

CLEANING UP THE MESS – TIME TO REDEFINE ‘DISINFECTION’?

Gergely Erdélyi

F-Secure Corporation, Tammasaarenkatu 7, PL 24, 00180, Helsinki Finland
Tel +358 9 2520 0700 • Fax +358 9 2520 5001 • Email Gergely.Erdelyi@F-Secure.com

ABSTRACT

The meaning of the term ‘disinfection’ has changed during recent years. Today’s increasingly complex viruses often introduce rather complex changes to the system configuration. These changes are made to achieve certain goals, or simply as a side-effect of the infection. File disinfection alone is no longer enough in most cases.

In certain cases the system becomes completely unusable if the malware is removed without reverting its modifications first. Sometimes these changes don’t prevent the system from working but it might take a long time to revert them manually.

This paper elaborates on the techniques used by viruses and the counter steps today’s anti-virus applications have to take to clean the system up properly. The paper also discusses the new features anti-virus programs must have to be able to fight today’s and the possible future infections.

1. INTRODUCTION

Disinfection of computers from viral infection has been a common practice from the very beginning of anti-virus history. However, the meaning of the term and the actions required to remove an infection with all the intentional system modifications and side-effects has changed recently. Recent malware more and more often employs techniques to make the disinfection of the system as hard as possible. This kind of malware will be called ‘hard-to-remove’ malware and infection throughout this paper.

When a hard-to-remove infection has to be disinfected there are two ways out. Either the system and all the applications have to be reinstalled and reconfigured or the infection has to be removed and all the changes made by the malware during the infection reverted. Although the first option is always safe, it is not always practical. Real world sites with thousands of infected computers would require enormous amounts of human work from the system administrators if all the computers had to be reinstalled from scratch. The second option, being more applicable to the real world, is the main topic of this paper.

2. THE PAST

Hard-to-remove viruses are not at all new. There have been several cases in the past when the removal of certain viruses was very difficult and required the development and distribution of a dedicated removal tool. The only thing that made the situation a little better compared to these days is the fact that the older viruses were slower in spreading, especially globally speaking.

2.1 One_Half

The One_Half virus started to spread in May of 1994. One_Half is a multipartite, stealth, polymorphic virus that infects COM and EXE files as well as boot sectors of hard drives. One feature that made One_Half somewhat unique is the boot sector infector and stealth code. When the virus is started from an infected hard drive boot sector it becomes resident and installs its own disk access routine to the system. Using this routine it can hide the infection in the boot sector so that whenever an application tries to read the infected sectors the virus shows the original clean sectors instead. The very same routine is responsible for most of the damage One_Half caused. Every time the computer boots One_Half encrypts two cylinders on the hard drive moving backward from the end of the drive. The encryption is transparent to the applications since the virus decrypts the data on the fly using its stealth routines. When it has encrypted half of the hard drive cylinders it reboots the computer and displays the following message:

• **Dis is one half.**

Press any key to continue ...

There is no data loss until something tries to access the disk using port level access or the virus is removed without first decrypting the encrypted data. Repairing the Master Boot Record is quite easy using the ‘fdisk /mbr’ command from a clean DOS boot floppy. The unfortunate side-effect of this method is that it removes the virus and also the encryption key that is embedded into the virus body. This sort of removal results in a serious data loss. The recovery is not impossible but

it created a lot of work for data recovery specialists around that time. The data is even harder to recover when the virus infects the system several times and it is removed without decryption while certain parts of the hard drive were encrypted several times. [1]

2.2 DIR-II

DIR-II appeared in 1991 and employed a new technique for infection. Instead of infecting the files themselves, DIR-II modifies the File Allocation Table so that the first cluster of all the infected files points to the DIR-II infected cluster. The original cluster pointer is stored in an unused area of the cluster entry in scrambled format. If the virus is not active when the operating system tries to access the files it reads the virus body instead of the file data. If the virus is resident it descrambles the original pointer and reads the right cluster for the operating system. If the virus is resident the infected files look normal and it is possible to make backup copies of them.

There are two ways of removing DIR-II:

- While the virus is active the infected files can be renamed to some extension other than COM and EXE. Since DIR-II infects EXE and COM files only it will not infect those files after renaming. After the DIR-II has been cleaned from the disk the files can be renamed to their original extension.
- The cluster pointers have to be restored from the scrambled form and written back to the File Allocation Table. This method works also if the virus has been removed already. From the way this virus infects files it's clear that simple removal of the virus without reverting the changes results in massive data loss. In this particular case the virus can be removed without a special tool but preparations are required before the removal [2, 3].

