PROCEDURES AND TECHNIQUES FOR PREVENTION OF AND RECOVERY FROM FAST SPREADING MALWARE

Ralph Philbrick and Stephen Posniak
Office of Information Technology, US Equal Employment Opportunity Commission, Washington, D.C. 20507, USA

Tel +1 202 663 4356; +1 202 663 4449 • Email ralph.philbrick@eeoc.gov; stephen.posniak@eeoc.gov

ABSTRACT

During the peak point (February and March) of the Netsky–Beagle war of 2004, most organizations and all of the anti-virus vendors were challenged by the speed with which the successive variants of these worms were propagating. Rolling out daily definition/pattern updates was not sufficient to prevent a relatively high rate of infection.

Operating in a *Novell Netware* environment, the U.S. Equal Employment Opportunity Commission (EEOC) devised a procedure for scanning key registry values during the client login process and automatically executing the appropriate repair tool on client workstations which were found to have been definitely infected. This approach, developed in order to conserve network resources, also resulted in a much more efficient and rapid recovery procedure during such periods, as well as a quicker way to log specific types of damage. We provide specific technical details for the *Novell Netware* and Symantec Anti-Virus 8.0 environment, but will also suggest methods for adapting it to other networking software environments.

EEOC BACKGROUND INFORMATION

The U.S. Equal Employment Opportunity Commission enforces Title VII of the Civil Rights Act of 1964 (race, color, sex, national origin, religion), the Americans with Disabilities Act, the Age Discrimination in Employment Act and the Equal Pay Act. Additional information about the EEOC's mission and activities is available on the agency's public website at http://www.eeoc.gov. Headquartered in Washington DC, EEOC has 50 field offices located in larger cities around the US, and also administers work sharing agreements with about 90 state and local government fair employment practices agencies.

It operates a network of 76 servers and 2,500 client workstations in a *Novell Netware Version 6* server environment, with *Novell GroupWise Version 6.5* as its email system. The network also makes use of *ZenWorks*, and deploys *Symantec Anti-Virus Corporate Edition* on all servers and client workstations. At the time of the events described in this paper, the network included a mix of *Windows 98* and *Windows XP* client workstations.

CURRENT MALWARE ISSUES FACING IT ORGANIZATIONS RESPONSIBLE FOR ADMINISTRATION OF ENTERPRISE NETWORKS

1. All major anti-virus packages rely on definition or pattern file updates. So long as this remains the case, the possibility always exists that a particular computer may become infected by a malware exploit before it receives the definition file that defends against it.
2. Much of the current malware consists of blended threats; it arrives as a worm and then exploits other vulnerabilities once inside the firewall. This increases the speed at which at it can spread, and increases the likelihood that a particular computer will become infected.
3. Much of the current malware attacks installed anti-virus software and attempts to disable it. Successfully disabling anti-virus software leaves a computer vulnerable not only to new malware, but also to viruses it would otherwise be able to defend against. Additionally, enterprise anti-virus software which has been disabled is unable to report a malware attack, thus making it especially difficult to track down and identify infected machines.
4. Supplemental repair tools, once they become available, are often cumbersome to deploy in an enterprise environment. Additionally, many viruses are very time-consuming to clean, even when a repair tool is used. (This is a consequence of the virus being cleaned, not a deficiency of the tool itself. If a virus could potentially infect every file on a computer system, then the cleaning tool must scan every file on that computer system to clean it – a very time-consuming process, to say the least.)
5. This results in two competing goals confronting the IT organization. On the one hand it has an obligation to disable the virus as quickly as possible. On the other hand it has an obligation to keep the enterprise's information systems operational. A 'cure' that interferes with the organization's productivity as much as, or more than, the malware itself is unacceptable.

THE NETSKY/BEAGLE WORM WAR AND ITS IMPACT ON EEOC

During the first few years of deployment, the EEOC Network was not highly vulnerable to most of the prevailing malware, not only because of its deployment of anti-virus software, but also because *Novell GroupWise* in most instances would not execute attachments that users might open, but instead would simply display the code. With the growth in the variety of types of malware found in the wild, as well as the increase in HTML-format messages which can activate executables without the need for the user to click on an attachment, the level of this intrinsic invulnerability markedly decreased, resulting in a corresponding increase in EEOC's dependence on current, properly deployed anti-virus definitions.

