# MALWARE ANALYSIS 2

## FANS LIKE PRO, TOO

*Peter Ferrie*
Microsoft, USA

There are all kinds of amazing things that can be done in JavaScript, especially when the size is constrained, such as playing the 1KB game 'Mine[love]craft'. However, when you take the size-optimization techniques from there, combine them with structure and variable-name obfuscations, cram in every malicious action that comes to mind and, of course, have no limit on the file size, then you can end up with something that looks like JS/Proslikefan.

### WMF-WTF?-GQ

The virus begins as a wall of text, using no unnecessary whitespace (so the entire script is a single line of nearly 46KB characters in length). It uses random-looking variable names that are all eight characters long (or seven characters, for particular objects) and which differ only in the fifth and sixth characters (or just the fifth character for the seven-character version), making it difficult to tell them apart. As a result, we end up with lines like 'wmfyefgq+wmfywgq[90]+wmfyipgq+wmfygpgq(wmfyrsgq(wmfyoegq,wmfybjgq))+wmfykigq' (quick, how many unique variables are there?).

The virus uses other size optimizations, such as '!0' to replace 'true' and '!1' to replace 'false', exponent form instead of large numbers (e.g. 36e5 instead of 3600000 to represent one hour), and avoids semicolons as much as possible by using commas instead. The use of commas even extends to the return statement, where the virus places multiple assignment lines prior to the actual return value. One thing to note, though, is that every line has a purpose. There are no garbage instructions in the code at all. The obfuscation is strictly to make the reading difficult, rather than to mislead the reader.

The code begins like this:

```
(function(wmfyddgq,wmfynygq){wmfyqqgq="",...})(funct
ion(){return window},functi
on(wmfydvgq,...){wmfyilgq=...}(...,function(wmfygzgq
){...},...);
```

This can be 'simplified' to

```
(function(){})(),function(){}();
```

The line declares two anonymous functions, and invokes first the left one and then the right one. The first function is declared as accepting two parameters, which are defined during the invocation. The parameters are both anonymous functions, too. The first parameter function returns the name of an object ('window'). The second parameter function accepts one string parameter and splits the string into an array of its individual characters (not shown).

## WINDOW OF OPPORTUNITY

The virus executes the first parameter function, and attempts to access the 'window' object. The access is performed inside a protected block so that the virus can intercept any error that occurs. The virus is expecting an error to occur (because the object does not exist), and will not proceed correctly if the error does not occur. Thus, the virus cannot run from a web page. This might also serve as an anti-emulation trick in some environments.

If an error occurs, the virus uses the second parameter function to split a long string into its individual characters. The virus iterates through the characters in the resulting string, assigning one character to each of 12 variables until the entire string is decoded. Instead of using an 'if(condition)<body>', the virus uses a feature of JavaScript that is relatively little-known, but which is used very heavily in the js1kb demo world, where a Boolean evaluation that returns false will short-circuit the rest of the line in the case of an 'and' combination, and the true case will do the same for the 'or' combination. So, for example, instead of the following (which will perform the addition and assignment only if the length of wmfypxgq is not equal to eight):

```
if(wmfypxgq.length!=8)
        wmfypxgq+=wmfydagq,
        wmfydagq=wmfykngq()
```

the virus uses this:

```
wmfypxgq.length!=8&&(wmfypxgq+=wmfydagq,wmfydagq=wmf
ykngq())
```

JavaScript will evaluate the left half ('wmfypxgq.length!=8') and while the condition is met (that is, while the length is not equal to eight), it will execute the code in the right half (append the current character and fetch the next one). It should also be noted that the use of the comma allows the virus to omit the braces that would normally surround a multi-line body. This use of commas appears fairly consistently throughout the virus code. The use of the conditional shortcut, on the other hand, is highly erratic. This might suggest that multiple authors were involved, or perhaps just one author displaying different stages of development of the code.

The decoded strings are 'toString', 'charAt', 'charCodeAt', 'sort', a fake decryption key for the second text (see below), 'constructor', a base64-encoded encrypted string, 'fromCharCode', a base64 dictionary, a real decryption key, 'apply' and 'random'.

## RDA, SCRIPT STYLE

After decoding the strings, the virus invokes the second of the anonymous functions in the array (the one that begins 'function(wmfydvgq,...'). This function generates up to 1,221 unique base-20 values to use as part of a decryption key for the decoded base64 strings. Each unsuccessful value is placed in an array so that it will not be used again. In the event that the key is not recovered after 1,221 attempts, the virus exits silently.

