Report

McAfee

The Rise of AutoRun-Based Malware

By Vinoo Thomas, Prashanth Ramagopal, and Rahul Mohandas

Table of Contents

Abstract

Most people associate today's computer viruses and other prevalent malware with the Internet. But that's not where they started. Lest we forget, the earliest computer threats came from the era of floppy disks and removable media. With the arrival of the Internet, email and network-based attacks became the preferred infection vector for hackers to spread malicious code—while security concerns about removable media took a back seat. Now, however, our attention is returning to plug-in media.

Over the years, floppy disks have been replaced by portable hard drives, flash media cards, memory sticks, and other forms of data storage. Today's removable devices can hold 10,000 times more data than yesterday's floppy disks. Not only can they store more data, today's devices are "smart"—with the ability to run portable software programs[1] or boot operating systems. [2,3]

Seeing the popularity of removable storage, virus authors realized the potential of using this media as an infection vector. And they are greatly aided by a convenience feature in operating systems called AutoRun, which launches the content on a removable disk without any user interaction.

This paper traces the advancements in AutoRun-based malware. We also discuss methods to proactively detect and stop malware that spreads via removable drives, using a combination of traditional anti-malware and cloud-computing techniques.

The Return of Removable-Disk Malware

During the last couple of years we have seen malware authors achieve stunning success as they increasingly incorporate the AutoRun technique into malware families. In addition to traditional AutoRun worms that use this feature, pure-play backdoors,[4] bots,[5] password stealers,[6] and even parasitic viruses[7] that previously required a user to double-click an executable file to infect the system have incorporated the AutoRun technique. And the easy availability of source code for these families on the Internet allows script kiddies to repackage and compile new variants.

The year 2008 included several high-profile incidents in which AutoRun-based threats made headlines. Poor quality-control practices from hardware manufactures led to repeated incidents of USB memory sticks,[8] hard drives,[9] MP3 players,[10] and digital photo frames[11] being sold to customers with AutoRun malware preinstalled. Unsuspecting customers got more than what they paid for.

The U.S. military[12] banned the use of thumb drives, flash media cards, and all other removable data storage devices from their networks to try to keep the worm from multiplying any further. For some departments, the ban was a minor inconvenience, but to soldiers in the field who relied on removable drives to store information, this drastic measure was too extreme.

Security vendors were not spared either. In a major embarrassment, Telstra[13] distributed worm-infected USB drives to participants at the AusCERT security conference. Luckily the worm did not have a payload, and no serious damage was done.

AutoRun-based malware may have also traveled into orbit and onto the International Space Station[14] via an infected USB drive owned by an astronaut. The laptop used by the astronaut reportedly did not have any anti-malware software to prevent an infection.

1 http://www.u3.com
2 http://www.remote-exploit.org/backtrack_download.html
3 http://www.cnet.com.au/software/operatingsystems/0,239029541,339271777,00.htm
4 http://vil.nai.com/vil/content/v_142042.htm
5 http://www.cert-in.org.in/virus/worm_hamweq.htm
6 http://vil.nai.com/vil/content/v_147533.htm
7 http://vil.nai.com/vil/content/v_147094.htm
8 http://www.theregister.co.uk/2008/04/07/hp_proliant_usb_key_infection/
9 http://blogs.zdnet.com/security/?p=2016
10 http://www.virusbtn.com/news/2008/01_08a.xml
11 http://www.securityfocus.com/brief/670
12 http://blog.wired.com/defense/2008/11/army-bans-usb-d.html
13 http://blogs.zdnet.com/security/?p=1173
14 http://news.bbc.co.uk/2/hi/technology/7583805.stm

These incidents are a small percentage of episodes that were publicly reported. In reality, a majority of security incidents go unreported to avoid media attention. These incidents also teach us that every industry—military or commercial—is increasingly susceptible to threats that use AutoRun as an infection vector.

Distribution of AutoRun-Based Malware

To investigate the most prevalent compilers and packers used to create AutoRun worms, we gathered data for all AutoRun-based worms that McAfee® Avert® Labs received in the last three years. The following charts display compiler and packer distribution of AutoRun-based worms.

