

Preventing and Detecting Malware Installations on NT/2K

H. Carvey 2001-10-24

Preventing and Detecting Malware Installations on NT/2K

by *H. Carvey, CISSP*

last updated October 24, 2001

Introduction

Recent history has seen the release of several Trojans, worms, and viruses (collectively known as malicious software or malware) specifically targeting Windows NT and Windows 2000 (2K). For the most part, much of the malware has taken advantage of long-since patched vulnerabilities. For example, the sadmind/IIS worm exploited the directory transversal vulnerability in IIS, which had been patched in Nov 2000. In fact, it seems as if many of the recent worms - including sadmind/IIS, Code Red, and Code Blue - were released as a wake-up call to encourage administrators to patch their systems. After all, Code Red was noisy, but it did very little damage to the systems it infected.

For many administrators, protecting malware is a constant race to catch up with malware authors. Often, the release of a new piece of malware, be it a worm, virus, or Trojan targeted at some vulnerability in a Microsoft product follows the publication of an advisory that identifies the vulnerability. Generally, this means that administrators have a narrow window of opportunity in which to download the patch, test it, and roll it out to production systems and across the enterprise. Many times, administrators are not able to complete the test-roll out cycle before the malware proliferates. So what can administrators do to protect their systems from malware installations or, at the very least, detect them when they occur?

The purpose of this article is to address actions that NT/2K administrators can take to prevent and detect malware installations on their systems. These actions go a step beyond configuring application settings (such as disabling script mappings in IIS) and installing patches. By taking advantage of the inherent capabilities of the operating system itself, administrators can prevent or significantly hamper malware installations, and detect when such installations occur.

How Systems Become Infected

In order to protect systems from malware, administrators first need to understand how the malware gets installed on a system. Exploiting Web server and e-mail client vulnerabilities are popular methods of infecting systems, as is relying on unwary users to run an arbitrary

executables or scripts that are received from a friend or other 'trusted' sources. Once these files execute, many of them generate a footprint on the system by writing files to certain locations and creating or modifying Registry keys. Administrators can use this activity as a roadmap for preventing the malware from being installed in the first place, and to detect when such an installation has been attempted.

Malware such as the *sadmind*/IIS worm copy the command interpreter (*cmd.exe*) to */scripts/root.exe* in the webroot. Some viruses create or delete various files. Trojans copy files to the hard drive and, in order to remain persistent across reboots and user logins, they create entries in certain Registry keys. Administrators can take advantage of discretionary access control lists (DACs) to protect files and Registry keys from modification, and system access control lists (SACLs) to log when attempts to access those resources occur.

Protecting the System

Administrators need to keep in mind that many Trojans and viruses, as well as some worms, are installed in the security context of the user account under which they are executed. This means that the installation process can only access those resources that the particular user context is able to access. This is the key to preventing malware installations. If the user context does not have 'write permission' to specific directories or permissions to create or modify certain Registry keys the malware needs to access in order to install and execute, then the malware will be prevented from installing, or at the very least the installation will be crippled. For example, the [Potok worm](#) attempts to create a new user on the infected system. However, as is stated in the technical description of the worm, if the user context under which the worm is running does not have sufficient privileges to add a user, this functionality is inhibited.

One of the challenges confronting security administrators is to determine when an infection has take place and, if so, what damage the infection may have caused. There are two methods by which administrators can view the changes that malware makes to a system during installation. The first method is to actually install the malware, such as a Trojan, while using an installation monitor program such as [InControl5](#). These programs take a pre-installation snapshot of the system, and compare it to the system after the malware installation. This way all changes to the system, including file and Registry key creations and modifications are identified. The administrator can then go about cleaning the system.

The second method is much simpler. Antivirus software vendors such as [F-Secure](#) and [Symantec](#) maintain malware encyclopedias, providing information regarding what changes a particular worm,

Trojan, or virus makes to the system it infects. Other sites, such as Carnegie Mellon's [CERT/CC](#) and the U. S. Department of Energy's [CIAC](#) also provide a wealth of information regarding various malware. Administrators can use the information available from these sites to protect their NT/2K systems from infection.

For example, Trojans such as Back Orifice 2000 and NetBus need some method of remaining persistent across reboots and logins. A simple means of doing this is for the installation procedure to create an executable file on the hard drive (very often in the winnt\system32 directory) and add an entry to the following Registry key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
```

Other Registry keys that are particularly popular for hiding pointers to include:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
HKEY_CLASSES_ROOT\exefile\shell\open\command
```

Technical information for the [Vote worm](#) indicates that the worm makes an addition to the Run key, pointing to a Visual Basic script file that the worm creates in the Windows directory. The [Magistr virus](#) adds a reference to an infected file in the same Registry key. Therefore, if the administrator configures his systems so that users cannot create files in the winnt\system32 directory, or create sub-keys or set values in the above listed Registry keys, many Trojans, and some other forms of malware, will be prevented from completely installing.

Many times during a malware installation (specifically a Trojan installation), additional files may be added to the system. For example, [NTSecurity.nu](#) provides a fake GINA.DLL that can be used to capture passwords when users log into a system. In order to prevent a malicious user (or a Trojan executed by an unsuspecting user) from employing this DLL in order to escalate privileges or gain access to other systems, administrators should be sure to use strong passwords (enforced through means inherent to the system). They should also prevent users from creating subkeys to the following Registry key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion
\WinLogon
```

The default security settings for this Registry key on Win2K Professional are that Users have only Read access.

Rootkits are yet another issue that administrators must be prepared to deal with, as such things will become more popular for the NT/2K platforms, corresponding to the development of more sophisticated Trojans and worms. Administrators should take the necessary precautions to prevent malicious users from installing a rootkit by setting DACLs accordingly on files, directories, and Registry keys. For example, the rootkit instructions state that once the package is downloaded and unzipped, the administrator can type `deploy` to install that rootkit, and then use the `'net start'` and `'net stop'` commands to manage the functionality of the rootkit. Since this particular rootkit operates as a service, administrators should prevent users from creating subkeys to the following Registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
```

Once the necessary DACL is in place, the administrator should then set the SACLs to ensure that failed attempts to create or modify subkeys are recorded in the Security EventLog.