After each attempt at decrypting the text, the virus tries to run the resulting code. Instead of using the 'eval' function, or just declaring the code as a function and running it, the virus uses the 'array.sort.constructor' trick. This trick is derived from a way of obtaining a function reference by using only the alphabetic characters that can be generated using the minimum number of symbols (see the description of JJEncode [1] for the details). It has no special use in this context, since the virus has access to all possible characters. It is included simply to obfuscate the code further.

If the text has been decrypted correctly, the virus attempts to access the 'document' object. The access is performed inside a protected block, so that the virus can intercept any error that occurs. Once again, the virus is expecting an error to occur (because the object does not exist), and will not proceed correctly if that does not happen. This might also serve as an anti-emulation trick in some environments.

If an error occurs, the virus attempts to access the 'WScript' object. This access is also performed inside a protected block, so that the virus can intercept any error that occurs. However, in this case, the virus is not expecting an error to occur, and will not proceed correctly if one does. Specifically, if an error occurs, the virus fails to assign the real decryption key for the second text.

If the second text is decrypted correctly, the result is a block of code that is packed by Dean Edwards' JavaScript packer. This packer has remained enormously popular since its release in 2005, the 2007 release in particular – despite being outperformed by later packers such as JSCrush.

In any case, after unpacking and 'beautifying', we are left with a script of over 1,650 lines of dense code. There are no comments or blank lines. The code is a collection of 68 anonymous functions, some of which accept yet more anonymous functions as parameters, and some of which are not even used, such as the function to extract data from cookie files. There is no reason for such a large number of functions, other than to make the analysis more difficult.

The virus uses RC4 to decrypt an enormous array of strings, many of which are small enough to have been used as constants within the virus body, but again, they serve to make the analysis more difficult. The virus then attempts to instantiate several objects: 'WScript.Shell', 'ADODB.Stream', 'Scripting.FileSystemObject', 'shell.application' and 'MSXML2.ServerXMLHTTP.6.0', and exits if any of them cannot be loaded.

## ANTI-VM

The virus constructs nine arrays containing different groups of strings:

- One contains process names that the virus will attempt to terminate.
- One contains a word list that could be used as a dictionary attack, but which is not used by the virus.
- One contains a set of host names that the virus contacts in order to send and receive information.
- One contains a list of registry values relating to security policies.
- One contains a list of registry values relating to the *Windows* Security Center.
- One contains a list of domain suffixes which are used during URL generation.
- One contains a list of registry values relating to the *Windows Firewall* and the use of proxy servers.
- One contains a list of registry values relating to SafeBoot.
- One contains a list of registry values relating to the display of hidden files.

The virus uses the *Windows* Management Instrumentation interface to query the system configuration, as a virtual machine detection technique. The virus looks for a SCSI controller whose manufacturer name contains either 'Xen' (which appears twice in the list, perhaps in a copy-and-paste error, which suggests that the intended target is missing), 'Citrix', or 'Red Hat'; a BIOS whose manufacturer name contains 'innotek', 'Bochs', 'Xen', or 'QEMU'; a disk drive whose model name contains 'Bochs', 'VBOX', 'QEMU', 'Red Hat', 'VMware', 'Virtual HDD' (this is a typographical error – *VirtualPC*'s hard disk is named 'Virtual HD', and so the virus runs freely in *VirtualPC*), or 'Xen'; a process named 'CaptureClient.exe' (part of the *Capture* honeypot project); a computer system whose manufacturer name contains 'Parallels'; a processor whose manufacturer name contains 'Bochs' or 'QEMU'; or a computer name that contains either 'mcafee' or 'cnc-lab'. The matching of the computer name is case-insensitive. The virus exits if any of these is found.

## FEELING INSECURE

The virus alters the registry to enable the hiding of files that have the hidden or system file attribute set. This is achieved by setting the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Hidden' registry value to the number 2. The virus constructs a registry value and data for writing. The registry value begins with 'lm', followed by two to five hexadecimal digits which are constructed in a convoluted fashion. The virus applies RC4 to the computer name and the 'lm' string, converts the result to a string of hexadecimal digits, and extracts some of the digits from the result. Thus, the value looks random but is actually constant on the given machine. An example of the registry data format is: 'C:\Program Files\[2–5 hex digits, but using 'ml' instead of 'lm' as the RC4 parameter]\[2–5 hex digits, using 'lm' as the RC4 parameter].js'. The virus attempts to write to the registry, but fails to specify a root, so the value is not created. This is a bug in the virus code.