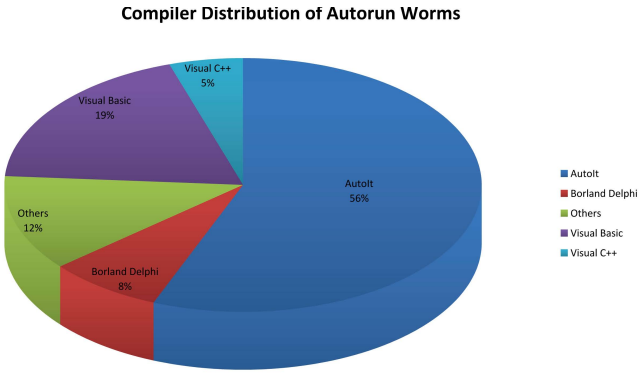

Figure 1: Compiler distribution from McAfee sample collections (as of December 2008)

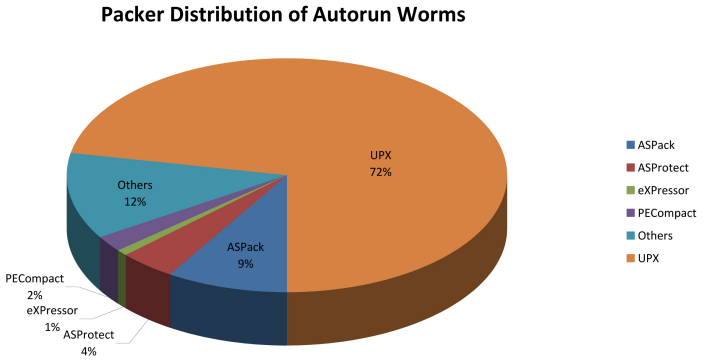

Figure 2: Packer distribution from McAfee sample collections (as of December 2008)

From the data collected we observe that AutoIt and UPX are the most widespread compiler and packer, respectively, for creating AutoRun-based malware. We attribute this popularity to three key reasons:

- AutoIt[15] and UPX[16] are open-source software and freely available on the Internet
- Malware source code for creating AutoIt-based worms is readily available on the Internet
- Files compiled with AutoIt 3.2x versions and earlier can be decompiled to dump the original script using a freely available decompiler[17] on an AutoIt blog

These reasons allows script kiddies to look at existing AutoRun worm samples written in AutoIt and make modifications to the script to suit the author's requirements. The modified script is then recompiled to create fresh variants of the worm and can be tweaked until it bypasses anti-malware detection.

15 http://www.autoitscript.com/AutoIt/
16 http://upx.sourceforge.net/
17 http://leechermods.blogspot.com/2008/02/autoit-decompiler-unpacker-and-script.html

Typical alterations made to the body of the worm are updated links to download further malware and provocative messages within the worm. We have seen AutoIt-flavored malware in French, Indonesian, Punjabi, Vietnamese, and other languages with regional themes.

```
; <AUT2EXE VERSION: 3.2.0.1>

; ---------------------------------------------------------------------------
; <AUT2EXE INCLUDE-START: C:\Documents and Settings\Le Dinh Thanh\Desktop\zin.au
; ---------------------------------------------------------------------------

;-----------------------------------------------------------------
; Author : VIET NAM TEAM
; Name: TermeX Bot
; Version : 3.5
; Usage : Advertise via Y!M,MSN,AIM
; Published : 20-9-2006
;-----------------------------------------------------------------
```