3. SYSTEM AREAS AFFECTED BY MALWARE

3.1 File system

File system is one area that will be modified in some way when an infection takes place. There are rare exceptions like the CodeRed worm that runs in memory and does not reach the disk but the rest of the malware introduces some sort of modification. The most obvious change in the file system is the actual infection. In case of file infectors the infection can affect just about any file in the system the malware is capable of infecting. The cleaning of these files can be accomplished using the traditional removal methods implemented in all anti-virus programs. In the traditional approach the removal affects one file only which might not be enough for the complete removal of certain viruses. Companion viruses are good example of this and so are others that copy the host file content to another file. In these cases the original file has to be restored that might require even complex operations; for instance restoration of the original data from a scrambled file.

With stand-alone malware, the infection usually does not affect existing host files. The malware

is simply copied to a common directory in the system then referenced somewhere (e.g. Registry) so it will be started automatically. The most common places are:

- Windows directory
- Windows System directory
- Recycle Bin directory
- User’s Startup folder in Start Menu

Removal of stand-alone malware can be most often accomplished by deleting the malware from the place where it installed itself.

Low-level file system modifications are no longer common. The reason for this might be the diversity and complexity of the standard *Windows* file systems. There are many different variations in use:

- FAT12
- FAT16
- FAT32
- VFAT extension for the FAT systems
- Several versions of NTFS

Implementing low-level handling routines for these file systems is definitely not trivial and the lack of low-level documentation makes it even harder. Because of this it is not likely that many of the future malware would include effective low-level file system modification routines. However, we cannot completely outrule the possibility that some malware will appear with built-in file system drivers.

The W32/Stream virus uses an otherwise rarely used feature: streams. On NTFS any file (or even directory) can have several named streams and the file content is actually the first, unnamed stream of the file. W32/Stream uses this feature by copying the host to an alternative stream using the name ‘FILENAME.EXT: STR’ and replacing the file itself with the virus code. As long as the file stays on NTFS the virus can access the original file but the content is lost if the file is copied to a file system that does not support streams. W32/Stream is a special case of companion viruses that uses file system features to make the infection harder to notice. [4]

Since virtually no anti-virus software supports scanning of NTFS streams yet, W32/Stream proves that even the filesystem – which is a widely explored area – might hide surprises.