The virus creates the registry value 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run\[2–5hex digits, using 'cu' as the RC4 parameter]'. The data is set to the Application Data directory, for example, 'C:\Documents and Settings\me\Application Data\[2–5 hex digits, using 'uc' as the RC4 parameter]\[2–5 hex digits, using 'cu' as the RC4 parameter].js'.

The virus attempts to make many other changes to the registry – some of which are successful and some of which fail. It disables the *Windows* Security Center notifications by deleting the WSC registry value. It attempts to disable SafeBoot by deleting the registry key, but there is a bug in this code, and the attempt fails. It attempts to delete the SafeBoot registry key from HKCU, even though there is no 'System' hive in that location. It does, however, disable the *Windows Firewall* and the use of proxy servers, by changing their options in the registry. This is achieved by setting the 'HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\EnableFirewall' registry value, the 'ProxyEnable' and 'MigrateProxy' registry values under the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings' registry key, and the 'HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ParseAutoexec' registry value to zero.

The virus disables the *Windows* Security Center service by changing its start option in the registry. This is achieved by setting the 'HKLM\SYSTEM\CurrentControlSet\Services\wscsvc\Start' registry value to the number 4. The virus disables notifications in the *Windows* Security Center from the anti-virus and firewall services, and enables overrides for them. The virus is aware of the changes in registry layout between *Windows XP*, *Windows Vista* and later. For *Windows XP* compatibility, the virus achieves the effect by setting the 'UpdatesDisableNotify', 'FirewallDisableNotify', 'AntiVirusOverride', 'FirewallOverride' and

'AntiVirusDisableNotify' registry values under the 'HKLM\SOFTWARE\Microsoft\Security Center' registry key to the number 1. For *Windows Vista* and later compatibility, the virus achieves the same effect by setting the 'AntiVirusDisableNotify', 'FirewallDisableNotify' and 'FirewallOverride' registry values under the 'Security Center\Svc' registry key to the number 1.

The virus disables access to the command-interpreter, registry tools such as regedit, Task Manager, and the 'Display' option in the *Windows* Control Panel. This last one seems curious until you see that its name is 'NoDispCPL'. It seems likely that the virus writer thought that it meant 'No Display Control PaneL'. The effect of all of these is achieved by setting the 'DisableCMD', 'NoDispCPL', 'DisableRegistryTools', and 'DisableTaskMgr' registry values under the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies' registry key to the number 1.

The virus disables access to the 'HomePage' setting in the *Internet Explorer* control panel. This is achieved by setting the 'SOFTWARE\Policies\Microsoft\Internet Explorer\Control Panel\HomePage' registry value under both the 'HKCU' and the 'HKLM' registry hives to the number 1. The virus disables infection reporting from *MSRT*. This is achieved by setting the 'HKLM\SOFTWARE\Policies\Microsoft\MRT\DontReportInfectionInformation' registry value to the number 1. The virus disables the System Restore configuration. This is achieved by setting the 'HKLM\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore\DisableConfig' registry value to the number 1.

The virus disables the *Windows* Control Panel, the 'Windows Update' option, and the 'Folder' option from within *Windows Explorer*. This is achieved by setting the 'NoControlPanel', 'NoWindowsUpdate' and 'NoFolderOptions' registry values under the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer' registry key to the number 1.

The virus enables the hiding of known file extensions in *Windows Explorer*. This is achieved by setting the 'HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\HideFileExt' registry value to the number 1. The virus disables System Restore. This is achieved by setting the 'HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestoreDisableSR' registry value to the number 1.

## HOT PROSPECTS

The virus checks whether the directory 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]' exists, and creates it if it does not. The virus sets the hidden and system file attributes on the directory in any case, and remembers if the directory was newly created. This state is checked later, and is used to decide whether the visible payload will execute. The virus attempts to open a file in that directory, whose name is '[2–5 hex digits, using 'it' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and writes to it the number of seconds since midnight on 1 January 1970. This could be considered the 'install time'.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'r' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus opens its own file, reads it entirely, and then searches for what happens to be the last line in the virus code. This line is a long sequence of hexadecimal digits. The virus extracts 24 characters from the middle of the line, and uses it as a key to decrypt another string. The virus creates the originally requested file, and then writes the decrypted string to it. This might be a 'revision' number.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'id' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and then writes a string of 12 random hexadecimal digits to it, converting to upper case if necessary. This is a machine-specific 'ID' that is used to communicate with the command-and-control server.

