A Discussion of Polymorphism

A Discussion of Polymorphism
by Gary M. Watson
31 May 1992

A polymorphic virus is a type of encrypted virus. Let's talk about those first. Many anti-virus programs rely on what we call a "scanner" which looks for an unusual sequence of machine language instructions or other unique data that indicates that a given virus is present. To defeat this, virus writers started encrypting their viruses by applying (for example) a random number exclusive-or'ed with the body of the virus. This obsfucates the unique string of bytes. So, programs like McAfee's scan had to do one of two things: look for the decryption routine (which cannot itself be encrypted since the 808x microprocessor would fail to execute it); or attempt to decrypt the body of the virus and look for the unique string of bytes in the body of the virus.

Well, to defeat even this, the virus writers created polymorphism. This basically involves a routine in the body of the virus that, on the fly, and during each new infection process, actually generates a new and unique decryption module and algorithm to be used in the new infection. The idea being twofold: make sure the decryption routine does not have a predictable sequence of bytes in it; and second, make sure that the body of the virus is encrypted with a sufficiently complex and unpredictable algorithm that cannot be decrypted by the anti-virus scanner.

So, what McAfee, Frisk and others did was to analyse the methods used by each polymorphic generator and then create an algorithm that detects the types of mutations created by each polymorphic virus out there. According to preliminary results from Vessilin Bontchev, McAfee's Scan 89B misses 0.04% of the mutations of "Fear", a typical virus that uses the Mutation Engine polymorphic object module; F-Prot misses 0.14%; and Dr. Solomon's FindVirus misses 7.86% of them. Bontchev considers anything less than 100% detection to be almost a complete failure, since any one of the missed samples could re-infect the system at a later date. The current release of Scan is 93 (as of this date), and it is unknown what its detection percentage is.

So, how do polymorphic viruses work? Well, let's constuct a pseudo code version of one way that it could be done. This will be similar to the way that the Mutation Engine works.

1. We wake up as as virus loaded into memory. No part of us is encrypted at this time. We consist of interrupt handlers for such things as the INT 21H ms-dos service request, etc.; a mutation routine; and a payload of whatever type.

2. We intercept a ms-dos INT 21h Open File request. We decide that this is a fine time to infect that file.

3. We call the mutation routine, which creates a unique encyptor and decryptor pair. The mutation routine then calls the newly created encryptor, which takes the memory image of the virus (remember, the virus is NOT encrypted while in memory), writes the decryptor out to the front of a data buffer in memory, followed by the encrypted copy of the virus. Keep in mind that the virus itself contains not only the traditional virus code, but also the routine that generates encryptor/decryptor pairs.

4. We infect the victim file using the memory buffer that contains the decryptor/encrypted virus combination.

5. We go ahead and perform the ms-dos function call that the user had requested.

Now, whenever the victim file is executed, ms-dos will jump to the beginning of the decryptor, which then decrypts the memory image of the encrypted virus and transfers control to the now decrypted virus, which executes normally.

Most people do not understand how the mutation routine iteslf generates the necessary unique encryptor/decryptor pair. The method used by the Mutation Engine is extremely complex, but a simpler one can be devised for demonstration purposes, and the difference will be only in complexity, not content.

Consider the sample encryptor/decryptor pair shown below, side by side for ease of comparison. For ease of human understanding, it is presented as if the mutation routine was creating assembly code. In reality, it merely emits binary machine code which is stored directly in memory.

Note: it is my considered opinion that the following would be a fairly weak polymorph if it were implemented, that is, McAfee et. al. would have little difficulty detecting this. Therefore, I feel that presenting this example in a public forum poses no significant danger and has potential to clear up much of the misunderstanding surrounding this issue.

Note that this very simple polymorphic example has few predictable byte strings in it, since one does not know which pointer register is to be used, the contents of any of the random numbers aa through jj, nor how many bytes the routine will choose to garble in one iteration. Note that by changing these factors, an extremely complex encryption takes place, as both the algorithm and the keying data cange considerably. Even the *quantity* of keying data changes from one infected file to the next.

The strength of the polymorphic virus is in how thoroughly it mutates, that is, in how unpredictable the sequence of instructions is in the decryptor module. In particular, no more than about 4 bytes can be predictable, else a simple scan string could be constructed and then all the polymorphic code would be a waste of effort on the part of the virus writer.

(By the way, if I have made any typos in the above code examples, *GOOD*. Pretend I did it on purpose to confuse any beginning virus writer who tried to use the code as is. Of course, I did not really show the code necessary to implement the polymorph, I only showed what it could do.)