Figure 3: A Vietnamese AutoRun worm

```
 IF $GDIMARR0000 [0]= ""  THEN
$GDIMARR0000 [0]= "cyber cafe scandal visit www." & $A2681B081B014 & " "
 ENDIF
$GDIMARR0000 [4]=  INIREAD (@SYSTEMDIR & "\" & "setting" & ".ini" , "setting" ,
"tin[4]" , "" )
 IF $GDIMARR0000 [4]= ""  THEN
$GDIMARR0000 [4]= "Latest video shot of infosys girl www." & $A2681B081B014 & "
"
 ENDIF
$GDIMARR0000 [6]=  INIREAD (@SYSTEMDIR & "\" & "setting" & ".ini" , "setting" ,
"tin[6]" , "" )
 IF $GDIMARR0000 [6]= ""  THEN
$GDIMARR0000 [6]= "stream Video of Nayanthara and Simbu  www." & $A2681B081B014
& " "
 ENDIF
$GDIMARR0000 [7]=  INIREAD (@SYSTEMDIR & "\" & "setting" & ".ini" , "setting" ,
"tin[7]" , "" )
 IF $GDIMARR0000 [7]= ""  THEN
$GDIMARR0000 [7]= "Aishwarya Rai videos www." & $A2681B081B014 & " "
 ENDIF
```

Figure 4: An AutoRun worm using Indian scandals as bait

```
        If not $array[1] = "User" or "Admin" or "Administrator" then
    $msg = " ( est envoyé par " & $array[1] & " , pas de virus ) "
        else
        $msg = ""
        EndIf

EndIf
else
        $msg = " ( est envoyé par " & $msg & " , pas de virus ) "
EndIf

; List of random messages
Dim $tin[13]
$tin[0] = "c'est ma carte de voeux de Noel que j'ai fait seulement pour toi " &
$tin[1] = "Microsoft donne 2007 copies gratuits de Windows Vista pour 2007 premi
$tin[2] = "vote pour notre Miss de beauté aujourd'hui :x " & $website & "/?miss_
$tin[3] = "the only way to clean some online viruses that may lead you into trou
$tin[4] = "Joyeux Noel et Bonne année !!! " & $website & "/?id=greetings << "
$tin[5] = "l'entrainneur de Chelsea est gravement blessé par Gallad " & $website
$tin[6] = "Attention!!! Il y aura un tremblement de terre ce soir : " & $website
$tin[7] = "J'ai fait 10 cadeaux pour les 10 premieres personnes qui commentent s
$tin[8] = "you are virus infected . Use this tool to remove viruses from your PC
$tin[9] = "Enculé !!!  " & $website & "/?id=news  X-(  "
$tin[10] = "Osama Bin Laden est arreté " & $website2 & "/?news_id=18388 " & $msg
$tin[11] = "Creer les bombs hyper forts avec Whisky, Coke et Mentos " & $website
$tin[12] = "J'ai gagne au LOTO:  " & $website & "/?id=winning_list Viens feter c
```

Figure 5: A French-language AutoRun worm

The bait messages within the worm's body include admonitions to not view porn, religious chants, sensationalistic news, and videos of local sex scandals.

AutoRun Woes

AutoRun[18] is a convenience feature in operating systems that automatically launches the content on removable media as soon as a drive is inserted into a system. The AutoRun process is triggered using the file autorun.inf,[19] which specifies the path to an executable that runs automatically.

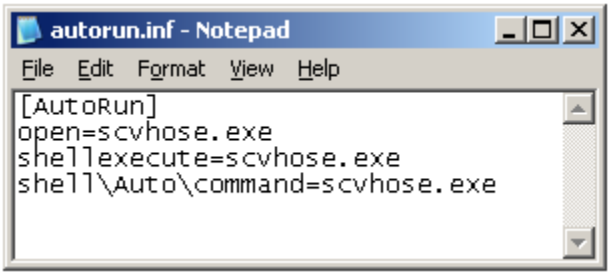

Figure 6: A typical autorun.inf file

Unfortunately for most of us, malware authors have seized on this benign feature to autolaunch malware without any user interaction when a removable device (for example, a memory stick or external hard drive) is inserted into a system. The computer recognizes a newly connected removable drive, detects the autorun.inf file, and loads the malware.

Another infection vector occurs due to drive mapping. When a computer maps a drive letter to a shared network resource with a malicious autorun.inf, the system will (by default) open autorun.inf and follow its instructions to load the malware. Once the system is infected, the malware will do the same with other removable drives connected to it or other computers in the network that attempt to map a drive letter to its infected shared drive—hence, the frequent replication.