The virus attempts to open a file whose name is '[2–5 hex digits, using 'v' as the RC4 parameter]'. If the file can be opened, then the virus reads it entirely. Otherwise, the virus creates the file, and writes the virus filename to it.

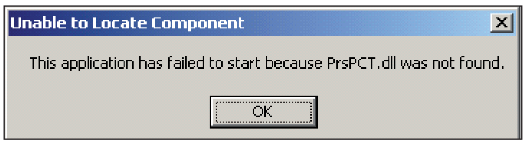
## START ME UP

The virus enumerates files in the 'startup' directory for all users. The virus is aware of the different locations of that directory between the different versions of *Windows*. On *Windows XP* and earlier, it is '%userprofile%\Start Menu\Programs\Startup\'. On *Windows Vista* and later, it is '%userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\'. The virus deletes any '.js' files that exist in the directory, apart from any file whose name matches '[2–5 hex digits, using the current hour of the day as the RC4 parameter].js'. The virus opens its own file and reads up to the last line, calculates the new key to place in the last line, and then writes the combination to the 'startup' directory.

As a side note, the path of the startup directory is constant, no matter which locale is active, despite appearances to the contrary. Specifically, when viewing the directory in *Windows Explorer*, the name is localized so that, for example, the French version of *Windows* will show 'Menu Démarrer'. This should be obvious to a programmer, given that there is no API to retrieve the path to the startup directory.

The virus creates the '%appdata%\[2–5 hex digits, using 'uc' as the RC4 parameter]' and '%programfiles%\[2–5 hex digits, using 'ml' as the RC4 parameter]' directories, as referenced above, and then hides them. It creates the '[2–5 hex digits, using 'lm' as the RC4 parameter].js' file in the Program Files hidden subdirectory, and the '[2–5 hex digits, using 'cu' as the RC4 parameter].js' file in the Application Data hidden subdirectory.

If the 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]' directory was newly created, then the virus copies itself to '%temp%\[12 random hexadecimal digits].js', runs that copy, and then displays the following message:




The message will remain on the screen for 30 seconds, and then the original copy of the virus will exit, leaving the one in the temporary directory still running. Otherwise, the virus waits for a random amount of time, from slightly less than one second up to almost ten seconds, before continuing with the execution.

If the 'C:\[2–5 hex digits, using 'prospect' as the RC4 parameter]\[2–5 hex digits, using 'lock' as the RC4 parameter]' file exists, then the virus opens the file, reads it entirely, and checks that it contains only numbers. This file contains the date and time of the most recent execution of the code. The virus exits if the last execution was less than 15 seconds ago, since this is an indication that another copy is actively running. If the file does not exist, then the virus creates the file and writes the current time to it.

## LOCK, STOCK, BARREL

The virus enumerates the CPUs and creates an array of the CPU names and number of cores. It also enumerates the video cards and creates an array of the video card descriptions.

The virus randomly reorders its list of hostnames, and then begins to enumerate the entries in the list. For each of the hostnames (currently: 'copertps.com', 'specrtop.org' and 'etpsoprc.ru'), the virus attempts to contact the host, send it a specific base64-encoded RC4-encrypted string, and receive another string in return. If a string is returned, the virus decodes and then decrypts it. If the resulting string contains the word 'prospect', then the host is accepted and will be used for any further requests for the next hour. If no string is returned or it does not decode correctly, then the virus continues the enumeration. If no acceptable host is found, then the virus generates a collection of URLs algorithmically, orders them randomly, and then attempts to contact each of the first ten in turn. For each of the algorithmic hostnames, the virus attempts to contact it and send it the specific string, as described above. If the proper string is returned, then that host will be used for the next hour.

The algorithm for URL generation is as follows: for each of the domain suffixes ('ru', 'net', 'info', 'in', 'eu', 'org', 'com', 'se', 'biz' and 'name'), the virus constructs a string in the format: 'prospect'.<month>.<date>.<four-digit year>.<domain suffix>. The virus hashes this string using a simple home-made algorithm, and then creates a new string of six to 12 lower-case letters, followed by the domain suffix. The virus constructs up to 10 unique URLs per domain, resulting in an array of potentially 100 entries (the count will be fewer if the hashes of any two of the URLs are identical).

