FORENSIC ENTROPY ANALYSIS OF MICROSOFT WINDOWS STORAGE VOLUMES

P. J. Weston* and S. D. Wolthusen†

* *Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*
E-mail: pete.weston@ntlworld.com
† *Norwegian Information Security Laboratory, Gjøvik University College, N-2818 Gjøvik, Norway and Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*
E-mail: stephen.wolthusen@rhul.ac.uk

Abstract: The use of file or volume encryption as a counter-forensic technique depends on the ability to plausibly deny the presence of such encrypted data. Establishing the likely presence of encrypted data is highly desirable for forensic investigations. We claim that the current or previous existence of encrypted volumes can be derived from studying file and volume entropy characteristics using knowledge of the development of volume entropy over time. To validate our hypothesis, we have examined several versions of the Microsoft Windows operating system platform over a simulated installation life-cycle and established file and volume entropy metrics. Similarly we verified the hypothesis that the ageing through regular use of an installation is identifiable through entropy fingerprint analysis. The results obtained and tests devised allow the rapid identification of several volume-level operations and also detect anomalous slack space entropy indicative of the use of encryption techniques.

Key words: File System Entropy, Installation Aging, Encrypted File Systems

1. INTRODUCTION

The use of volume encryption by itself is insufficient to keep data confidential or as a counter-forensic technique when access to key material can be obtained or enforced. This may occur e.g. pursuant to Part III of the *UK Regulation of Investigatory Powers Act 2000*, requiring that a suspect supply decrypted information and/or cryptographic keys to authorised government representatives [1]. A skilled adversary will hence aim to use a combination of cryptography and steganography[*]to achieve plausible deniability, whilst forensic investigators must identify the presence of encrypted volumes for further analysis as an in-depth manual inspection may not always be feasible.

Entropy is a measure of the amount of information present in a signal or file.[**]A low entropy measurement implies a well-ordered, well-structured signal whilst a high entropy measurement indicates a signal with little apparent order or structure. Encrypted information, however, must not show discernible order or structure lest this make the encryption vulnerable to various forms of statistical attack i.e. encrypted data must have high entropy. Measurements of entropy and randomness taken against a computer system should reveal some information about the current state of that system and the data stored on it even when file signature evidence has been removed or replaced.

The very presence of high entropy data on a system may therefore prevent a suspect from plausibly denying the presence of encrypted information. However, file compression may result in similar high entropy encoding, requiring a more careful analysis. We have hence sought to characterise the relative entropy found for encrypted and unencrypted (including compressed) data, but also the effects of utilising different operating system versions as well as usage patterns. The research reported in this paper aimed to determine experimentally the extent to which measures of entropy and randomness can differentiate between encrypted and unencrypted data, different computer operating system versions and configurations and typical and atypical computer usage. A large number of Microsoft Windows workstation installations are run through a simulated ageing process consisting of —

- applying patches and updates,
- installing and configuring applications, and
- creating and deleting large numbers of data files of known types.

Forensic images of each workstation were captured at key points during the ageing process and entropy and statistical randomness measurements were taken from each image for each storage volume and every installed image file. In order to verify the results obtained through simulation, a number of images captured from production workstations are analysed and compared.

As Microsoft Windows is by far the dominant platform in terms of desktop systems deployed, we focus here on this platform; analogue experiments for the Linux environment have, however, been conducted with broadly comparable results, only some of these are reported here in the interest of completeness. However, the

[*]see e.g. the United Nations Office on Drugs and Crime report "The use of the Internet for terrorist purposes", Sep. 2012 discussing steganographic mechanisms used for covert communication by terrorist entities with a case study involving the Revolutionary Armed Forces of Colombia (FARC).

[**]We rely on the standard Shannon entropy in the following.

Based on "Forensic entropy analysis of Microsoft Windows storage volumes", by P. J.Weston and S. D. Wolthusen which appeared in the Proceedings of Information Security South African (ISSA) 2013, Johannesburg, 14 to 16 August 2013. © 2013 IEEE

existence of several alternative file systems and more frequent significant release changes reduces the time series available for equivalent analysis. The remainder of this paper is structured as follows: In section 2. we briefly outline related work, followed by a description of the experimental setup in section 3. used for the subsequent analyses in sections 4. and 5. for file and volume analysis, respectively. We discuss key findings in section 6. before describing our conclusions and future work in section 7.

2. RELATED WORK

Research into signature and content analysis forms the basis of many file identification techniques, while work on multimedia data in particular is seen as vital to digital forensics. As this is a crucial pre-requisite for effective carving in the presence of fragmented or deleted data, file system behaviour allowing the effective grouping and identification of fragments has been studied by a number of researchers; Holleboom and Garcia investigated and performed experiments on information retention in slack-space for micro-fragments of previous files occupying the same clusters [2] with extensions by Blacher for the case of NTFS [3]; this also provides a further bound on the entropy of such clusters that is to be expected over the life-cycle of a frequently-reused storage medium.

