

Winlamer virus

Igor G. Muttik

Virus Researcher, S&S International PLC

The first polymorphic viruses appeared long ago. We even saw many different polymorphic construction sets and engines available in sources and as object files. It is no wonder to see nowadays a heavily polymorphic virus based on either one of these engines or even using its own polymorphic generator. Some of these polymorphic viruses are really very variable and it is difficult to detect them reliably. But until appearance of Winlamer virus the polymorphic viruses infected only normal DOS executables. But why?

There is actually a problem, which explains why the first polymorphic virus for Windows appeared only now (apart from the fact that implementation of a virus infecting a Windows program is much more difficult than writing a COM-file or boot sector infector). The problem is that under Windows the code of a program is write protected. So, the program cannot modify itself and self-decrypting code (like polymorphic virus decrypting itself) cannot exist at all. For the first sight it might seem that polymorphic virus under Windows is simply impossible.

Winlamer virus is interesting because it overcame this problem and became the first polymorphic virus for Windows.

Execution of the infected file

Winlamer is a direct-action (non-resident) virus, infecting executable files in the NE-EXE format (this is a format of 16-bit Windows applications; almost all Windows 3.1 executable files have this format).

On execution of any file infected by Winlamer virus the control is passed to the decryption routine. Winlamer, like any polymorphic virus is encrypted and to get to the virus body the polymorphic code should decrypt it first. This implies that the virus must have an ability to modify its own code, but under Windows code is write protected when the program gets the control. To get a right to modify the program Winlamer uses simple and obvious method -- it issues an Application Program Interface call (DPMI/Windows API: AX=000Ah/Int_31h), which duplicates the code segment selector to AX register. Then the virus assigns obtained value to a data segment selector (MOV DS,AX). And now for this data segment there is no more restriction to modify its contents. This API call is one of the first actions done by the polymorphic decryptor. This call is dissolved in the meaningless garbage commands of the decryptor and it is issued before the virus has decrypted even one byte of its body. When the virus gets the write access to the encrypted body it starts to decrypt itself as a normal polymorphic virus. So, the decryption loop follows mentioned API call in the polymorphic decryptor. Method used by

Winlamer to encrypt and decrypt itself is a simple XOR [BX],keybyte instruction.

When the body of the virus is decrypted, it gets the control. First the contents of DS register is restored. Second Winlamer issues the call to check whether DPML is loaded (AX=1686h/Int_2Fh). That might seem strange as the virus has already used one of the DPML services and Windows 3.1 has built-in DPML driver, so under any circumstances the DPML driver should be available. If the virus detects that DPML is not responding -- it just passes the control to a host file. We see that this verification is merely to check whether the program is run under true Windows or under some sort of artificial environment (debugger or emulator). In the latter case virus will not replicate at all.

Then the virus allocates memory and sets necessary rights to access it (again using DPML services). The remaining actions of Winlamer virus are very similar to what normal direct-action DOS viruses do: get DTA (disk transfer address), get and save current directory, set \WINDOWS to be a current directory and issue findfirst/findnext (AH=4Eh,4Fh/ Int_21h) calls until all victims are infected. The only thing worth mentioning is that Winlamer does not check whether \WINDOWS directory really exists. So, Winlamer will not infect any file if you changed the default name of this directory when first installing Windows on your computer.

The virus looks for all *.EXE files with the 'NE' signature in the \WINDOWS directory and infects them. It does not infect any executables with other extensions (ex., screensavers in *.SCR,

etc.). Winlamer tries to infect all NE-EXE files in \WINDOWS directory in one run and that takes quite a noticeable time. Finally, when all victims are infected the control is passed to the host file.

Winlamer appends its body at the end of the victim file and sets necessary entries in the NE-header to get the control when the program is started. File size change is variable but typically it is about 2000-2100 bytes. Infected files are marked by adding 100 years to a current file time stamp. Unfortunately this is not visible under DOS (two high digits of the year are not shown by DOS's "dir" command as well as by most of other disk utilities).

Virus internals

Winlamer virus has got no payload.