Avert Labs first added detection in January 2007 for autorun.inf files that accompany malware as Generic!atr.[20] Since then we have observed an alarming increase[21] in malware using AutoRun as an infection vector. We'll give you an example of how rampant the problem of AutoRun malware is in the real world: Shown below is the McAfee global virus map, which tracks statistics of infections observed by McAfee users worldwide.

18 http://en.wikipedia.org/wiki/Autorun
19 http://msdn.microsoft.com/en-us/library/bb776823.aspx
20 http://vil.nai.com/vil/content/v_141387.htm
21 ESET. "Global Threat Trends," (November 2008). http://www.eset.com/threat-center/case_study/Global_Threat_Trends_November_2008.pdf

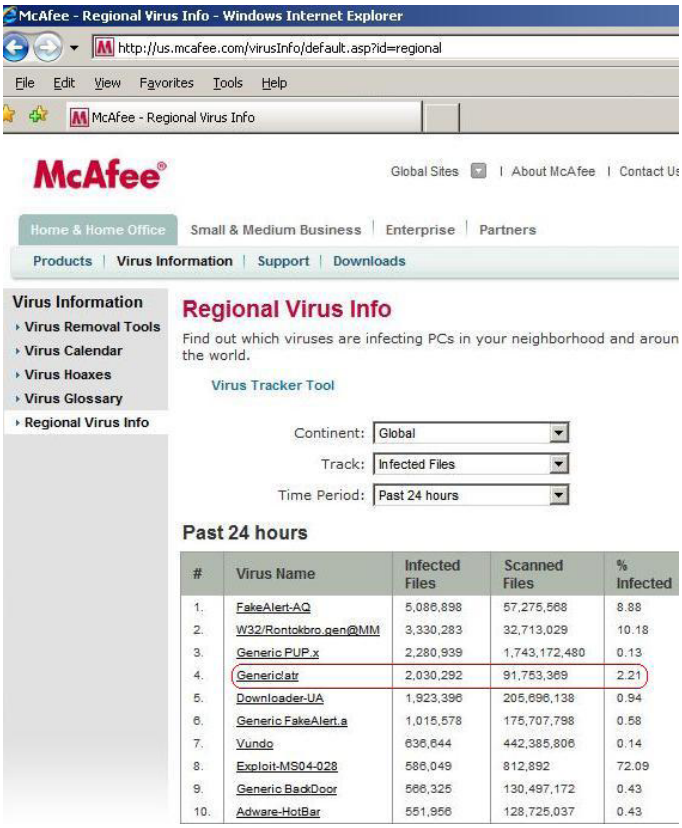

Figure 7: McAfee global virus statistics (as of 1 December 2008)[22]

We detected Generic!atr on more than two million files in a 24-hour period. This malware has been in the top five detections globally ever since the signature was added to the McAfee DAT files. Figure 7 shows detections only on computers running McAfee anti-virus whose users have chosen to report their detections. When you take into account the millions of computers on the Internet and other vendor[23] detections[24] of AutoRun-based threats, one can understand how rampant the problem really is.

Incomplete autorun.inf cleaning

What happens if the anti-malware software deletes only the malware file and does not delete the accompanying autorun.inf in the root of the drive? Whenever a user clicks on My Computer and tries to navigate to the root of a drive, explorer.exe automatically loads autorun.inf. If the malware file was deleted but autorun.inf was left behind, then whenever users attempt to open the contents of the drive via Explorer, they will see the following error:

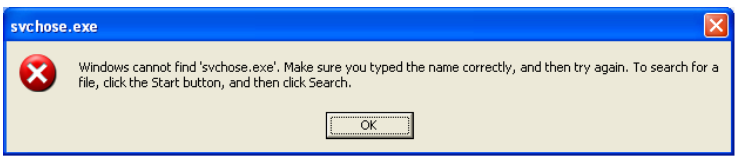

Figure 8: Users see this error message when accessing an infected drive after an incomplete cleaning

22 http://us.mcafee.com/virusInfo/default.asp?id=regional
23 http://blogs.technet.com/mmpc/archive/2008/06/20/taterf-all-your-drives-are-belong-to-me-1-one.aspx
24 http://www.eset.com/threat-center/case_study/Global_Threat_Trends_November_2008.pdf

This error message is logical because the executable path listed in autorun.inf no longer exists. Thus every time the drive is accessed, Windows will deliver this message until the user manually deletes autorun.inf.