A recent overview of the state of the art of multimedia forensic investigations is given by Poisel and Tjoa [4] while Ahmed *et al*. give examples of advanced methods used to improve file identification [5]. Shannon's analysis of ASCII and entropy scoring building on his namesake's work is of particular interest [6], as is recent work by Wu *et al*. showing how entropy and randomness testing can be used on encrypted images [7]; the tool `TCHunt` by 16 Systems identifies TrueCrypt images specifically by combining a search for volume (including sparse volumes) characteristics of TrueCrypt with a simple entropy analysis. Statistical analysis of file system clusters can yield insights on file types even for isolated clusters as discussed by Veenman [8]; for more specific file analyses, Lyda and Hamrock describe an entropy-driven approach for detecting encrypted malware, albeit relying only on block frequency (binning) to obtain a relatively coarse metric [9]. For the case of packed malware — which is beyond the scope of the present paper — this may not be sufficient if counter-forensic techniques are employed as recently described by Ugarte-Pedrero *et al*. [10]. This is also closely related to the need to predict the composition of file fragments; algorithms for which have e.g. been studied by Calhoun and Coles [11] with related approaches for classification described by Roussev and Garfinkel [12].

3. EXPERIMENTAL SETUP

The focus of the present work is, without loss of generality, on the Microsoft Windows operating system platform. Initially, eight production disk images were captured from pre-existing Microsoft Windows workstation installations using forensic capture tools. A total of 7 variants of Microsoft Windows 7, Vista, and XP were installed in default VMware Fusion 3.1 virtual machines (VM) using the VMware "*Easy Install*" wizard [13] at this time. At a later date, 2 variants of Microsoft Windows 7 and 8 were installed in default VMware Fusion 5.0 virtual machines and a Windows 8 production disk image was captured. The Boot volume of each installation was analysed. In all cases, the file system used was NTFS, and the cluster size set to eight 512 byte sectors. Except as noted, all storage sectors were zeroed prior to installation of the operating system instances. All images captured during this project were captured from Windows installations after the operating system had been shut down. Microsoft Windows workstation installations are categorised in this paper as either "Home" or "Business" depending upon the usage pattern and the installed applications and data. A common application suite — consisting of Microsoft Security Essentials, Adobe Acrobat and Flash Player, and VMware Tools — was installed on all virtual machine images. In addition, Oracle Java and Microsoft Office were installed on business VM images. Open Office, Mozilla Firefox and Thunderbird, Google Picasa, Apple iTunes and the Steam client application were installed on home virtual machine images to reflect different usage patterns. All applications were the latest versions at time of installation (June 2012 or September 2013). Document, music, picture, video and archive files were added to each virtual machine image; business virtual machine images contained relatively more document files, whilst home virtual machine images contained relatively more media files of different types. The relative proportions of each file type is the same in all cases but a different set of files were used for the later workstation images.

A total of 69 VM images were created through a simulated production life-cycle consisting of patching Windows, copying and deleting data files on the Boot volume, and exercising the installed applications. In ascending order of age, the simulated life-cycle stages are referred to in this paper as "*Initial*", "*Patched*", "*Base*", and "*Copyx*" (where *x* is the number of iterations). Captured images are referred to as "*Actual*" with a number identifying the image and a letter suffix indicating life-cycle stage where known ("a" is older than "b", etc.). Newer images are identified by the "new" suffix. So as to obtain images reflecting realistic longer-term use, the authors relied on images obtained from volunteers for validation. However, whilst all images and scripts utilised in obtaining the results described here can be made freely available on request, this does not apply to these validation images for privacy reasons. Encrypted data was obtained by creating TrueCrypt containers of various sizes using AES, Serpent and Twofish encryption algorithms and RIPEMD-160, SHA-512 and Whirlpool hash algorithms; AES with RIPEMD-160 was the default configuration. The same (weak) password was used in all cases, although all algorithms are sufficiently robust to eliminate influence on the entropy of encrypted data.

Data sectors were extracted from the volume images using tools provided in *The Sleuth Kit* [14]. Entropy and randomness calculations were performed on extracted data

	7-Zip File	7-Zip Cluster	TrueCrypt File	TrueCrypt Cluster
Count	237	1539	99	1024
Mean	7.999	7.955	8.000	7.955
Median	7.999	7.955	8.000	7.955
Min.	7.996	7.940	8.000	7.942
Max.	8.000	7.967	8.000	7.967

Table 1: 7-Zip and TrueCrypt Entropy (Bits/Byte)

at the byte level using the `ent` utility [15]. Where byte-level entropy calculations are impractical or inappropriate, data compression has been used as an analogue; all compression ratios reported in this paper result from GZIP compression at the "`-4`" compression level, utilising the Lempel-Ziv algorithm at its core [16], albeit utilising the DEFLATE format [17]. We note that not all applications were available for each version of the platform, resulting in some results not being available for all points of a time series; these data sets are reported as far as compatibility was retained. The results do not address possible changes of behaviour of users over time such as migrating from one application or platform to another as such behavioural changes and hypotheses underpinning behaviour were beyond the scope of the research.