EEOC's increased vulnerability became more evident to its networking and information security staff during late February and early March 2004. (This was a time when many other organizations were also forced to come to grips with such vulnerabilities.) The emergence of W32.Netsky.B on February 18, followed by Netsky.C on February 24 and Netsky.D on March 1, amounted to the first true flinging of the malware gauntlet for EEOC. In the case of each of these fast-spreading worms, the earliest that the necessary detection definitions could be downloaded and posted on the network was the day after discovery. As the frequency of spoofing of valid email addresses also increased during this period, the exposure of EEOC staff to such malware attachment-bearing messages also increased. In some cases, the messages deliberately resembled bounced email which the user might think he or she had originally sent.

The EEOC Office of Information Technology (OIT)'s first response was to issue an internal general alert message cautioning all EEOC staff not to even open (let alone click on any attachments of) strange-looking or unexpected messages which appeared suspicious in any way. Users were advised first to contact the Help Desk or one of the Information Security Officers if they had any questions about such messages. The same alert also explained that in the current situation (and occasionally in the future) there would often be as much as a day between the time such malware started propagating and the time that the anti-virus definitions which detected and disabled it could be deployed on the network. EEOC has deliberately avoided issuing such general alerts on a frequent basis, in order to avoid the 'boy who cried wolf' syndrome (i.e. familiarity breeds a tendency to bypass the message, if not actual contempt for it.)

One of the responses to the above alert from some of EEOC's program officials and users was to point out that it was the job of some EEOC staff responsible for public 'Outreach' and related functions to open massive amounts of email from the public, and that it was not practical to expect them to spend time trying to judge the authenticity of specific messages or to consult OIT for support and assistance in each such instance. During this period, the number of infected workstations had increased markedly over the normal rate by the time that the correct definitions had been deployed and that some valid scanning and log reviews could take place.

During the worst part of this period, as many as 70 infected workstations were detected after the fact across the network. What complicated the situation further was that on some of those (still relatively few) workstations which Netsky had infected, it had succeeded in disabling *Symantec Anti-Virus* (*SA-V*). As a result, not only were some workstations infected but they were also *not* being reported by *SA-V* as infected. This presented OIT with a serious problem. We had an unknown number of infected workstations connected to the network, and we had no idea which workstations they were.

Based on subsequent follow-up discussions with other organizations and with AV vendors, it has been our observation that such exceptional difficulties were also being encountered elsewhere.

EEOC'S RESPONSE

What we present here is a strategy to efficiently identify and halt the spread of a virus *after* it has succeeded in compromising an unknown number of computers in your organization. We will be using our own experience combating the Netsky worm as a case study. Our approach was developed for a *NetWare 6* server environment with a mix of *Windows 98* and *Windows XP* client workstations. However, the approach is generally applicable to any network environment that includes *Windows* workstations and makes use of login scripts. The software used is incorporated into *Windows* itself, and makes use of WSH (Windows Scripting Host) WMI (Windows Management Instrumentation) and *Windows* batch files.

The specific steps for implementing a rapid recovery strategy are:

1. Include a batch file as part of your network login script. (Usually you would place the batch file at the end of the login script.) The batch file should be hosted by a network server in a read-only folder. By placing the code in a batch file rather than running it directly from the login script, it becomes easier to deploy new updates. No further editing of the login script is necessary. All further modifications can be managed simply by copying a new version of the batch file to each network server that hosts it.
2. Use WSH scripts running from the batch file to identify a virus infection.
3. When possible, use a cleaning tool to disable the virus once the WSH script has identified that a computer is infected. In our example code (Figures 1 through 3), a Symantec cleaning tool, FXNETSKY.EXE is being run from a WSH script. Cleaning tools from most other anti-virus software products can be scripted just as easily.
4. Once the virus has been identified, use either the cleaning tool or the WSH script to generate a log file and store it to a network drive. Generate the name of the log file using *Windows* environment variables to uniquely identify the user and/or workstation name of the infected computer. This enables a network administrator to quickly identify which computers on a network have been infected.