The disconnected autorun.inf results in additional cleaning overhead, with customer queries and complaints about the worm not being properly cleaned. An administrator would have to manually delete this autorun.inf or submit it to the anti-malware vendor to add detection.

Traditional detection methods

Anti-malware vendors have traditionally used checksum or string-based logic to detect malicious autorun.inf files. A common technique is to detect strings that are unique to an autorun.inf. Here's an example of pseudo code for a string-based signature detection of an autorun.inf file:

```
Eliminate on string "[AutoRun]" at beginning of file
Detect on string "open = malware executable name"
```

However, malware authors have subverted traditional methods of detection by obfuscating the contents of autorun.inf. Introducing junk characters in the file will defeat traditional hash or string-based detection.

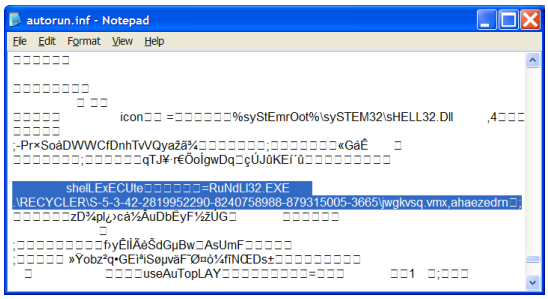

Figure 9: An example of an obfuscated autorun.inf file used by W32/Conficker.worm

Figure 9 shows how the W32/Conficker.worm usually works—by creating an autorun.inf file such as this.[25] The worm drops this file, which contains about 60KB of random binary data.[26] The command to execute the worm is concealed within the garbage data.

The bad guys have also introduced random executable filenames, so that on each infected machine and with every execution the filename in the path to the AutoRun executable will change.

With malware authors using a combination of junk characters to obfuscate along with random filenames on every execution it's easy to bypass conventional detection methods used by anti-malware researchers.

Moreover, if the malware uses a weak or common filename—such as autoplay.exe or setup.exe—in autorun.inf, researchers would hesitate to add detection using these strings because the chances of the signature producing false-positives would be very high.

Smart removal of autorun.inf

We propose to solve this problem in the following way: Whenever malware is detected on a system by the anti-malware scanner, before we clean we first check for the presence of an autorun.inf file in the root of the drive. If an autorun.inf is present, we then scan its contents for the filename of the malware that the scanner detected. If the malware filename is present, then the scanner labels that autorun.inf as malicious and associated with the detected malware. The scanner then deletes both the AutoRun and malware files.

25 http://vil.nai.com/vil/content/v_153710.htm
26 http://isc.sans.org/diary.html?storyid=5695

Pseudo code for removal logic

```
if (malware detected in drive) then
{
   if(drive has autorun.inf) then
        {
        if(autorun.inf has the malware entry) then
            {
               detect and delete autorun.inf
            }
        }
}
```

