The Use of File Size Parity of Windows .exes and .dlls as a Malware Indicator of Compromise

Peter Bentley, School of Computing and Engineering, University of Gloucestershire, Cheltenham, UK

***Abstract* –** To gain persistence on Windows machines, some Advanced Persistent Threats (APTs) hide their malware in plain sight as standalone files. It is inferred from Microsoft documentation that Portable Executable (PE) file length should be even. This paper documents the analysis of *.exe* and *.dll* file length parity on three versions of Windows operating systems. It uses simple techniques to analyse the parity of .exe and .dll files and demonstrates that not all are of even length file. This may be used as an indicator of compromise and found such on one machine.

Keywords –Microsoft Windows; File Size; File Length; *exe*; *dll;* Portable Executable (PE); Advanced Persistent Threat; APT; Malware; Indicator of Compromise.

I Introduction

Microsoft documentation states that Portable Executable (PE) file lengths should be even [1]. This paper demonstrates that source code does not always compile to a PE of even length. This may be because: although Microsoft and suppliers ensure that most PEs compile to an even length at least part of the supply chain compiles code in a different way or with a different compiler or some commercial developers or APTs compile source code in a different way to Original Equipment Manufacturers (OEM.

A simple technique for analysing the distribution of file sizes on Windows based machines is demonstrated and from this malware has been uncovered.

Analysis of file length distributions has been performed before [2] but no analysis of .exes and .dlls could be found.

No bespoke software was written for this paper. The analysis was carried out using freely available Microsoft and OpenOffice software. This paper presents analysis of the parity (last digit) of file sizes. The technique found a suite of programs that were not authorised to be on the machine. The paper concludes with a discussion a critique of the work and suggestions for further research.

II Background

Microsoft describe the format of PE files as follows. The Sizeof image field " ... must be a multiple of SectionAlignment " which in turn "… must be greater than or equal to FileAlignment. The default is the page size for the architecture." [1]

It appears, therefore, that file sizes are based on multiples of the field FileAlignment which "… should be a power of 2 between 512 and 64 K, inclusive. The default is 512." One can immediately see that this will produce an even length file size. The key phrase here may be "should be a power" but when SectionAlignment is greater than FileAlignment, and an odd number, the file size could end up as an odd number.

Other information from Microsoft is mixed: Pietrek [3] does not shed light on a PE file size but Pietrek [4] alludes to platform differences "You should only need to use the 32 or 64-bit specific versions of the structures if you're working with a PE file with size characteristics that are different from those of the platform you're compiling for."

Pietrek [4] also states that "The distinction between *exe* and *dll* files is entirely one of semantics. They both use the exact same PE format. The only difference is a single bit that indicates if the file should be treated as an *exe* or as a *dll*." This paper assumes that this distinction still holds.

It is noted that Microsoft reserves at least one field in the PE format for Borland [5].

III Data Collection

Two methods of data collection were used: the first used the Microsoft command "*forfiles*" [6] on Windows 8 and 10 operating systems; the second used the "*dir*" [7] command on Windows XP. This is because "*forfiles*" was not available for the XP machine and it was decided not to download it. Also, the different ways of data collection provide some independence assurance that the results are valid.

Data was collected from the Windows 8 and 10 machines when new. Data was also taken from the Windows 10 machine when it had been heavily used.

A one-line batch was written which contained the command "*forfiles*". This command selects and executes a command on a file or set of files and is useful for batch processing. This batch recursively outputs, one item per line, a list of all *.exe* files with associated size

and full path name. The batch was modified for *.dll* in place of *.exe* and was run on Windows 7, 8 and 10. The command used within the batch is:

*forfiles /p c:\ /s /m *.exe /c "cmd /c echo @fsize @path"*

The output was written to respective text files and then imported into separate spreadsheets from where the file length parity (i.e. is it odd or even?) was derived.

Another one-line batch was written which contained the line:

Dir C:\ /S [7]

Again, the output was written to text files and then imported into a spreadsheet. However, in this case a lot of hand editing and sorting was needed to produce data in the required format. Spreadsheets used were Microsoft Excel and the OpenOffice equivalent.

The author had administrator and user rights for each machine. Even so some directories were not available for analysis and so the analysis of the total number is incomplete.

IV File Size Last Digit Analysis (to test odd/even parity of the file size)

Table 1: Count of File Length Modulo 10

OS and Filetype	File length Modulo 10 Count									
	0	1	2	3	4	5	6	7	8	9
W-XP.dll	2108	113	2331	104	2229	73	2094	66	1952	135
W-XP.exe	477	29	581	22	616	14	587	26	777	30
W-8.dll*	2808	0	3161	0	2657	0	2659	2	2643	4
W-8.exe*	428	0	396	0	434	0	418	0	413	0
W-10.exe*	609	5	660	11	651	8	659	9	654	16
W-10.dll	16481	2334	14596	2140	14994	2134	14944	2152	15448	2232
W-10.exe	1076	183	1041	204	1092	169	1047	192	1027	172
W-10.dll+	10501	3960	10509	4182	10620	4059	10106	3862	10304	3941
W-10.exe+	1430	538	1408	661	1387	671	1469	589	1358	601

(* New or almost new builds. Data for Windows 10 executables were gathered when this paper was an idea and not a fully formed piece of research, hence no data for Windows 10 new build .dlls)

(+ Operational frozen system)

It can be seen that there is at least a deficit of odd length files and for new builds a dearth.

