writes virus with polimorphing, EPO, perprocess, ... becoz he want his virus survive so time as possible.
We have now two similaritys of virus programming and genetic programming. Virus generates descendants and only best programs survive.

## Mutation in virus

Im not referring to polimorphing or metamorphing. Again, we have a set of operational blocks.
We generate randomly a set of generation 0 programs and we take one:

```
gen0-> ChangeDir-ChangeDir-Infector-Payload-Infector-Payload-ChangeDir-Payload
```

We could program a infector block with ability of mutating current code.It could change a block by other or change order of blocks.For example, this infector-mutator block infects a file changing order of two blocks:

```
gen1-> ChangeDir-Infector-ChangeDir-Payload-Infector-Payload-ChangeDir-Payload
```

In gen1 virus has gain level becouse now it infects two directorys in one execution.
Now, virus of gen1 infects other file but in this time it change a block by other.

```
gen2-> ChangeDir-Infector-ChangeDir-Antidebug-Infector-Payload-ChangeDir-Payload
```

Of course, virus could infects a file with a bad combination of blocks:

```
other gen2-> ChangeDir-Payload-ChangeDir-Payload-Infector-Payload-ChangeDir-Payload
```

This other virus of gen2 will have less probability of surviving.

Note mutation must not occur always virus infects a file becouse if we have a good individual and always it infects it change its code,it will lose level. Virus must infect with mutation sometimes only.

## Crossing in virus

I think this could be very interesting, however it could be very umprobable of occuring too.
I was thinking a lot of about how could occur crossing here. For crossing, we need two proge-nitors, however we cannot have a list of all infected files. In addition, virus's descendants will infects others files, and others and others and its impossible to know what files are infected. I think crossing must occur when a virus try to infect a already infected file.
If a virus detects it's trying to infect a infected file, it will not infect it, however it could "learn" a block (randomly) from infected file and keep it to use it in next infection. In this manner, in the next generation, infected file will be a descendants from two progenitors.
In addition if a vxer in Jamaica and other vxer in Holland write two different virus but with a standart for recognizing blocks when the virus from Jamaica found a infected file with the virus from Holland it will learn blocks from virus from Holland. If a lot of vxers write their viruses with a standart, these viruses will learn from other viruses and more poweful viruses will survive becoz they will have the best blocks of all viruses.
I think this could seem difficult and umprobable of occuring...it seems for a film than for real life,however i think its not a bad idea :-m

## Problems

Of course, for this propose it's necessary a standart that will define blocks formats, identifiers, infection modes, etc... If i write a virus waiting it will find other virus for exchanging blocks, my virus must be able to recognize blocks from others virus. You will be thinking that if my virus can recognize a block from other viruses, antivirus heuristic can do it too.Yes :(

In addition, antivirus software could create software oriented to detect blocks
instead entire virus.
For example, we suppose a group of vxers use this method. They decide all virus that they will
create will use a common mark infection (For example they could set a field of header to a
value). With this mark all virus created could recognize a infected file (infected by them or
by other virus of the same "group") and then it could scan infected file searching blocks
to learn. However in the moment that a av knew that mark of infection is the mark used by
this group of programmers, then all virus created by these programmers will be easily detected.

## Outline
-------

I wrote a outline about how could be used this teory about genetic virus. This is only a
outline,an example that would have to be modified with a lot of changes for it works well.

### Outline of a block
------------------

A block is a functional element that can be executed in any offset, indepentdently of any
code out of block. A block uses functions implemented in its code and variables there too.
In the other hand, a block would have to be able to be identified. It's not necessary
to know exactly what block is it, however it's necessary to know it's a block and it can
be used by other genetic virus.
Blocks could have not variable size or we could put a field of the block for keeping size
of the block.I prefer second option.
I think blocks's format must be more simple as possible. A block must not have a big header
with a lot of characteristics of the block.
A example of that i think must be a block is this:

```
 -----------------
+  N byte of      +
+ initial code    +
 -----------------
+ 4 bytes for     +
+ size of code(X) +
 -----------------
+ X bytes of code +
+ of the block    +
 -----------------
```