4. FILE ANALYSIS

Entropy and randomness was measured at the byte-level for a statistically significant number of test files of various types: the mean and median number of files of each type analysed were 716 and 255 respectively with the minimum number of files of any one type being 11. Media and modern document file formats were found to exhibit high mean entropy in the range 7.270 to 7.981 bits/byte; most executables and older document formats exhibited lower mean entropy in the range 3.822 to 5.989 bits/byte. This reflects the use of improved compression algorithms in the newer file formats.

For file archives, entropy results reflect the relative performance of the compression algorithms used: LZNT1 shows the lowest mean entropy (947 files analysed, mean 7.558 bits/byte) and (G)ZIP and 7-Zip the highest mean entropy (96 files, mean 7.981 and 237 files, mean 7.999 bits/byte respectively). Encrypted TrueCrypt files consistently exhibit the highest possible byte-level entropy. TrueCrypt and 7-Zip clusters analysed in isolation, however, exhibit very similar (lower) entropy to each other (see Table 1).

χ-square, byte mean value, Monte Carlo π, and serial correlation values were calculated for all tested files. The χ-square test indicated that, with few exceptions, only 7-Zip and TrueCrypt files exhibit uniform randomness at the byte level. At the file level it was also noted that TrueCrypt containers consistently return χ-square results slightly closer to uniform randomness than the tested archive files; this does not apply when TrueCrypt containers are viewed at the cluster level (see Table 2).

	7-Zip File	7-Zip Cluster	TrueCrypt File	TrueCrypt Cluster
Mean	256.353	253.776	254.078	254.652
Median	258.247	253.375	254.073	253.750
Min.	190.077	182.750	204.452	185.500
Max.	324.218	340.375	322.160	326.500

Table 2: 7-Zip/TrueCrypt Chi-Square Statistics

4.1 Media File Analysis

Specific analyses were conducted for a number of file types including text, formatted text (XML, PDF), document (different Microsoft Office formats as well as Open Document format files), and media (image, video, and audio) data. Of particular interest in the context of the present paper are compressed file formats, which are characteristic of media data, but also more recent modern file formats noted above. Here, we have analysed basic descriptive statistics for file samples

	MP3	M4A
Count	6465	1088
Mean Entropy	7.967	7.981
Standard Deviation	0.041	0.011
Sample Variance	0.002	0.000
Kurtosis	334.617	80.982
Skewness	-15.283	-7.736
Minimum	6.710	7.822
Maximum	7.995	7.994
Confidence Level(95%)	0.001	0.001

Table 3: Entropy - Descriptive Statistics (Music files)

Table 3 shows that the compressed MP3 and M4A music file formats exhibit very high mean levels of entropy (similar to the entropy levels exhibited by compressed image file formats). Music file formats similarly also exhibit very little variance or deviation from their mean entropy value. For the music file types tested, χ-square randomness indicators have values well above values expected for random data (see Table 4). We can see easily that the tested music file formats do not exhibit randomness at the byte level; this is to be expected given the internal structure of the formats used, but yields a usable and efficient distinguishing feature. It is clearly necessary to perform this type of analysis as can be seen from the compression levels found in common file types; figure 1 provides a summary of the mean entropy for file types; this distribution is notably different once entropy is studied at the block or cluster level.

4.2 System File Analysis

Entropy and randomness values were calculated for Windows system and meta-data files at various points in the (simulated) Windows life-cycle. Entropy-frequency plots of the complete set of files forming each Microsoft operating system were found to be quite different to a

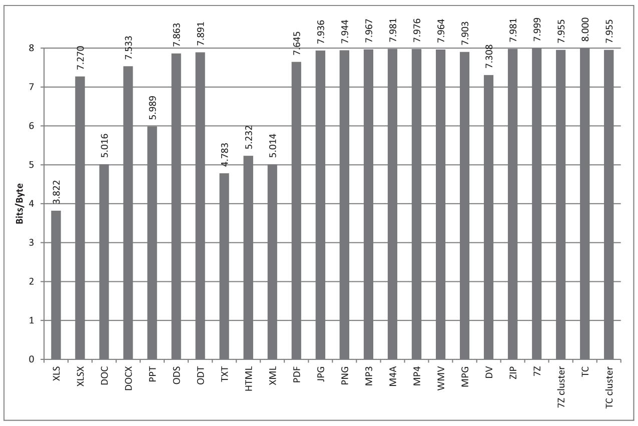

Figure 1: Overall File Mean Entropy (by Type). At least 1000 files per format were analysed.

