Theme: Metamorphism

1. Disclamer

The followin' document is an education purpose only. Author isn't responsible for any misuse of the things written in this document.

2. Foreword

Why I wrote this article? It's simple. First polymorphic engines were coded becoz there weren't techniques to detect poly virus. In that time, AVs could only search for specific string in files and so detect known virus. Nowadays, when AV world knows heuristic analysis, there ain't same purpose of polymorphism existency as before first heuristic AV releasion. Before that, poly virus couldn't be detected by normal techniques. Now, known poly virus can be easily detected. Unknown poly virus can be also detected (with some problems, e.g. poly is more than one-layered). If not, virus will "get known" and will be detectable without any problems. Well, polymorphism is now only "standard", "nice feature" or "needness". But we (VXers) need to find another technique(s), that will let our virus survive for some months and make those months hot for AVers. We need to find some technique(s), that will paralise AVers for some those months. We need to find some technique(s), that will make AVers thinkin' about their product and make needness to rebuild it. Which technique(s) it will be?

3. Introduction to metamorphism

Perhaps it will be metamorphism. Metamorphism is something like polymorphism. Poly can generate variable decryptor, meta can generate variable whole code. Weird? Imagine meta engine like black box (X-D). Input will be code u wanna mutate and output will be mutated code, that will be different with input, but its functionality will stay same. If u wanna code metamorphic engine, that will be able to mutate strange code, u will probably get mad X-D. Really. U must be very smart to calculate ALL possible fails, u must be very patient and u must know a big amount of informations about your processor. How to code some meta engine? Aaah, u have luck, coz I'm writin' some meta. And becoz I'm still stayin' in learnin' process, I learnt more and more and that more I wanna discuss here.

4. Meta codin'

Codin' meta is very hard. In my opinion, its one of the hardest things on virus programmin'. However, I decided to code one meta engine. I think, every good meta engine should contain:

```
a)     Internal dissasembler
b)     Opcode shrinker
c)     Opcode expander
d)     Opcode swapper
e)     Relocator/recalculator
f)     Garbager
g)     Cleaner

ad a)  Internal dissasembler will disassemble instruction by instruction
ad b)  Opcode shrinker will shrink/optimize two (or more) instructions to one
ad c)  Opcode expander will expand one instruction to two (or more)
ad d)  Opcode swapper will swap two (or more) instructions, if it will be
       possible
ad e)  Relocator/recalculator will relocate/recalculate all relative
       references, such as jumps, calls and pointers
ad f)  Garbager will insert one (or more) do-nothing instruction(s) between
       real code
ad g)  Cleaner will clean garbage insterted by Garbager from code
```

Well, imagine, that everything described above has became truth and u have one kewl metamorphic engine for your virus. What will AVerz do? They won't be able to find scan string, coz really every generation will be different to other generation. Well, here comes only quality heuristic scanner. Scanner will search for suspicious code. If will scanner find it, it will report "unknown virus". If not, what will happen? What will AVerz say? Hmm, dunno... Let's wait for some mad, that will code fully meta-engine (maybe it will be me X-D), then we will see, what will happen. I'm really curious...

But back to reality. Now u probably know, why I said "one of the hardest things". Your meta should know every opcode and it should be able to process instructions pararely to provide shrinkin' and swappin' (shrinkin'/swappin' of code is much harder than expandin'). Very hard to code will be recalculator for all relative references. Jumps and calls to known (already mutated) code will be easy. Addresses will be only decremented by incrementation counter. Jumps and calls to unknown (not yet mutated) code will be harder. U will probably need to mark opcode (write address to table) and redundantly check, if code is already known and so if jump/call can be recalculated. Relocation of pointers will be probably solved by user defined table (its the ugliest effect - u will have to mark down all pointers). Garbager should use only limited set of instructions (probably NOPs only), coz garbage shouldn't affect other instructions. And finaly Opcode swapper. It should analyse every instruction and test, if its action won't affect second instruction. If not, instructions can be swapped, otherwise they can't.

That's not all. There r more problems and more limits, such as:

- U won't be able to mix variables with code (next tables?)
- U will need to keep somewhere original virus code (compressed?) or runtime recalculate all values in pointer-table (bring me an idea!)

Yeah, I said it will be hard. And have I ever said u, that the tiniest metas r about 20kB? X-D

5. Polymorphism vs. metamorphism

What is better? That's very frequently question. In my opinion, metamorphism is 100%-ly better. Whole code will be mutated (not only decryptor) and if u will add some polymorphic decryptor (better more-layered) to virus (to make static analysis harder), meta engine will mutate it too, so your babe will be both of poly and meta. Well, that virus will be absolutly mutant.

6. Closin'

Still alive? Yeah? So congratulation and if u r goin' to code your own meta engine, don't forget to contact me. I also hope, u will contact me, if u will bring me some new idea and also idea, how to solve problems with previous idea X-D. Don't give up!