3.2 Registry

On modern *Windows* systems most of the system and application configuration data is kept in the Windows Registry. Since it contains essential configuration information it is often a target for modifications. The most often modified keys are the so-called run keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
```

HKEY_LOCAL_MACHINE can be substituted with HKEY_CURRENT_USER, the branch which holds the configuration data for the user currently logged in. By adding itself under the run keys the malware can make sure that it will be started when *Windows* starts or the user logs in.

Since the Registry holds almost all the important configuration details about the computer it is possible to severely cripple the system or render it completely unusable.

One example of this is the Aula Trojan. Aula modifies the system configuration so that the user will not be able to

- shut the system down
- access the Registry
- run old DOS applications
- access the task manager
- change password
- change system configuration (using the Control Panel)

From the above list it's clear that this Trojan affects the system usability so severely that it becomes almost unusable after the infection. The recovery from this situation is not easy since many operations needed for the easy recovery are disabled. [5]

The main problem with the Registry is that the Registry data is stored in a quite complex, almost completely undocumented binary format. In practice this means that it can be fixed only by using the standard *Windows* Registry access API. This assumes that the system is running and fully functional, which might be limiting factor in a system that has been heavily modified by the malware.

NT systems have strict control over user access rights so the Registry handling routines need to be able to cope with that. If the user account under which the anti-virus is running is different than the user logged in, additional steps are required to access the user specific branches in the registry.

3.3 Configuration files

Even though most of the system configuration data is stored in the Registry since *Windows 95* there are still several configuration files in use. These are text-based files that are relatively easy to modify. The most important text-based configuration files still in use are:

- autoexec.bat
- config.sys
- win.ini
- system.ini

These files are mostly used on *Windows 9x*-based systems. For compatibility reasons they exist on *NT*-based systems as well. The autoexec.bat is a exception since it is only partially supported under *NT*. *NT* parses only the environment variables from autoexec.bat for backward-compatibil-

ity reasons. No other commands or file calls are supported. This limits the achievable impact of modification but it might be enough for a companion virus trick for example.

Even though the amount of configuration information in these configuration files is much smaller than that in the Registry it is still possible modify them so that it would benefit the malware in some way. Good examples are the ‘run=’ key in win.ini or the ‘shell=’ key in system.ini that can be used to start the malware. Autoexec.bat and config.sys are more or less free-form text files so they require standard text processing tools/functions to modify. The INI files are in the *Windows* INI format which can be processed using function calls provided by the Win32 API.

3.4 Security configuration

In recent malware we often see routines that try to weaken the security on the computer by changing certain settings. The easiest and most common way of opening the computer is to enable file sharing for everyone with no password. This opens the machine up completely, making it possible for anyone who has network access to the machine to take full control over it. Drive sharing is not the only option, there are many different ways for leaving open doors for an attacker.

CodeRedII, for example, drops a simple Trojan that modifies the *Internet Information Server* (IIS) settings so that the C: and D: drives are accessible as /c and /d through the web server. It also copies the standard cmd.exe to the web server’s script directory and marks it executable in the IIS settings. This way anyone with access to the machine’s web server has full control over the machine, can upload/download files and execute arbitrary commands. CodeRedII works on NT-based machines only with IIS installed. [6]

Disinfection routines must be able to restore the changed security settings to sensible default values. Defining the ‘sensible default values’ might not be easy. If the malware destroys the original settings (which is the most common case) the most secure approach is to place restrictions on the access by default. However, this is not always what the user or the system administrator wants, so human intervention is needed. If the interaction is not convenient the user must be presented with the list of the changes made to the system during the disinfection procedure. The security fixing process usually includes

- reset of user account and group permissions
- removal offending user accounts and groups
- removal of unwanted drive shares
- removal of unwanted file sharing through web server
- reset of web server access control settings
- etc.

3.5 Processes

Assuming a running system with an active malware in the memory certain functions providing access to the running processes might be required. The first problem about running processes comes from the *Windows* file locking mechanism. The image of any process cannot be removed

as long as the process is in the memory. There are different ways of overcoming this problem.

There is a mechanism that uses an INI file ('wininit.ini' on 9x systems) or the Registry (*NT* systems) for removal or renaming files at the next reboot. The shortcoming of this method is that either

- all the removal steps must be performed before the reboot which might not be possible if the malware is active

or

- the disinfection process has to be performed in several steps with a reboot in the middle which makes it a bit more difficult to coordinate.

On *NT* it is possible to rename the file in question so it can not be loaded at the next reboot but that assumes a reboot in the middle of the removal process.

The most convenient method is to kill the offending process. The process in a *Windows* system can be enumerated and the image file they had been started from identified. After checking that the process is really the virus process – which is a nice collaboration of the scanning engines and the system disinfection routines – the process can be killed safely. With no malware lurking around and (probably reverting our cleanup changes) the disinfection can be performed safely on a running system even without a single reboot. There might be special cases, however, that require special consideration. Nimda, for example, can attach itself to the running *Explorer* as an external thread on *NT* systems. The *Explorer* has to be killed to terminate the malware, but a clean copy of *Explorer* has to be restarted otherwise the system becomes unusable.

The handling of processes can be accomplished with relatively simple methods not considering the platform dependency. Even though the methods are different on different versions of 9x and *NT* systems there is no major difference between them. The platform dependency can be easily hidden from the disinfection routines behind an additional software layer.

Services are a closely related topic to running processes. It's increasingly common to disguise the malware as some service process. The advantage of this – from the malware's point of view – is that the service processes are invisible to the average user. These processes do not show up in the Task Manager process list for example.

On *Windows* 9x-based systems a service can be created by calling `RegisterServiceProcess()` from an already running process. The other option is to add a value under `SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices` in the registry. *Windows* will start all the programs registered under this key as system services. The removal of these services is relatively simple. The process has to be killed if it's running and the value has to be removed from the registry if it was registered there.

Under *NT* the registration goes through the Service Control Manager API that controls all the service processes in the system. New services are registered with the `CreateService()` call. The registration of a new service requires Administrator privilege that limits the malware when running with regular user privileges.

During the disinfection process the malware service has to be removed using the SCM API to make sure that no trace is left that could cause malfunction. Services can be removed with the `DeleteService()` call that requires Administrator privilege to succeed.

4. RECENT CASE STUDIES

Recent cases prove that hard to remove viruses are far from being extinct. More and more malware comes with increasingly complex logic for spreading and system infection.

4.1 Sircam

Sircam is a mass-mailer that started to spread in the middle of July in 2001 and caused a global epidemic in a couple of days. This virus picks random document files from the infected machine and sends them along with the infected attachment. This behaviour makes Sircam really harmful since it might disclose confidential documents from the infected computers.

When Sircam infects the system it copies itself to several locations and modifies several Registry values to make sure it will be started when the system starts up. There is one step in the infection procedure that makes this virus hard to remove.

Sircam modifies the EXE startup registry key to

