Computer Viruses (part 2 of 4)

This article last reviewed: 15 April 1988

TECHNICAL ISSUES

How Viruses Propagate

Viruses can propagate by a variety of methods. The most common way for a Macintosh virus to replicate itself is to have an INIT that installs a background (VBL) task that checks for specific occurrences, such as a disk insertion, and then copies itself somewhere to that disk.

VBL Tasks

The Macintosh has always had a limited form of background processing available to it through the use of the Vertical BLanking queue. Every time the screen on a Macintosh (except for a Macintosh II) is refreshed, any routines installed in the queue are executed. The Macintosh II has a dummy VBL queue for compatibility reasons since the advent of a variety of screens has led to different vertical retrace periods.

VBL tasks can be installed in the queue by any program. The program has to load a routine into a section of memory and install the routine into the VBL queue by calling the Vinstall ROM routine. It is the responsibility of the installing program to make sure the segment of memory containing the routine remains available even after the program has exited. Each VBL task has a specified time period it should be left "asleep" before it is called. Every time the routine is executed, a counter is decremented for that routine. When that counter reaches zero, the routine is deleted from the queue unless the routine itself resets the counter.

Lengthy VBL tasks such as the one that might be used to replicate a virus can interfere with the normal operation of the Macintosh by interrupting processes that shouldn't be interrupted. A perfect example of this is printing to a LaserWriter over an AppleTalk network. If a VBL task takes too long in its execution, the printing process could terminate abnormally and leave the machine's connection to the network in an unstable state.

For the purposes of a virus, an INIT is most likely to be the culprit responsible for installing a VBL task.

INITs

INITs are routines that are run when the Macintosh is booted. For the most part, they have full access to all of the commands normally available to a standard Macintosh program. The major difference is that the low memory globals have not been set up yet, so any INIT needing access to structures normally stored in low memory must create its own.

INITs in the System file:

When a Macintosh boots, the INITs in the System file in the "blessed" folder are the first code to be executed. These INITs should generally be Apple INITs only -- any non-Apple INITs should be considered suspect.

The INIT 31 mechanism:

A special INIT in the System file, INIT 31, was created to allow for the execution of non-Apple INITs without having them installed in the System file itself. When all of the other INITs in the System file have been executed, INIT 31 walks through the System folder looking for files of types INIT, RDEV, cdev, and executes any INIT resources it finds in these files. The order in which the files get loaded is alphabetical. Needless to say, a simple way for hiding parts of a virus is to drop INITs into legitimate files already existing in the System folder with these file types.

CDEVs

The file type cdev indicates a file containing a Control Panel device. When the Control Panel is loaded, it walks through the directory of the System folder looking for any files of type 'cdev'. When it finds a file of this type, it loads the ICN# of that file (assuming it has one) into the list of icons shown on the left side of the Control Panel. When you click on the icon of the cdev in the Control Panel, the code in the cdev resource in the file of type 'cdev' is executed. A virus could easily use this mechanism as a way to infect a system, install a VBL task, etc.

Many cdev files have INITs in them with the cdev controlling the settings that the INIT will use when it is installed. A good example of this is the settings for a screen blanker. The INIT actually installs the VBL task, but the cdev controls when dimming occurs. None of the standard Apple system cdev files have INITs in them, but there is nothing to prevent a virus installing an INIT in these files as a way of hiding its code.

DRVRs

DRVR resources typically can have one of two functions: they can be the code for a desk accessory, or the code for drivers necessary for the system to perform some function such as printing. Once again, the key word here is 'code'. Whenever code is involved, the potential arises for the perpetrator of a virus to take advantage of it.

Just as with cdevs, when a DRVR gets opened, either by the choosing of a desk accessory or by the system, code is executed at that point. This is the stage at which a virus might fulfill its purpose.

CODE Resources

Each application has at least two CODE resources. The first of these CODE resources has an id of 0 and contains what is known as the jump table. This table provides the basic information necessary for various parts of a program to call routines in other CODE segments. The current rage in viruses is to modify the CODE ID = 0 resource of an application so that a CODE segment it installs in the application gets called before the application is actually run. This CODE segment could go out and check if the virus has infected the current system, and if it hasn't, install itself. All the perpetrator of a virus has to do at this point is upload a copy of an infected application to a BBS, and it spreads across the world.

Applications that allow external procedures:

Viruses could take advantage of the external procedures that are allowed by some applications. The perfect example of this is HyperCard, with its XCMDs and XFCNs. This is how the MacMag virus was transmitted.

Keywords: <None>

==========

This information is from the Apple Technical Information Library.

19960215 11:05:19.00
Tech Info Library Article Number: 2822