EXPLANATION OF CODE EXAMPLES

The first example (Figure 1) shows the script we actually ran in the EEOC to combat the Netsky worm. As you can see, it is very simple. All it does is attempt to read the registry key value that the Netsky worm uses to activate itself at *Windows* startup. If it finds the registry key, it runs the cleaning tool and saves a log file to one of the network drives to which our users have access. Otherwise, it finishes silently. In a WSH script, attempting to read a registry key value that doesn't exist generates an error, so a 'try-catch' block is used to catch the error and 'do nothing' if the registry key didn't exist. If reading the key value was successful, then an error was not thrown, and the cleaning tool is run on the workstation.

Figure 1:

```
var oWShell= WScript.createObject("WScript.Shell");
// If the key value exists, we know we're looking at
a "problem child."
// If the key value does not exist, it generates an
error which we trap silently.
try
{ var keyVal=
oWShell.RegRead("HKLM\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Run\\ICQ Net");
 oWShell.Run("F:\\PUBLIC\\Updates\\FxNetsky\\fxnetsky.exe
/s /exclude=f:\\ /exclude=g:\\ /exclude=s:\\ /
log=p:\\%USERNAME%FxNetSky.log");
}
catch(e)
{
}
// Now we check again for another variant that uses
a different RUN key value.
// This could be nested rather than executing
sequentially, but it's easier to read and easier to
extend.
// And if someone actually managed to get infected
with BOTH variants, they deserve to have the clean-
ing tool run twice!
try
{ var keyVal=
oWShell.RegRead("HKLM\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Run\\wserver");
 oWShell.Run("F:\\PUBLIC\\Updates\\FxNetsky\\fxnetsky.exe
/s /exclude=f:\\ /exclude=g:\\ /exclude=s:\\ /
log=p:\\%USERNAME%FxNetSky.log");
}
catch(e)
{
}
```

The second example (Figure 2) is a 'manual' approach to identifying and disabling a virus. If a repair tool is available, use it. But what can you do if no cleaning tool is available? So long as you can identify the registry key value being used to load the virus code (either by reading the virus behaviour from an anti-virus website, or by examining an infected machine) you may not be able to clean the virus from the computer, but you can at least identify and disable it. The next example shows how this approach could have been applied to cleaning the Netsky worm.

Figure 2:

```
var oWShell= WScript.createObject("WScript.Shell");
var oFile=
WScript.createObject("Scripting.FileSystemObject");
var sFileOut=
"C:\\"+oWShell.ExpandEnvironmentStrings("%USERNAME%")+"
FxNetSky.log";
// If the key value exists, we know we're looking at
a "problem child."
// If the key value does not exist, it generates an
error which we trap silently.
try
{ var keyVal=
oWShell.RegRead("HKLM\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Run\\ICQ Net");
 oWShell.RegDelete("HKLM\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Run\\ICQ Net");
 oTextFile= oFile.createTextFile(sFileOut, true);
 oTextFile.writeLine("NetSky Virus Detected!");
 oTextFile.close();
}
catch(e)
{
}
```

The third example (Figure 3) is much more involved, and anticipates a possibility that we have not encountered yet. The values in the RUN registry key can actually be any conglomeration of characters, as can be the filename being run. It's only a matter of time before some virus writer gets the bright idea to use randomly generated characters for the RUN key entries, and for the filenames being run. This would make an automated cleaning tool much more difficult to develop.

This script example relies on knowing what entries *should* exist on a standard workstation configuration in your organization, and then removing anything that's not on the 'approved' list. Understand that you could end up unintentionally disabling legitimate software if you use this script and aren't thorough about building your list of approved applications! Nonetheless, if your organization is being hit by a rapidly spreading virus that you have no other means to defend against, accidentally disabling one or two legitimate applications may be a small price to pay. Interestingly, a script like this one could also be used to help enforce software use policies. If you don't authorize the use of AIM or ICQ for instance, this script would prevent them from loading at startup.