```
HKEY_CLASSES_ROOT\exefile\shell\open\command =  
'' 'c: \recycled\SirC32.exe' '%1'*''.
```

After this starting any EXE file will require the presence of the worm. If the worm is removed without fixing the Registry the system becomes unusable since EXE files can not be started. When Sircam tries to infect machines on the local network it looks for writable shares and tries to replace rundll32.exe with itself. The original content of the file is copied to run32.exe that must be copied back during the disinfection process otherwise certain settings will not be accessible anymore (Control Panel for example). [7]

4.2 Nimda

Nimda was first found in the wild on 18 September 2001. The worm quickly spread around the world and infected millions of computers. Nimda uses four ways of spreading that made it really aggressive and quick spreading.

File infection

Nimda locates EXE files from the local machine and infects them by putting the file inside its body as a resource, thus ‘assimilating’ that file. These files then spread the infection when people exchange programs such as games.

Mass mailer

Nimda locates email addresses via MAPI from your email client as well as searching local HTML files for additional addresses. Then it sends one email to each address. These mails contain an attachment called README.EXE, which is the executable worm. The messages contain a malformed MIME header that tries to exploit a vulnerability in *Internet Explorer* that makes it possible to execute the attachment automatically when the message is opened. More

information on this vulnerability has been published by *Microsoft* in the security advisory labelled *MS01-020*. [8]

Web worm

Nimda starts to scan the Internet, trying to locate vulnerable WWW servers. Once a web server is found, the worm tries to infect it using a known security hole in *IIS (MS00-078)* [9]. If this succeeds, the worm will modify random web pages on the site to offer an EML file for download containing a malformed MIME header and the worm as attachment. The end result of this modification is that web surfers browsing the site with unpatched *Internet Explorer* will automatically become infected by the worm.

LAN propagation

The worm will search for file shares in the local network, either from file servers or from end user machines. Once found, it will drop a hidden file called *RICHED20.DLL* to any directory which has *DOC* and *EML* files. When other users try to open *DOC* or *EML* files from these directories, *Word*, *Wordpad* or *Outlook* will load and execute *RICHED20.DLL* causing an infection of the PC.

The difficulty of Nimda disinfection lies in the complexity of the changes it makes to the system configuration when it infects a computer. The following steps are needed to disinfect a system properly:

- removal from all the infected files
- removal of the loader code from infected HTML files
- removal of all the dropper files (*.EML, *.NWS)
- fixing of the shell= variable in the system.ini files
- fixing of the Registry values *Hidden*, *ShowSuperHidden* and *HideFileExt* values in the registry at *HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced* so that the hidden files will be visible again in *Windows Explorer*
- deactivation of the Guest account on *NT*
- removal of all the open shares from the computer.

The extensive list shows clearly that the disinfection of a Nimda-infected computer is very complex. Special disinfection tools had to be developed and distributed to help the customers to get rid of this complex virus [10]

5. Cleaning up the mess

After all this the conclusion is that it is necessary to extend the existing anti-virus infrastructure with new features to be able to cope with these complex disinfection tasks. Today's anti-virus programs have capabilities for programmed detection and removal that can be extended to provide disinfection capabilities for the whole system.

As with any other technical problems, there are many different way how this can be solved.

5.1 Stand-alone tools

Stand-alone tools seem to be the most common solution for today's complex removal tasks. Stand-alone tools have many advantages:

- Small size: the disinfection tool can have the routines for one or more specific malware or malware versions. These tools are written in some low-level language, like C or C++ so their size is relatively moderate.
- These small tools are easy to distribute through the Internet or any other media. They should even fit onto a single floppy and can be mailed to someone. Small size also benefits people with limited bandwidth, modem users being one good example.
- Not only customers can use them. Anyone who downloads the tools can use them since they do not depend on any specific anti-virus software.

Beside their advantages, these tools have a couple of disadvantages too.

Usually there is a lot of common in these tools which makes the version tracking somewhat difficult. Even though the usefulness of these tools drops as the time goes by it is still possible that some computer with a new operating system gets infected with an older virus. Making sure that the tools still work if a new version of the operating system or service pack comes out is not easy. There might be duplicated code that has to be checked in many different tools.

In the case of more complex malware when more sophisticated functionality (non-trivial detection for example) is required it might be hard to integrate that into the tool.

5.2 Tools built into the anti-virus programs

Aside from the stand-alone tools the special disinfection routines can be built into the anti-virus programs. One of the most important features in a disinfection tool is a quick and effective update mechanism. Whenever disinfection for new malware has to be added the disinfection system must be updated. The most obvious choice is to distribute the update bundled to the virus definition databases that are published regularly. On the implementation side there can be different ways of doing this. First of all let's look into the environment where the special routines would be implemented.