However, if there were a non-biased odd/even split of file lengths this should be reflected in the data. It is clear that there is a distinct odd/even split in file lengths. This observation is consistent with the work of Evans and Kuenning: "… file-size distributions are "polluted" by large collections of similarly sized files, such as icons or configuration files associated with a particular application." [8].

V Discussion and Application to Malware Discovery

It can be seen from Table 1 that there is a significant split between the counts of odd and even file sizes.

For the Windows 10 new build there are 3233 even length .exe files and 49 odd .exe files which, therefore make up just under 1.5% of all files. After a lot of use this percentage rises to just over 12.5%. It may be inferred from this that many updates and/or non-Microsoft .exes compile to an odd length.

Further analysis reveals that many of the odd length files have names that contain x86 (E.G. C:\Program Files (x86)), AMD or "uninstaller". Some are from the SysWOW64 Windows subdirectory. The existence of duplicates of files and files of similar name and identical length which are files for associated software.

The most significant discovery was the existence on one of the analysed machine of a suite of programs that had not been authorised for installation by the legitimate owner. Six .dlls from this suite had odd file lengths.

It may be inferred from the data that the compiler Microsoft is using in its supply chain compiles code to multiples of two for the file size (i.e. even length executables) but other compilers, and suppliers compile code to both odd and even length.

VI A Critique of this Work

No previous analysis of *exe* and *dll* files could be found and this, together with the results, led to the suspicion that perhaps there was a problem with the methodology: where could it have gone wrong and mistakes been made? The use of two different ways of collecting the data (*dir* and *forfiles*) means that it would be difficult to make the same mistake across all operating systems. Similarly, using two different spreadsheets (Microsoft Excel and OpenOffice) across operating systems provides independence. Digit extraction from the file length field is easily checked across the data sets.

This paper suggests that the use of two different sets of analysis software (Microsoft Excel and OpenOffice) provides an independent check for the methodology. The only commonality across both sets of analysis is the Microsoft program *forfiles*.

Another way to collect data would be to access the Master File Table directly from storage e.g. hard disk drive (HDD) or solid-state device (SSD).

VII Suggested Further Lines of Work

Not all Microsoft operating systems have been analysed. It is suggested that the analysis be done on all Microsoft operating systems, using a new build, under admin rights.

It is also suggested that the work be performed on other operating systems e.g. Linux.

VIII Concluding Remarks

There is sufficient evidence to suggest that the compiled length of files in Microsoft Windows operating systems are restricted to an even length by the OEM. It is possible that all compilers used by Microsoft have this feature as a default or a setting. This feature may, or may not, be present in different compilers used by the wider IT profession but clearly some of these compilers will compile *.exe* and *.dll* files to an odd length.

Although an even length file size does not rule out infection by malware, an odd length file size may indicate malware infection. This feature may be used as part of a wider indicator of compromise for APTs.

REFERENCES

[1] Microsoft (2018) PE Format Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/ms680547(v=vs.85).aspx (Accessed: 20th May 2023).

[2] Downey, A. B. (2001) 'The Structural Cause of File Size Distributions', *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* Cambridge, Massachusetts, USA: ACM New York, NY, USA. pp. 328-329. Available at: https://dl.acm.org/citation.cfm?id=378824

[3] Pietrek, Matt (1994) Peering Inside the PE: A Tour of the Win32 Portable Executable File Format Available at: https://msdn.microsoft.com/en-us/library/ms809762.aspx (Accessed: 20th May 2023)

[4] Pietrek, Matt (2002) An In-Depth Look into the Win32 Portable Executable File Format MSDN Magazine February 2002, Available at: https://msdn.microsoft.com/en-us/magazine/bb985992(printer).aspx (Accessed: 20th May 2023)

[5] Microsoft. (2018) PE Format. Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/ms680547(v=vs.85).aspx (Accessed: 20th May 2023)

[6] Microsoft (2018) forfiles Available at: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/forfiles (Accessed: 20th May 2023)

[7] Microsoft (2018) dir Available at: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/dir (Accessed: 20th May 2023)

[8] Evans, Kylie M. and Kuenning, Geoffrey H.. (2002) A study of irregularities in file-size distributions. In *Proceedings of the International Symposium on Performance Evaluation of Computer and TelecommunicationSystems (SPECTS)*, San Diego, CA. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.2569&rep=rep1&type=pdf