If the host has a directory named 'u', then the virus fetches an update to its code from that location, and replaces the file containing the running script. However, the virus does not run this new file, so the update is not applied until later.

## GETSYSTEMINFO()

Once per hour, the virus looks in the 'Application Data' and 'Appdata\Roaming' directories for each user, for the files named 'sitemanager.xml' and 'recentservers.xml' in a directory named 'FileZilla'. The virus reads either (or both if present) file in its entirety, and extracts some interesting properties from the files: host name, port, communication protocol, user name, and password. This information is uploaded to the 'r' directory on one of the hosts or generated URLs, as described above.

Once every 30 minutes, the virus calls a routine which now simply returns. However, enough of the code remains to determine that it would have uploaded files that were downloaded from a *WordPress*-hosted website. It would also have uploaded audio, video, graphical, and archive-format files that were requested from websites such as *Pinterest*, *Twitter* and *Sourceforge*.

The virus will, however, upload the complete system information to the 'k' directory on one of the hosts or generated URLs, as described above, and possibly receive a response containing commands to run. The information that is sent is:

- the uptime as measured in approximately 30-minute intervals
- a magic number that might identify the exact version of the virus

- the list of CPU names (see above)
- the computer name
- the number of CPUs
- the list of video card descriptions (see above)
- a value corresponding to the anti-virus software that is installed (see below)
- the account name for the logged-on user
- the current time zone
- a copy of the virus body
- a 'random' value (system-specific, as described above)
- the country code (see below)
- the *Windows* version
- the execution state of a particular process
- a virus-generated ID (see above)
- the processor architecture (32-bit or 64-bit)
- the language code (see below)
- the local time
- the special code that is appended to the virus body.

The virus determines which anti-virus software is installed by checking for the existence of the following directory names in the Program Files directory, and assigns each one a unique value:

| | | |
|---|---|---|
| Kaspersky Lab | Sophos | F-Secure |
| Spyware Doctor | Webroot | Avira |
| Panda Security | McAfee | ESET |
| Microsoft Security Essentials | Bitdefender | Sunbelt |
| Alwil Software | Symantec | COMODO |
| Microsoft Security Client | Trend Micro | AVG |
| AVAST Software | DrWeb | |
| Malwarebytes' Anti-Malware | | |

The virus uses the *Google* Geolocation services to determine the country code for the host IP address. The API in question has been deprecated since 2010, but continues to be available for a limited number of requests. In the case of the virus, it needs to make only one request.

The particular process that interests the virus is an .exe file with a name which the virus generates by using the key 'btcm'. If the process is found to be running, then the virus sets the priority to run only when the system is idle.

The language code is determined by requesting the language version of the operating system, and then looking up the corresponding entry in the RFC1766 MIME database.

## COMMAND AND CONQUER

The virus can receive a list of commands to execute. The commands are very short: 'e', 'hp', 'r', 'd', 'fbc', 'dbs', 'b', 'u', 'fbl', 'redu' and 'fbf'.

The 'e' command can be used to run arbitrary script code where the results are not checked.

The 'hp' command is intended to be used to redirect all URL connections to the requested site. This would be achieved by placing the site name in the appropriate protocol under the 'Prefixes', 'DefaultPrefix' and 'Prefixes\www' registry keys under the 'HKLM\Software\Microsoft\Windows\CurrentVersion\URL' registry key. However, there is a bug in this code, which means that the command does not work.

The 'hp' command can set the Start Page in *Microsoft Internet Explorer*. This is achieved by setting the 'Software\Microsoft\Internet Explorer\Main\Start Page' registry value in both the 'HKCU' and the 'HKLM' hives. The command can optionally change the start page in *Google Chrome*. This is achieved by changing the appropriate settings in the '%userprofile%\Local Settings\Application Data\Google\Chrome\User Data\Default\Preferences' file. If the *Chrome* option is selected, then *Mozilla* will be targeted, too. The virus searches for the 'user.js' file in the subdirectories of the '%appdata%\Mozilla\Firefox\Profiles' directory. If the file is found, then the virus will change the start page in that file.

The 'r' command can be used to run any executable files on the local system.

The 'd' command can be used to download and run a specified file from a specified URL. The virus will contact the server and wait up to approximately seven seconds for a response.

The 'fbc' command was probably a routine used to start a chat on *Facebook*, but the code is not present in this version of the virus.

The 'dns' command can be used to change the DNS server on the local system. This is achieved by changing the 'DhcpNameServer' and 'NameServer' registry values under the 'HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters' registry key.

