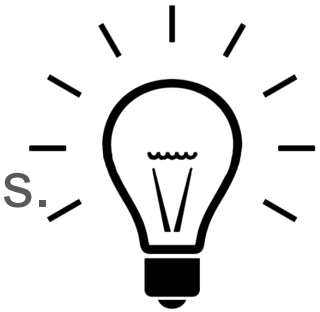
Using Entropy Analysis to Find Encrypted and Packed Malware

Paper written by Robert Lyda & James Hamrock
Presented by Kevin Chu

Keywords



- Entropy - level of difficulty or the probability of independently predicting each number in a series.
- Bintropy - a binary entropy analysis tool that the researchers used
- PE Executable - standard binary file format for an Executable or DLL
- Confidence interval - % chance that results holds true on repeated sampling

What is Entropy Analysis?

- Entropy analysis examines the statistical variation in malware executables.
- This would allow analysts to quickly and efficiently identify packed and encrypted samples.

Why do we care?

- Malware authors use encryption or packing (compression) to conceal their malicious executables' string data and code.
- These methods, which transform some or all of the original bytes into a series of random-looking data bytes, appear in 80 to 90 percent of malware samples.
- This is a convenient and quick technique for analyzing a sample at the binary level and identifying suspicious PE file regions

Approach

- They used the Bintropy tool on four separate tests with training data sets for native, compressed, encrypted, and plaintext files.
- Native - 100 Windows 32-bit PE executables
- Packed - executables generated by using UPX and Morphine 1.2
- Encrypted - executables encrypted by Pretty Good Privacy

Approach (con’t)

- The experiments showed that executables generally contain many blocks of mostly zero-value data bytes, which compilers commonly generate to pad or align code sections. This technique can greatly reduce an executable’s entropy score, because it increases the frequency of a single value.
- To compensate for this, Bintropy was altered to analyze only “valid” byte blocks (blocks in which at least half of the bytes are nonzero).

Results

Table 1. Computed statistical measures based on four training sets.

DATA SETS	AVERAGE ENTROPY	99.99% CONFIDENCE INTERVALS (LOW TO HIGH)	HIGHEST ENTROPY (AVERAGE)	99.99% CONFIDENCE INTERVALS (LOW TO HIGH)
Plain text	4.347	4.066 – 4.629	4.715	4.401 – 5.030
Native executables	5.099	4.941 – 5.258	6.227	6.084 – 6.369
Packed executables	6.801	6.677 – 6.926	7.233	7.199 – 7.267
Encrypted executables	7.175	7.174 – 7.177	7.303	7.295 – 7.312

Results (con’t)

- Using a 99.99 percent confidence level, executables with an average entropy and a highest entropy block value of greater than 6.677 and 7.199, respectively, are statistically likely to be packed or encrypted.
- These two values are the lower confidence interval bounds of the entropy measures we computed for packed executables, and form the basis for the methodology for analyzing malware executables for the presence of packing or encryption.
- If both the Bintropy computed average file entropy and highest entropy block score exceed these respective values, it labels a malware executable as packed or encrypted.

Results

- The eight standard sections: .text, .data, .rsrc, .reloc, .rdata, .idata, CODE, and DATA are created by default by most PE-generating compilers.
- The remaining five sections: .aspack, UPX1, UPX2, pec1, and pec2 are created by packing technologies that replace the default PE-formatted sections and their bytes with custom ones

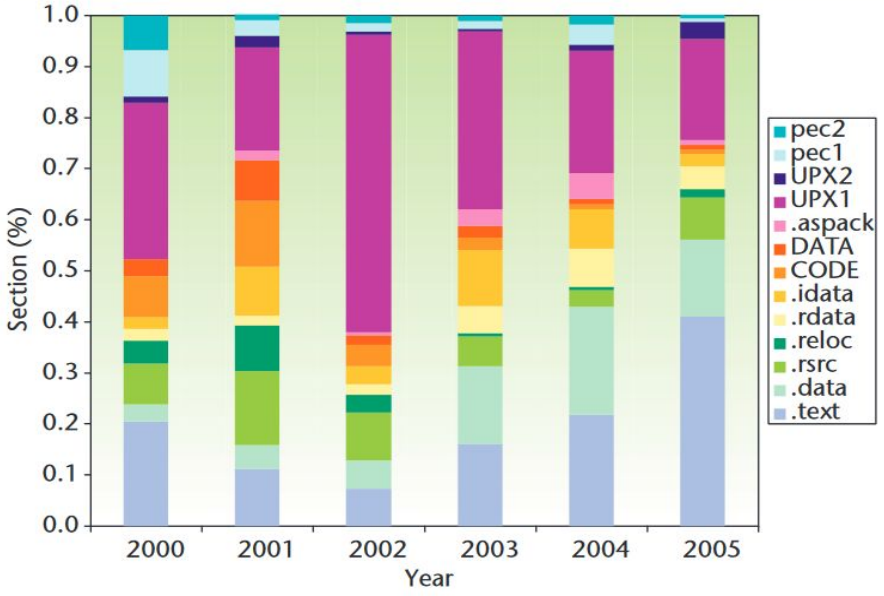


Figure 1. Percentage of encrypted or packed sections over a six-year period. UPX1 was the most prevalent of the packed sections across the period, followed by the .text section.