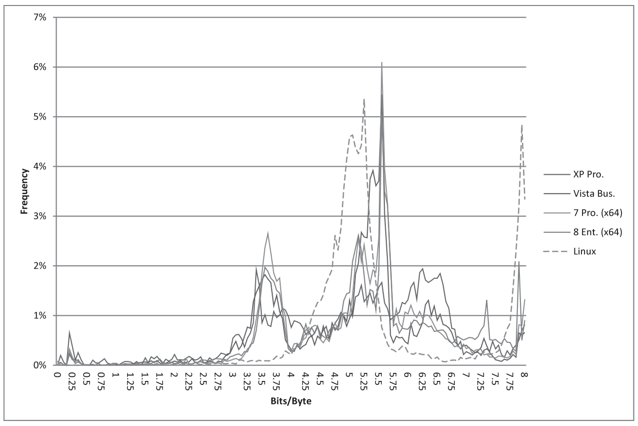

Figure 2: Operating System File Entropy (Initial Installation)

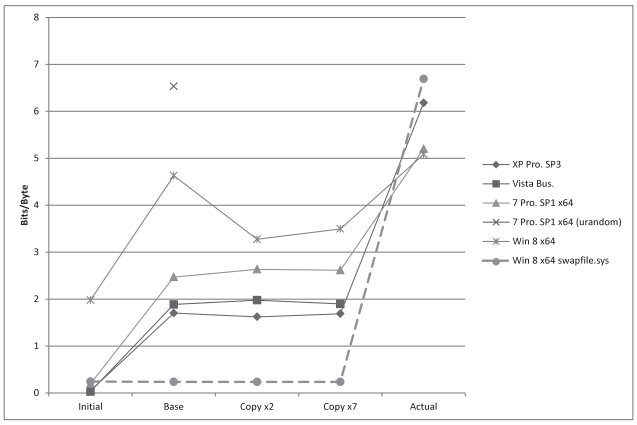

Figure 3: PAGEFILE.SYS File Entropy

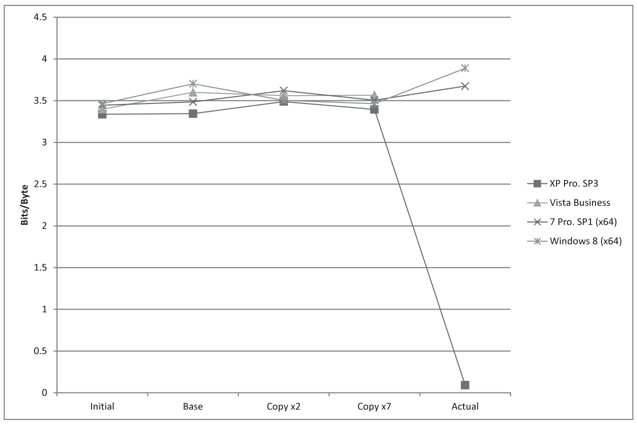

Figure 4: $LOGFILE File Entropy

similar plot of a recent Linux distribution. Figure 2 shows how different versions of Windows were observed to have relatively distinct entropy-frequency plots at initial installation. These distinctions tended to dissipate as data files were added to each installation although it was always possible to differentiate between Windows and Linux instances.

Windows uses the file `PAGEFILE.SYS` for virtual memory management. This paging file is not, by default, cleared when Windows is shut down [18] and hence entropy results can be expected to reflect data that has been swapped from main memory during Windows operation. Windows 8 "*Modern*" Apps use an additional `SWAPFILE.SYS` file to store their whole (private) working when suspended [19].

Figure 3 shows how the entropy of the Windows paging files generally increases over time. The test results indicate that the entropy of the paging file reflects the memory resources available on the host system. The Windows 8 (x64) test images, for instance, show significantly higher paging file entropy because they were generated on a virtual machine with half of the minimum memory requirement for this operating system. The greatest entropy of any paging file observed during testing was a "*Base*" Windows 7 image installed on a disk that had first been initialised by a UNIX urandom device. Paging files with unusually high entropy values may therefore be able to give some information about the history of a Windows system.

Windows 8 "*Modern*" Apps were not consciously exercised on the Windows 8 test images and consequently the Windows 8 `SWAPFILE.SYS` file exhibits extremely low entropy values on the test images. "*Modern*" Apps were, however, used on the production Windows 8 image and in this case swap file entropy is much higher. Entropy figures for the Windows 8 `SWAPFILE.SYS` file may therefore give some indication about the types of applications used on a Windows 8 sytem.

The NTFS journal attribute (`$LOGFILE`) is a fixed size circular log of 4 Kilobyte record pages where each journal page records the changes to be made to the file system [14, pp. 391-392]. Figure 4 shows that the NTFS journal entropy remains stable at around 3.5 bits per byte in most cases; this is to be expected given that the journal file is relatively small and journal pages are regularly reused. Note, however, that the Windows XP "*Actual*" journal contains few journal records and has an unusually low entropy. No explanation for this observation is available