The virus was written by a quite productive virus author -- someone who calls himself "Burglar" from Taiwan. At least the virus carries the following strings:

```
"Winlamer2 (C)Copyright Aug, 1995 by Burglar in Taipei."
```

And one more:

```
"PME for Windows v0.00 (C) Jul 1995 By Burglar"
```

Burglar is rather prolific -- he is also the author of some other sophisticated viruses. In June 1995 (just two months before the appearance of Winlamer) he wrote Wintiny.741 -- non-polymorphic direct-action NE-EXE infector. And Winlamer has a great code similarity with Wintiny -- infection routine is nearly identical. In fact Winlamer could be called a polymorphic variant of Wintiny.

Burglar also made the Phantasie Mutation Engine (PME) -- a polymorphic generator used both in his DOS viruses (PME.3788 or PME.Burglar) and in Winlamer. So, it took him two months to evolve from normal direct-action NE-EXE infector to a polymorphic one. We see that he worked quite intensively, even in the summer!

The Polymorphic Engine

The polymorphic engine of Winlamer virus (PME for Windows) is based on the PME for DOS, but it is much more simple.

As almost all polymorphic engines it has a built-in random number generator (RNG). Winlamer's RNG is based on the series of reads from the i/o port 40h (timer). The virus reads 3 times from the port, then XORs two last values and RCRs the result using the first value as a shift counter. The RNG returns random value in AL register.

The polymorphic engine of Winlamer is using a set of tables of subroutines (other possible approach is to base the engine on the tables of processor op-codes).

Winlamer's engine just "dissolves" the basic decryption routine in the garbage commands. This represents not the highest level of a polymorphic code. Most of the garbage commands generated by Winlamer's polymorphic engine are one-byte (just five processor commands are used: NOP/REP/REPNE/CLD/STD), some are two-bytes (XOR/OR/AND/ADD/ADC/SUB/SBB/CMP reg,reg) and some are 4-bytes (MOV/OR/ADD/ADC/SUB/SBB/CMP reg,[const]). Three-byte commands are not generated by Winlamer at all. When you find such command in the polymorphic code you will see that they belong to the decryption routine (ex., MOV AX,const; MOV BX,const; XOR byte [BX],keybyte; ADD BX,4).

Winlamer uses an elegant method to increase the contents of BX register in the decryption loop (instead of obvious INC BX). It executes two commands: NOT BX and NEG BX to achieve this. For the first sight the purpose of them might be unclear, especially when they are separated by a polymorphic garbage. This was undoubtedly used to add a bit of obscurity to a decryptor.

Conclusion

Winlamer virus has not been yet reported in the wild. Fortunately, the virus is not capable to infect both normal DOS EXE files and NE-EXE. In this case it would be much more virulent.

Obviously, Winlamer is an experimental virus. It might be the reason why it has got no payload and its replication is too noticeable. Most likely it was written just to prove the point that polymorphic is possible for Windows viruses.

I have to admit that I was pleased to see that this attempt was quite successful (unlike, say, the case with the 32-bit Windows-95 Boza "virus", which was so buggy that it is usually classified as intended). Winlamer looks very neat, accurately coded and elegant.

Now let's recall that we have already Winsurf virus, which is a memory resident Windows infector. How much time it will take for virus authors to mix these two techniques and create resident polymorphic viruses for Windows and also multipartite viruses, infecting DOS executables, Windows and Windows-95 applications? I believe they will appear soon, because all necessary approaches are already tested separately in this or that virus. The problem is only to combine known techniques. And that should not take much time...

Winlamer

Aliases: Winlamer2, WIN:Lame

Type: Direct action NE-EXE file infector,

polymorphic.

Infection: *.EXE ('MZ' and 'NE' only)

Self-recognition in files:

Year of last modification ≥ 2000

(adds 100 to a file timestamp)

Hex Pattern: No simple pattern in files (virus is polymorphic)

Virus is not resident -- no memory search pattern.

Trigger: None

Payload: None

Removal: Use backups or reinstall the files from the original diskettes.