Results

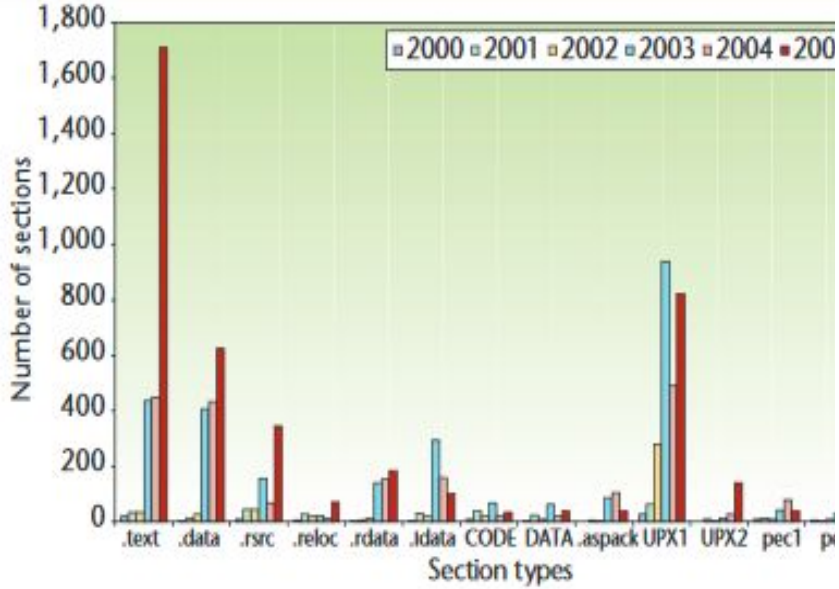


Figure 2. Number of encrypted sections by year. Packing of .text section increased the most dramatically across the period.

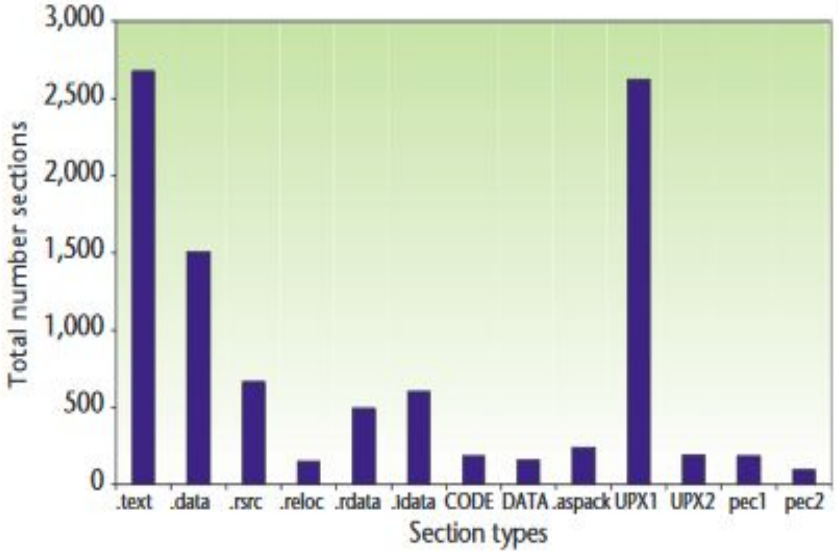


Figure 3. Total number of encrypted sections over the six-year study period. Packing of the .text and .UPX1 were the most prevalent during this time period.

Evaluation

- In 2000, .reloc was the most often packed or encrypted section. This section's popularity steadily declined across the remaining years, with only a slight increase in 2005.
- The second most-packed section in 2000 was UPX1, which is generated by the very popular UPX packing tool. Due to UPX's prevalent use in numerous W32/GAOBOT and W32/SPYBOT variants, the presence of the UPX1 section in the data set increased over the next few years, peaking in 2002.
- Thereafter, its prevalence steadily decreased, but UPX1 remained the most popular of all the sections we identified as packed across this six-year period, followed by the .text section, which is where the compiler writes most of a program's executable code.

Limitations

- It’s infeasible to absolutely determine if a sample contains compressed or encrypted bytes.
- False negatives can occur when large executables (those larger than 500 kbs) contain relatively few encrypted or compressed blocks and numerous valid blocks, thereby lowering the executable file’s average entropy measure.
- A standard t-test was applied to the data set, and it calculated a Type I error’s significance level (the false positive rate) to be 0.038 percent.
- Sophisticated malware authors could employ countermeasures to conceal their encryption or compression use.

Future considerations

- Analysts can perform further detailed analysis with other reverse-engineering tools, such as the IDAPro disassembler.
- A more fine-grained approach might be useful in identifying the particular transformation algorithms that malware authors apply to their malware.

Discussion & Questions

- Was there anything unclear about it?
- What did you like about it? Dislikes?
- What did you learn?
- Other applications?