N is a constant value for all blocks. In that code could be a small decryptor for the block
or simply a jump to the code of that block.
We could have problems becoz some blocks(for example a block that changes current directory)
will do changes that later these changes must be undo.
In addition we have not thought how virus will return to host.
I thought some ideas for this.
Structure of all blocks in a complete virus could be in this manner:

```
dd N_BCKs    (before all blocks there is a dd indication the total number of blocks)
---------

 BLOCK 1     (later,it will be the blocks)

---------

 BLOCK 2

---------

   ...
   ...
   ...

---------

 BLOCK n

---------
jmp host     (and finally a jump to the host)
---------
```

When an infector block infects a file with current blocks of its virus it must put after

all blocks a jump to the host.Ok,this is typical in all virus. However this jump has
a important role. This jump will be used by block to undo all changes they did.
Initially,before virus runs,that jmp point to host entry point. When a block needs undo
changes, it change that jmp for pointing to a part of its code where it will undo changes.
It will keep last direction where jmp jumped and when it undo changes it will jump to
that direction. It hooks that jmp. This method will be used by all blocks that need use it.
If a block hooks the jmp and later other block hooks the jmp again, it occurs in this manner:

Block1 hooks jmp:

```
-------------

  BLOCK 1

 undo code1: <---------
                       |
    ...              |
 jmp HOST EP           |
-------------          |
                       |
others blocks          |
                       |
-------------          |
jmp undo code1 --------
-------------
```

Blockn hooks jmp later BLOCK1 had hooked jmp:

```
-------------

  BLOCK 1

 undo code1: <--------------
                            |
    ...                     |
 jmp HOST EP                |
-------------               |
                            |
others blocks               |
                            |
-------------               |
                            |
  BLOCK N                   |
                            |
 undo codeN: <---------     |
                       |    |
    ...              |    |
 jmp undo code1 -------|----
-------------          |
                       |
others blocks          |
                       |
-------------          |
jmp undo codeN --------
-------------
```

In this manner all blocks will undo changes and first block that had hooked jmp will
jump to host entry point after all blocks had undo their changes.

--- x --- x --- x --- x --- x --- x --- x --- x --- x --- x --- x --- x ---

Well...after i had writed my article about genetic virus Kernel32 gave me this other doc
that he wrote. He hadnt seen my article and i was wonderful coz he thought very similar things
to that i wrote. I thought it was not fair that i sent my article for 29a when he had thought
very similar things than me so i send too this coz i think it's right.

[KERNEL32]

-----------------------------------------------------------------------------------------------

INVENTING A NEW TEQ

Ever thought about virii as Humans?
like virii, we all are living beings, right?
We all have our own DNA signature. So does virii.

I'll explain the idea,

THe "male" virus should be the one who gives the full signature :

* of a payload
* of certain Polyengine used by him
* of different functions
* etc

The "Female" virus should be the one wo recieves the DNAcode and add it to her code or replace her code with it.
One could always write a "DoubleSex" virus that recieves DNA and supplies it.

THe male should work like this:

Male

```
-------------------------
-                       -
-   Main virus program  -
-------------------------
-   Marker              -
-------------------------
-   Functions           -
-------------------------
```

Female

```
-------------------------
-                       -
-   Main virus program  -
-------------------------
-   Functions           -
-------------------------
```

First things first, the marker is very importend, it should contain the following:

* what type of files it infects (exe,dll, ocx, etc)
* what type of virus/worm it is (tsr, findfile, etc)

Basicly all the info that the female need to check if she will be compatible with the male.
You let the female take the functions that is not compatible with her, or she will crash.
The functions, should have alot of stuff like, Payloads, Anti-Debugging,
random number generators, etc.

The female should look for at the marker
and check if she is compatible with the male, and if so then she should decide what functions she will choose to add to herself.
simple hey, not quite. The male should supply how long the functions are too
and where the start/end of the function is.