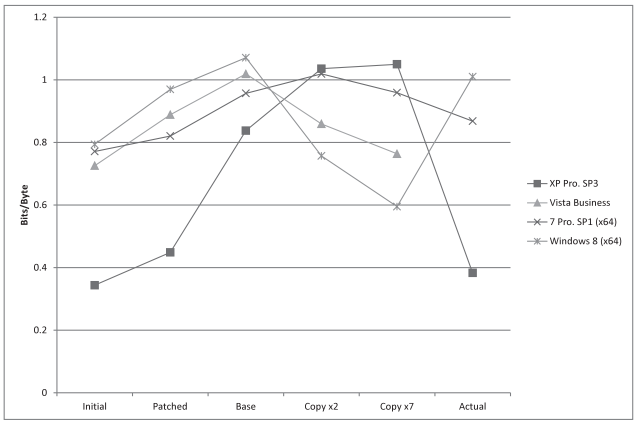

Figure 5: $BITMAP File Entropy

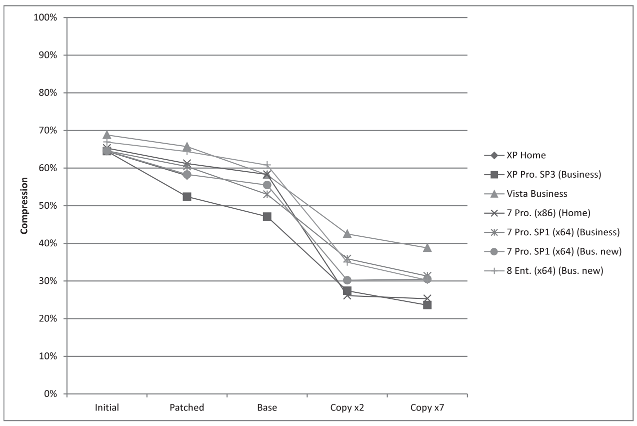

Figure 6: Boot Volume Allocated Space (GZIP)

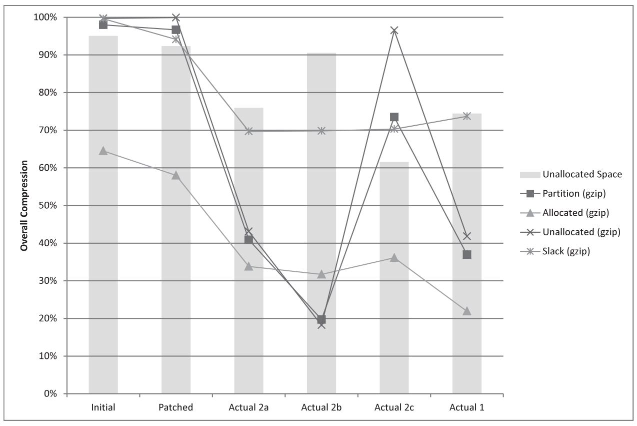

Figure 7: Boot Volume Compressibility (XP Home)

but such a significant deviation from the norm suggests that the journal file may be an area worthy of further investigation.

The entropy of the NTFS cluster allocation map attribute (`$BITMAP`) should remain low because Windows implements file allocation strategies that aim to minimise file fragmentation and performs regular de-fragmentation in the background. Test results confirm that the `$BITMAP` attribute has low entropy in all cases (see Figure 5).

5. VOLUME ANALYSIS

Storage clusters within the Windows Boot volume are flagged as either allocated or unallocated by the NTFS MFT `$BITMAP` attribute. Allocated clusters are known to contain current, live data whilst unallocated clusters may contain old, disused data. Sequence numbers within the NTFS MFT are incremented as MFT file and directory entries are (re)allocated and give an indication of the likelihood that unallocated and slack space will still contain the initial zeroed values. For the images described in this paper, maximum sequence numbers in the ranges 32 to 13015 and 11775 to 63655 were observed for the VM and production images, respectively.

Volume analysis was performed across the entire Boot volume. The MFT zone — a proportion of an NTFS volume reserved for MFT entries — is not considered in this analysis. When considering Windows XP volumes that have had more than 87.5% of their space allocated, it should be borne in mind that such volumes will have relatively fewer zeroed clusters than other XP volumes due to files having been allocated in the large MFT zone [20]. Figure 6 illustrates how the overall compressibility of allocated space on the Windows Boot volume decreases over time on all tested operating systems (i.e. entropy increases). The compressibility curve for all tested Windows versions begins to flatten in the 25–35% range as high entropy data files are added to the volume over time.

Overall compressibility of unallocated space on the Windows Boot volume decreases over time on all tested operating systems (i.e. entropy increases), as may be expected. Compressibility of unallocated space remains very high until used sectors containing high entropy data begin to be deallocated. The VM images show a smooth, gradual increase in the entropy of unallocated space as the Windows Boot volume ages. No such progression was seen on production images, however (see figure 7). The results for the "*Actual 2*" production Windows XP image demonstrate that it is possible for the entropy of unallocated space on a file system to be both lower or higher than allocated space entropy depending upon the usage history of the underlying media.

For the "*Actual 2*" image shown in figure 7, the allocated clusters from original media "*Actual 2a*" were copied onto media "*Actual 2b*" that had previously been used almost exlusively for media file storage. The file system was then subsequently copied onto new, zeroed media "*Actual 2c*". While both allocated and slack space compression ratios remained relatively constant during these relocations, compression ratios for unallocated space varied dramatically depending upon the original content of the new media.