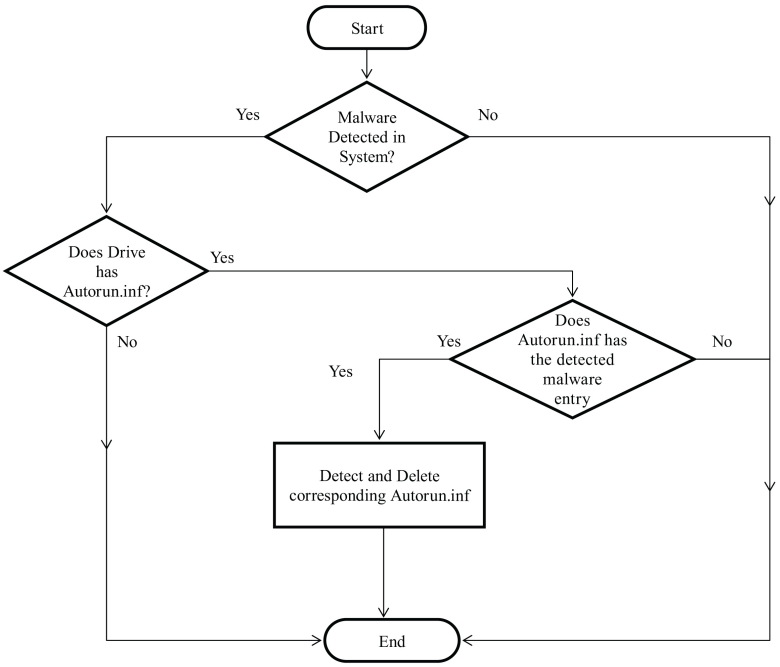

Figure 10: Sample logic for detecting and cleaning an associated autorun.inf file.
After deleting the detected malware, the anti-malware scanner then cleans associated malicious autorun.inf.

This flowchart acts as a generic signature to detect malicious autorun.inf files and will proactively detect new variants without the need for additional hash- or string-based signatures. This will also ensure accurate cleaning of these files without any false-positives.

Leveraging In-the-Cloud Computing Technology

Anti-malware software has traditionally been installed on individual computers around the world as endpoint protection. Depending on the vendor, these systems receive updated signature files and threat information on an hourly or daily basis. With the emergence of cloud computing technology,[27] this community of anti-malware nodes[28] can collectively contribute threat intelligence back to the cloud. With millions of active anti-malware endpoints, cloud technology in real time collectively captures and correlates fingerprints of new and potentially malicious code across the threat landscape. Each endpoint transmits compact fingerprints about a suspicious file to an automated evaluation system for immediate assessment. In seconds, the backend threat analysis tools can cross-check the characteristics of the file to determine its likely threat level and notify the endpoint to take appropriate action.