The 'b' command can be used to download an .exe file whose local name the virus generates by using the key 'btcm'. The virus will download the file only if the parameters for its execution are different from the previous execution, if any. If the download is requested, then the virus stops the existing 'btcm' process, if it is running, and then downloads and runs the new file. The virus intends to return the state of execution of the new file, but there is a bug in this code so the execution state always appears to be a failure.

The 'u' command can be used to update the virus code dynamically. If the virus has been updated successfully, then it clears the file that holds the last execution time, in order to allow the new code to start running without failing the '15 seconds' check.

The 'fbl' command was probably a routine used to 'Like' a page on *Facebook*, but the code is not present in this version of the virus.

The 'redu' command can be used to run arbitrary script code that accepts a single parameter, for example solving equations. The results will be uploaded to the 'reduce' directory on one of the hosts or generated URLs, as described above.

The 'fbf' command was probably a routine used to become a fan of a *Facebook* page, or to send a 'friend' request, but the code is not present in this version of the virus.

After all commands have been processed, and if the 'b' command has not been received, the virus stops the 'btcm' process, if it is running. It is unknown what this process does.

The virus periodically spends five seconds alternating between sleeping for one second and enumerating the list of running processes. The virus attempts to terminate any process whose name contains any of the following strings:

| | | | |
|---|---|---|---|
| rubotted | avg | avast | autoruns |
| tcpview | msconfig | hijack | otl |
| fs20 | msss | filemon | minitool |
| systemlook | mrt | jrt | wireshark |
| unlocker | procmon | mse | sdasetup |
| mbam | clean | rkill | ccsetup |
| resmon | procexp | fss | rstrui |
| housecall | ptinstall | npe | wuauclt |
| mcshield | sdefendi | regmon | issetup |
| mbsa | fiddler | avenger | gmer |
| roguekiller | dds | emergencykit | exeradar |
| avenger | hitman | combofix | perfmon |
| reged | spybot | klwk | eset |
| windows-kb | hotfix | zoek | |

In all cases except for the 'hotfix' entry, the matching is case-insensitive. The case-sensitivity of the hotfix entry appears to be a bug in the virus code.

## AUTORUN.INFECT

The virus repeatedly enumerates the list of drives, waiting for a USB device to be inserted. When a USB device is found, the virus creates a directory on each of its drives, '\[2–5 hex digits, using 'usb' as the RC4 parameter]', and then hides this directory. The virus places a file inside the directory, '\i[2–5 hex digits, using 'lnk' as the RC4 parameter].js'. For every other directory in the root of the drive, excluding any named 'recycled', the virus creates a shortcut using the name of the directory followed by '.lnk'. The virus then hides the original directory. The icon for the shortcut is the folder icon, but the shortcut arrow is added to the corner of it (there is a registry change that can make it go away, but the virus does not make use of it). In any case, the action of the shortcut is to run the virus script, and then open an *Explorer* window showing the contents of the directory.

The virus places another file inside the hidden directory, '\g[2–5 hex digits, using 'ar' as the RC4 parameter].js', and then creates an 'autorun.inf' in the root directory of each of the drives on the USB device. The virus writes a random number of lines (from 35 to 100) of random text. For each of those lines, there is a 20% chance that the virus creates a section with a random name. The name is a random number from five to 10 characters. Otherwise, the line is an assignment using a random number from 10 to 30 characters on each side of the equals sign. Then the virus alternates between writing five to 10 random lines and one real line. The order of the real lines ('shell\explore\command=', 'shell\open\command=', 'open=' and 'shellexecute=') is also random. After all of the real lines have been written, the virus writes another random number of lines (from 15 to 50) of random text. For each of those lines, there is a 20% chance that the virus creates a section with a random name. Otherwise the line is an assignment, as before.

## CONCLUSION

One of the main problems with describing code that can update itself is that no two descriptions will be alike. The code could update itself in different ways, depending on certain circumstances – for example, different countries might be served different versions. Even requests at different times of the day might yield different results. The best that we can say is that 'this sample, with this hash value, behaves in this way' – and that's not saying much. Fortunately, the different variants that we have seen have a similar overall structure, which allows us to detect them generically. That's all we need to say.

## REFERENCES

[1] Ferrie, P. $$$_+$$+$$__+_$+$$_$+$$$_+$$_$. Virus Bulletin, February 2011, p.4. http://www.virusbtn.com/pdf/magazine/2011/201102.pdf.