The Microsoft NTFS file system stores data in fixed size allocation units called clusters. Files themselves, however,

	Chi-Square
MP3	9984.491
M4A	53119.803

Table 4: Minimum Chi-Square (Music files)

are seldom exact multiples of the cluster size and hence a certain proportion of the last cluster allocated to a file is not used and data could potentially be hidden there [21]. The unused space in a cluster is known as file slack. It is well-known that Microsoft Windows will fill unused space in the last sector into which data is written with zeros ("RAM slack"), but that it will not write data into any completely unused sectors in the cluster [14, 22].

Each file will therefore have one potentially lower entropy sector containing RAM slack plus potentially several further sectors which retain the data from whatever previously occupied them. In the case of a clean (zeroed) disk, therefore, slack space should overall have very low entropy because most of the sectors allocated to file slack are zeroed. Over time, however, as files are deleted and sectors are reallocated then the overall entropy of slack space should increase (although it should always remain comparatively low due to the zeroed RAM slack).

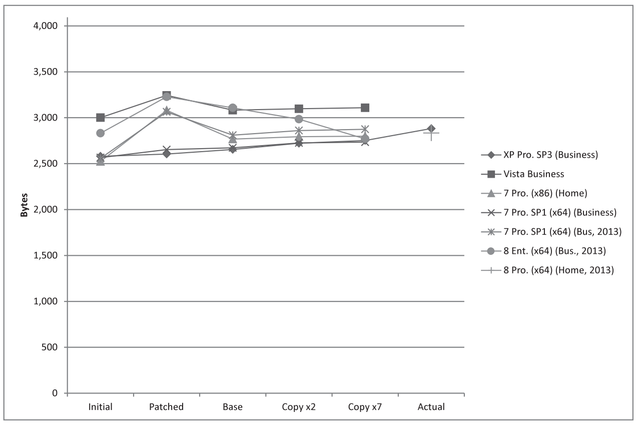

Figure 8: Mean Slack Space per File

Figure 8 shows that the mean slack space per file remains relatively constant over the life cycle of the Microsoft Windows operating system and applications, and that there is no significant difference in the mean value between versions of Windows investigated here. For all tested images the mean slack space per file on the Boot volume is well above the 2048 byte value that we would expect for purely random usage of 4096 byte clusters. This is potentially caused by Windows installations containing many files that are much smaller than half a cluster in size but may be an area worthy of further investigation.

Figure 9 shows that in the early part of the Windows life-cycle, slack space entropy — on initially zeroed storage media — is very low and then begins to gradually increase as the Windows installation ages and clusters are reallocated. The lowest slack space compression

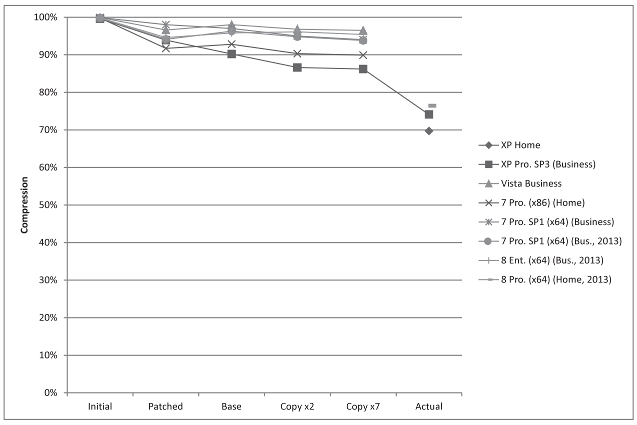

Figure 9: Slack Space Compressibility

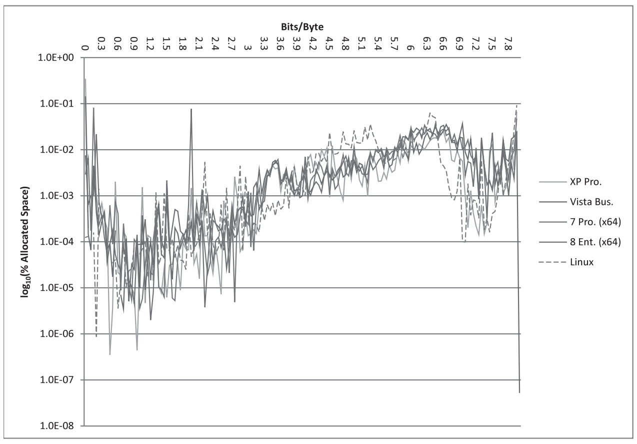

Figure 10: Allocation Unit Entropy (New Installation)

ratios were observed on the production Windows images and varied between 65% and 81%. In an attempt to identify an upper bound for slack space entropy, a boot volume was initialised to pseudo-random values (using a Unix "`urandom`" device) before Windows was installed; a slack space compression ratio of 57% was observed in this (approximate worst) case. We note that at the time of writing no actual volumes that had been in use for sufficiently long existed for Microsoft Windows 7, hence figure 9 only shows these data points for the case of Microsoft Windows XP volumes.