5.2.1 Executable code

The most obvious choice is the normal compiled, executable code. It's relatively easy to develop since all the development tools are available and familiar to the developers. Also big advantage is the immediately available huge set of functions and virtually unlimited access to the system services.

The use of familiar low-level languages and development tools provides the means to develop relatively small and efficient disinfection routines that can be distributed easily along with the database updates. However, as a side-effect, the use of low-level tools makes it more challenging to produce reliable and good quality software. This is especially true when it comes to virus outbreak scenarios with people working in the middle of the night, stressed out, pressed for time. In those conditions it might be harder to spot an off-by-one error in string copy routine...

5.2.2 Script environment

An alternative to binary modules is the use of some sort of higher level script language and its runtime environment to implement the disinfection routines. The main advantage here is the fact that higher-level languages let the programmer concentrate more on the actual problem than low-level issues. High-level languages usually have more extensive type and bounds checking and are far more fault-tolerant from the system point of view. In a script environment an error in the program will most likely not result in program crash. Programs implemented in script languages are usually small since the runtime environment provides most of the functionality. This makes it easy to distribute the updates with the usage of little bandwidth.

One disadvantage of the script environment is that if some functionality is missing from the runtime it's not easy to add it. If the runtime is distributed with the anti-virus software it has to be upgraded. An upgrade for the main application requires substantial development and testing effort that takes lot of time which is a critical issue during a virus outbreak.

5.2.3 Mixed script and binary solution

A better solution is to mix the best of both worlds and use them together. The problem with the script-based solution is that a special case, not covered by the runtime required an update to the runtime which is expensive both time- and money-wise. The effective and flexible binary code mixed intuitive and error-proof script language provides good foundations for an flexible disinfection system. With proper design of the API between the runtime and the extension module the need for a runtime upgrade can be avoided completely. The runtime of course might be upgraded to follow changes in operating system versions, fix bugs and limitations.

The relatively big runtime is provided by the anti-virus application, the binary extension module and the disinfection scripts are distributed via standard database update distribution channels. This approach provides maximum flexibility with modest bandwidth consumption. The update mechanism can be fully automated and transparent to the end user.

In the case of a new malware that requires some functionality not available in the script environment, the extension module can be updated and published together with the database updates and the disinfection scripts.

5.3 Security considerations

There are many aspects worth considering about security when functionality with complexity and power like this is implemented and taken into use on huge number of computers.

The files involved contain code that will be executed on the customer's machine. The system described above is really powerful and provides full control over the computer where it runs. Power with this magnitude must be under tight control protected from misuse by all possible means.

Anti-virus programs and databases have always been primary targets for retroviruses. Tampering of these files can pose serious threat to the computer that runs the code and the individual or organisation that owns and uses the computer. The reputation of the anti-virus vendor is also at stake since, from the user's point of view, the anti-virus application is responsible for the actions

of the tampered code.

The implementation must be careful since the code runs with powerful privileges. Since the code is updated regularly special care must be taken to ensure the best possible code quality in these modules. A vulnerability in these components might allow an attacker to elevate their privileges in the system. Any precedence like that is seriously harmful for both the user and the vendor’s reputation.

The protection involves extensive, well planned use of cryptographic hashes and digital signatures. Applied cryptography and Public Key Infrastructure has its own extensive literature and is beyond the scope of this paper.

REFERENCES

- [1] Mikko Hyppönen: Analysis of the One_Half virus, http://www.europe.f-secure.com/v-descs/one_half.shtml.
- [2] Mikko Hyppönen: Analysis of the DIR-II virus, <http://www.europe.f-secure.com/v-descs/dir2.shtml>.
- [3] Katrin Tocheva, personal communication.
- [4] Péter Ször, ‘Stream of Consciousness’, *Virus Bulletin*, October 2000, p.6.
- [5] Alexey Podrezov, personal communication.
- [6] Gergely Erdélyi, Description of the CodeRedII worm, <http://www.europe.f-secure.com/v-descs/bady.shtml>.
- [7] Alexey Podrezov and Gergely Erdélyi, Description of the SirCam worm, <http://www.europe.f-secure.com/v-descs/sircam.shtml>.
- [8] *Microsoft Security Bulletin MS01-020*, <http://www.microsoft.com/technet/security/bulletin/MS01-020.asp>.
- [9] *Microsoft Security Bulletin MS00-078*, <http://www.microsoft.com/technet/security/bulletin/ms00-078.asp>.
- [10] F-Secure Anti-virus Research Team: Analysis of the Nimda worm, <http://www.europe.f-secure.com/v-descs/nimda.shtml>.