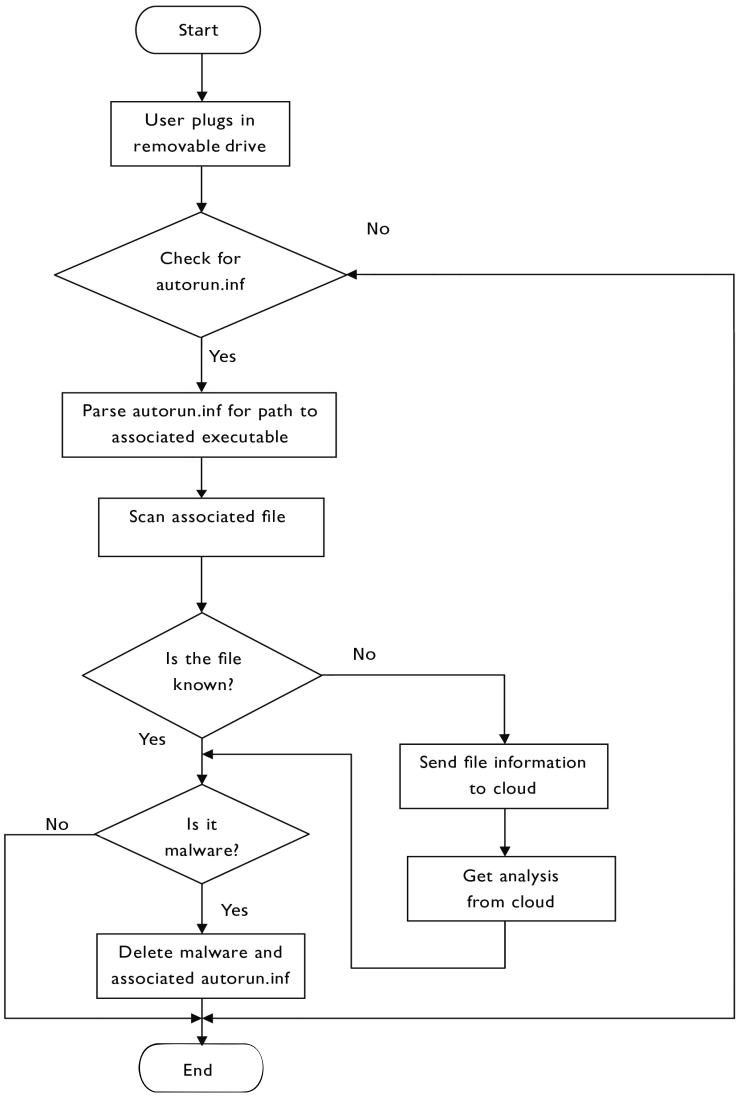

Figure 11: Using in-the-cloud computing to detect AutoRun-based malware

27 Buyya, R., Yeo, S., and Venugopal, S. "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," (September 2008). Proceedings of the 10th IEEE International Conference, Dalian, China. http://www.gridbus.org/~raj/papers/hpcc2008_keynote_cloudcomputing.pdf

28 http://www.mcafee.com/us/enterprise/products/artemis_technology/index.html

In-the-cloud computing can be applied to "smart-scan" USB drives, for example, as described in Figure 11. Whenever a user inserts a removable device into a computer, the anti-malware agent scans the root of the removable drive for an autorun.inf. If that file exists, it is parsed to trace the path to the executable, which is scanned. If no signature for the executable exists in the local signature database, an agent sends a fingerprint of the file for instant lookup to the anti-malware vendor's comprehensive database. If the fingerprint is identified as known malware, a response to quarantine or delete the file is sent back in milliseconds to the user's computer. In addition to blocking execution of the malicious file, the cleaning logic we proposed earlier can delete or quarantine the accompanying autorun.inf file as well.

The rate at which malware morphs and propagates makes it difficult for any security vendor to keep pace using traditional defenses. The need today is to correlate signature and behavioral techniques with real-time threat intelligence gathered from the user community.[29] The materialization of cloud computing technology[30] makes endpoints smarter and safer—by sharing collective threat intelligence and preventing damage before a signature update is available.

The Road Ahead

Why is AutoRun as an infection vector so popular—especially with machines running Microsoft Windows? AutoRun is enabled by default on all flavors of Windows, including the latest versions of Windows Vista and Windows Server 2008. With AutoRun-based infections on the rise, Microsoft could make a world of difference by addressing this exploited convenience feature. Microsoft has introduced improvements to AutoRun in Windows 7 and the same could be applied to older operating systems in future Windows updates.[31]

Looking at the recent evolution of AutoRun-based malware we can expect to encounter more complex and challenging samples of this genre. It is, therefore, essential to revisit traditional detection methods and improve upon our malware-defense strategies. The techniques discussed in this paper will go a long way toward containing the spread of AutoRun-based malware.

29 McAfee Avert Labs. "From Zero-Day to Real-Time," (2008). http://www.mcafee.com/us/local_content/technical_briefs/wp_zero_day_to_real_time.zip
30 Chappell, David. "A Short Introduction to Cloud Platforms," (August, 2006). http://www.davidchappell.com/CloudPlatforms--Chappell.pdf
31 http://blogs.technet.com/mmpc/archive/2009/04/28/windows-addresses-the-changing-autorun-threat-environment.aspx

About the authors

Vinoo Thomas *is a malware research lead with McAfee Avert Labs in Bangalore, India. His primary responsibilities include analyzing computer viruses, tracking global malware trends, and coordinating researchers. Thomas is a regular contributor to the McAfee Avert Labs blog and a columnist on computer security for the "Economic Times of India." He has several pending software patents and has published papers with EICAR, IEEE, and Virus Bulletin.*

Prashanth Ramagopal *is a research scientist with Avert Labs in Bangalore. He is an expert in reverse-engineering polymorphic malware and writing generic detection. When not fighting malware, Ramagopal follows his passion for photography.*

Rahul Mohandas *is a virus research engineer with Avert Labs in Bangalore. He pursues malware and vulnerability research, and frequently analyzes malware trends and exploits on the Avert Labs blog. Mohandas has presented at security conferences around the world.*

McAfee, Inc.
3965 Freedom Circle
Santa Clara, CA 95054
1.888 847 8766
www.mcafee.com