The "aging" of installation also is a potentially relevant element of information in that it not only affects the entropy of different elements of the volume such as overwritten but subsequently deleted or otherwise orphaned storage, but also serving as an indicator of an attempt to remove potential evidence by wiping a file system and subsequently replacing files; this may e.g. be the case if a system that had previously been infected with malware is replaced with a known good instance prior to the analysis taking place. Figure 10 shows the initial entropy per allocation units (normalised as bits per byte) plotted against the fraction of the volume occupied by allocation units (files) of this entropy.

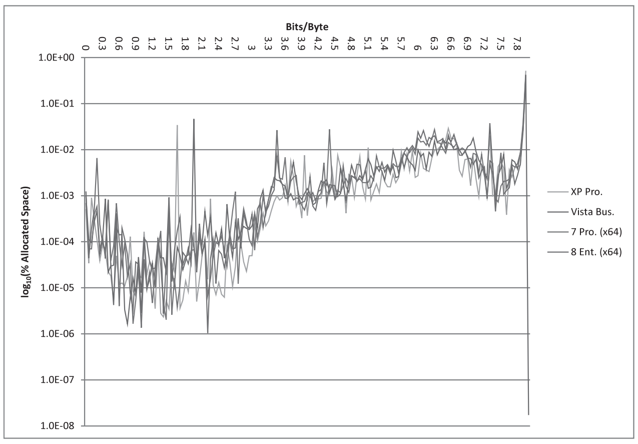

Figure 11: Allocation Unit Entropy (7 Generations of Use, Approx. 2 Years)

Figure 11 then shows the changes in entropy per allocation unit after seven aging iterations approximating 2 years of regular desktop use. Even without a detailed statistical analysis, the "aging" effect is clearly visible. However, whilst one would naïvely expect the entropy distribution to shift rightward, this is not necessarily the case.

We note that figure 10 also contains a plot for the initial distribution for a Linux installation; resource limitations did not allow analogous experiments to be conducted for Linux, so only results for different Windows variants are reported here.

6. ANALYSIS

A combination of entropy and randomness testing appears to be capable of detecting encrypted files from file content alone. Encrypted and highly compressed data prove to be effectively indistinguishable when only small amounts of each are analysed; a size boundary for reliable differentiation is not established in this paper.

Frequency analysis of the entropy of files on a Boot volume can give an indication about which type of operating system is installed. The system files located on a Windows Boot volume have relatively low entropy. High overall entropy in both allocated and unallocated space therefore indicates the presence of significant quantities of (high entropy) user data on the volume.

Allocated space compression ratios of around 30% appear to be typical for production Windows Boot volumes that have been used for some time; ratios significantly below this can be considered anomalous. Unallocated space on a Windows Boot volume does not appear to have a "typical" entropy value. An unallocated space entropy value significantly lower than the typical allocated space entropy value may merit further investigation.

Unallocated space with very low entropy indicates a recently created volume or one that has been deliberated wiped. When data is copied between volumes, it is likely that slack space will be transferred but unlikely that unallocated space will be transferred; this may lead to a mismatch between slack space entropy and unallocated space entropy from which we might infer a transfer or volume sanitisation.

Slack space compression of 65-75% is typical for a Windows installation that has been used for some time. Slack space compression ratios below 60% are unlikely to occur in normal Windows operation and would merit further investigation.

7. CONCLUSIONS

The results presented in this paper demonstrate that entropy and randomness measurements may be able to differentiate between encrypted and unencrypted data files with a reasonable degree of confidence, permitting automation. These same measurements may also help identify atypical Windows usage such as volume copies, volume wipes and unusually high entropy slack space. A strong result obtained in the analysis is the correlation of the compression ratio for slack space with the age of an installation, as any anomaly is likely to warrant further investigation. For the case of files we have found that — provided sufficiently large files are available for analysis — a combination of entropy and randomness tests will suffice to identify characteristics of encrypted data without having to rely on meta-data.

An adversary aware of these findings may attempt to cast doubt upon these measurements by highlighting that similar results can be obtained when analysing highly compressed files. When small amounts of data are involved then this can be an effective defence because entropy and randomness tests struggle to reliably differentiate encrypted and highly compressed data. For larger amounts of data, however, an adversary may be forced to use alternative defences such as filling unallocated storage with high entropy data and monitoring slack space entropy. Such countermeasures may themselves be identified as atypical usage which trigger further investigation.

Future work will seek to study the applicability of the results reported here to other types of (local) file systems and newer editions of the operating systems studied. We are particularly interested in analysing the characteristics of newer, log-based file systems as this has thus far not been studied to the best of our knowledge.

A further natural extension of the work described here is also the development of counter-forensic mechanisms that either avoid yielding tell-tale signatures identified, or to provide extensive decoys to increase the work-load and extent of manual investigation required as well as creating a plausible deniability scenario. Attention has been paid in the present work to facilitate automation of measurements as far as possible; it appears to be highly desirable to repeat measurements particularly at the file analysis level regularly as changes in file formats and e.g. encoders in case of multimedia files may change over time, skewing results.

The data and mechanisms used in generating the results described here are freely available from the authors subject to licensing conditions for the software used in the image files themselves except where privacy restrictions do not permit the release of personally identifiable information.