Detecting An Installation Attempt

Once administrators have configured DACLs on files, directories and Registry keys to prevent users from writing to these objects, they can then set SACLs on those objects to audit failed attempts to do so. This requires that the administrator first ensure that auditing is enabled, and then ensure that at least failure events for File and Object Access are audited. When doing so, however, administrators should be aware that the default settings for EventLog size should be modified to accommodate the amount of information that will now be logged. The default setting of 512KB is often inadequate to allow events to be logged without some events being overwritten. If EventLog entries are overwritten, important data regarding an incident may be lost. Administrators should also ensure that the EventLog files themselves are adequately protected from modification or deletion.

Administrators may choose to automate the application of the necessary settings for several systems from a central location. [Perl](#) is a very powerful tool with a great deal of functionality. [Dave Roth's Win32::Perms](#) module will allow the administrator to set and verify both DACLs and SACLs remotely. The [Win32::Lanman](#) module will allow the administrator to set and verify auditing functions as well as collect EventLog entries, and the [Win32::TieRegistry](#) module will allow the administrator to set and verify the various Registry settings that apply to the EventLogs, such as size, etc.

Once the DACLs and SACLs have been configured appropriately, attempts to write to the protected resources will cause failure audit events to be generated in the Security EventLog with EventIDs of 560. For example, attempts to modify a protected Registry key will generate events such as the following:

```
Computer:      MUSASHI
Category:     3 (Object Access)
Event ID:     560
EventType:    16 (Failure Audit)
Source:       Security
SourceName:   Security
Generated:    Fri Apr 21 22:01:17 2000
Written:      Fri Apr 21 22:01:18 2000
Flags:        0
User:         MUSASHI\admin
```

Description:

```
+Security
+Key
+\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
+-
+0
+112205
+2210750976
+admin
+MUSASHI
+( 0x0, 0x1847C)
+-
+-
+-
+%%4434
+-
```

Note: The above EventLog entry was retrieved using a Perl script.

Attempts by a user to write to a protected directory appear in a similar format, and differ only in the information contained in the 'Description' section.

Administrators can monitor systems more rigorously by enabling auditing for both success and failure events for Process Tracking. Auditing of these events will provide the administrator with a list of the processes that were created on the system and which user attempted to create them. When a new process is created, an event with ID 592 is generated, with corresponding information regarding the process ID and username. An event with ID 593 is generated when a process exits.

Using this information, administrators can not only track what processes were created and by whom, but also how long those processes were run.

Monitoring

Once the appropriate settings have been made and the DACLs and SACLs are in place, the EventLogs will contain meaningful information. Administrators should develop some means of automating the collection and analysis of this information as much as possible. The EventLog entries can be collected and managed locally by using EventLog collection tools such as [WdumpEvt](#), [dumpevt.pl](#), or [DumpEl.exe](#) from the NT Resource Kit. Another means of centralized collection of EventLog information includes using tools such as [BackLog](#) for converting NT/2K EventLog to syslog, and [SL4NT](#) or Kiwi Enterprise's [Syslog Daemon](#) for collecting the information. Another possible solution can be found at [MonitorWare](#).

Analysis of the collected EventLog information can be automated through the use of Perl scripts that look for particular items in the Eventlogs. A small modicum of programming skill can be used to look for specific events, patterns of events, or even develop trending information based on a variety of parameters. Such analysis will be specific to the particular enterprise from which the audit information is drawn.

Administrators should also develop some automated means for ensuring policy compliance in their systems. Some means must be established for collecting specific information regarding the security posture of the NT/2K systems they manage, and verifying that changes haven't been made to these settings. Perl scripts utilizing the previously mentioned modules will not only allow an administrator to apply the necessary settings, but also scan those settings on a regular basis in order to verify that they haven't been altered. For example, using the [Win32::Perms](#) module, the administrator can not only set DACLs and SACLs on particular files, but also verify that those settings have not been altered. The same module can be used to determine file ownership, if necessary. The [Win32::AdminMisc](#) module can be used to retrieve author and version information from the resource section of executables and DLLs, as demonstrated in [finfo.pl](#). [Mac.pl](#) can be used to record the last modification, last access, and creation times of files. Using these modules, as well as [Digest::MD5](#), administrators can produce single script that performs comprehensive file integrity verification similar to Tripwire. This script can then be run at regular intervals to verify security policy compliance. [Fsw.pl](#) demonstrates the use of a real-time file system monitor script that can alert the administrator when a new file has been added to the `winnt\system32` directory.

Conclusion

In addition to keeping Service Packs and hotfixes up to date on NT/2K systems, administrators and their managers should develop the appropriate security policies and procedures that pertain to configuring and monitoring their NT/2K systems. Automating policy compliance verification, EventLog collection, and audit data reduction and analysis will assist the administrators in managing the enterprise. For meaningful information to be recorded by audit sources, administrators should configure the DACLs and SACLs on specific objects, particularly those accessed during malware installation. Doing so will not only prevent the malware installation from succeeding, but also provide information to the administrator indicating that a malware installation was attempted.

[Privacy Statement](#)

Copyright 2006, SecurityFocus