Figure 3:

```
// WARNING!!! THIS SCRIPT WILL DELETE ALL UNAUTHOR-
IZED VALUES FROM THE RUN KEY!!!
// YOU MUST ADD YOUR AUTHORIZED KEY VALUES TO THE
Authorized ARRAY BEFORE RUNNING THIS SCRIPT!!!
var oWShell= WScript.CreateObject("WScript.Shell");
var HKLM = 0x80000002;
var sRegPath =
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
var oFile=
WScript.createObject("Scripting.FileSystemObject");
var sFileOut=
"C:\\"+oWShell.ExpandEnvironmentStrings("%USERNAME%")+"Virus.log";
var oTextFile;
var aAuthorized;
/***************************************
***** SAMPLE AUTHORIZATION LIST *****
*************************************
 aAuthorized= new Array
 (
 "CTStartup",
 "Jet Detection",
 "Tweak UI",
 "dla",
 "WINDVDPatch",
 "Daemon14",
 "ccApp"
 );
*/
try
{ oReg =
GetObject("winmgmts:root\\default:StdRegProv");
 oMethod = oReg.Methods_.Item("EnumValues");
 oInParam = oMethod.InParameters.SpawnInstance_();
 oInParam.hDefKey = HKLM;
 oInParam.sSubKeyName = sRegPath;
 oOutParam = oReg.ExecMethod_(oMethod.Name,
oInParam);
 var aNames = oOutParam.sNames.toArray();
}
catch(err)
{ WScript.Echo("Error Code: " + err.number + "
Description: " + err.description);
}
var aIllegal= new Array();
var allowed= new Boolean(false);
try
{ for(i in aNames)
 { if(aAuthorized == undefined)
  { WScript.Echo("Your authorization list is
empty!");
  break;
 }
 for(j in aAuthorized)
 { if(aNames[i] == aAuthorized[j])
  { allowed= true;
  break;
  }
  else
  { allowed= false;
  }
 }
 if(!allowed)
 { if(aNames[i] == "")
 {oWShell.RegWrite("HKLM\\"+sRegPath+"\\","","REG_SZ");
 }
 else
 {oWShell.RegDelete("HKLM\\"+sRegPath+"\\"+aNames[i]);
 if(oTextFile == undefined)
  oTextFile= oFile.createTextFile(sFileOut, true);
 oTextFile.writeLine("Unauthorized Run Value:
"+aNames[i]);
 }
 }
}
if(oTextFile != undefined)
 oTextFile.close();
}
catch(err)
{ WScript.Echo("Error Code: " + err.number + "
Description: " + err.description);
}
```

A note regarding the RUN key: in order to load themselves each time a *Windows* workstation starts up, most viruses add an entry to the RUN key of the registry. The full key is:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

This key can be examined and edited using any number of tools, including REGEDIT.EXE and REGEDT32.EXE. Be aware that there are other registry keys that *could* be used by a virus to load at startup. They are as follows:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

HKEY_USERS*programID*\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

These other registry keys can be read, modified, or deleted using the same method illustrated in the sample scripts. Only the name of the key would need to be changed.

Program shortcuts placed in the following folders will also run at startup:

C:\Documents and Settings\All Users\Start Menu\Programs\Startup

C:\Documents and Settings*userID*\Start Menu\Programs\Startup

Once the unwanted shortcuts have been identified, they can be deleted simply by running the DEL command from a batch file.

CONCLUSION

Control of fast-spreading malware requires a pragmatic combination of approaches to supplement the traditional, standardized deployment of anti-virus software. While this paper has focused primarily on the use of *Windows* login scripting, it is equally important to employ the judicious use of malware alerts to the enterprise's IT users, not only when significant changes develop in the characteristics of malware-bearing email messages, but also when new tools or software revisions are implemented. For example, although it was designed to be transparent to the user, when the login script tool mechanism was first deployed, a brief alert message was sent to all EEOC's IT users which stated: "*OIT has developed a method to repair workstations that have been infected with the W32.Netsky virus. It is imperative that you log out and log back in to activate the repair program so that we can remove this virus from all workstations as quickly as possible. If you experience any problems after logging back into the system, please contact the OIT Help Desk.*"

Similarly a few days later, users were alerted to OIT's implementation of a revision to the *Novell GroupWise* email client deployment. "*This particular virus can infect the system by the mere action of opening an email in HTML format. It is no longer necessary to open an attachment to execute a virus. OIT has been successful in containing the virus by implementing many safeguards, but at the cost of many man-hours. The latest safeguard implemented by OIT last Friday was to disable the 'Read Next Message' option in GroupWise so that messages cannot be opened automatically without first clicking on them. This measure will eliminate the chances that a virus is executed accidentally and prevent further outbreaks of this virus.*"

It is only through the use of such a pragmatic combination of approaches and tools, not only concerning email but also concerning the exploitation of web browsers and network shares, that organizations can hope to control the fast-spreading malware which periodically confronts them in this day and age.