REFERENCES

[1] H.M. Government, "Regulation of Investigatory Powers Act 2000," Her Majesty's Stationery Office and Queen's Printer of Acts of Parliament, Jul. 2000.

[2] T. Holleboom and J. Garcia, "Fragment Retention Characteristics in Slack Space – Analysis and Measurements," in *Proceedings of the 2010 2nd International Workshop on Security and Communication Networks (IWSCN 2010)*. Karlstad, Sweden: IEEE Press, May 2010, pp. 1–6.

[3] Z. Blacher, "Cluster-Slack Retention Characteristics: A Study of the NTFS File System," Master's thesis, Department of Computer Science, Karlstad University, Karlstad, Sweden, Jun. 2010.

[4] R. Poisel and S. Tjoa, "Forensics Investigations of Multimedia Data: A Review of the State-of-the-Art," in *Proceedings of the Sixth International Conference on IT Security Incident Management and IT Forensics (IMF 2011)*, D. Guenther and H. Morgenstern, Eds. Stuttgart, Germany: IEEE Press, May 2011, pp. 48–61.

[5] I. Ahmed, K.-S. Lhee, H. Shin, and M. Hong, "On Improving the Accuracy and Performance of Content-Based File Type Identification," in *Proceedings of the 14th Australasian Conference on Information Security and Privacy (ACISP 2009)*, ser. Lecture Notes in Computer Science, C. Boyd and J. González Nieto, Eds., vol. 5594. Brisbane, Australia: Springer-Verlag, Jul. 2009, pp. 44–59.

[6] M. M. Shannon, "Forensic Relative Strength Scoring: ASCII and Entropy Scoring," *International Journal of Digital Evidence*, vol. 2, no. 4, pp. 1–19, Apr. 2004.

[7] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon Entropy Measure with Statistical Tests for Image Randomness," *Journal of Information Sciences*, 2012, (article in press).

[8] C. J. Veenman, "File Fragment Classification — The Case for Specialized Approaches," in *Proceedings of the Third International Symposium on Information Assurance and Security (IAS 2007)*. Manchester, UK: IEEE Press, Aug. 2007, pp. 393–398.

[9] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Security & Privacy*, vol. 5, no. 2, pp. 40–45, Mar./Apr. 2007.

[10] X. Ugarte-Pedrero, I. Santos, B. Sanz, C. Laorden, and P. Garcia Bringas, "Countering Entropy Measure Attacks on Packed Software Detection," in *Proceedings of the 9th Annual IEEE Consumer Communications and Networking Conference — Security and Content Protection*. Las Vegas, NV, USA: IEEE Press, Jan. 2012, pp. 164–168.

[11] W. C. Calhoun and D. Coles, "Predicting the Types of File Fragments," *Digital Investigation*, vol. 5, no. (supplement), pp. S14–S20, Sep. 2008.

[12] V. Roussev and S. L. Garfinkel, "File Fragment Classification — The Case for Specialized Approaches," in *Proceedings of the 4th International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE 2009)*. Berkeley, CA, USA: IEEE Press, May 2009, pp. 3–14.

[13] VMWare, Inc., *Getting Started with VMware Fusion*, http://www.vmware.com/, Palo Alto, CA, USA, 2011, Last accessed 1 Nov. 2012.

[14] B. Carrier, *File System Forensic Analysis*, 1st ed. Reading, MA, USA: Addison-Wesley, 2005.

[15] J. Walker, "ENT – A Pseudorandom Number Sequence Test Program," http://www.fourmilab.ch/random/, Jan. 2008, Last accessed 1 Nov. 2012.

[16] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[17] L. P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," IETF Network Working Group Request for Comments RFC 1951, May 1996.

[18] Micrsoft Corporation, "How to Clear the Windows Paging File at Shutdown," http://support.microsoft.com/kb/314834, Oct. 2010, Last accessed 1 Nov. 2012.

[19] ——, "Windows 8 / Windows Server 2012: The New Swap File," http://blogs.technet.com/b/askperf/archive/2012/10/28/windows-8-windows-server-2012-the-new-swap-file.aspx, Oct. 2012, Last accessed 28 Oct. 2012.

[20] ——, "How NTFS reserves space for its Master File Table (MFT)," http://support.microsoft.com/kb/174619, Oct. 2008, Last accessed 1 Nov. 2012.

[21] E. Huebner, D. Bem, and C. Kai Wee, "Data Hiding in the NTFS File System," *Digital Investigation*, vol. 3, no. 4, pp. 211–226, Dec. 2006.

[22] D. M. Purcell and S.-D. Lang, "Forensic Artifacts of Microsoft Windows Vista System," in *Intelligence and Security Informatics: Proceedings of the IEEE ISI 2008 International Workshops: PAISI, PACCF and SOCO 2008*, ser. Lecture Notes in Computer Science, C. C. Yang, Ed., vol. 5075. Taipei, Taiwan: Springer-Verlag, Jun. 2008, pp. 304–319.