(U) Experience with Compression-Based Distance Metrics for Malware

Charles Nicholas[1,2] Kevin Stout[2]

[1]Department of Computer Science and Electrical Engineering, UMBC
[2]Department of Defense

August 2012

kmstout ©

The overall classification of this presentation is:
UNCLASSIFIED//FOR OFFICIAL USE ONLY

Normalized Compression Distance

- How can we tell if we have seen some piece of malware before?
- Normalized Compression Distance was introduced by Li *et al* in 2004 [1]
- If $c(x)$ is the length of object a when compressed, then

$$NCD(x,y) = \frac{c(xy) - min(c(x), c(y))}{max(c(x), c(y))}$$

- Intuition: similar objects will share substrings, and thereby "help each other" during compression

Properties of NCD

- A *distance metric* d satisfies three properties: for any three objects x, y, z
  - Reflexivity: $d(x, x) = 0$
  - Symmetry: $d(x, y) = d(y, x)$
  - Triangularity: $d(x, y) + d(y, z) >= d(x, z)$
- NCD satisfies these in theory, but not in practice, due to overhead imposed by compression algorithms. (We used the *xz* option in R's memCompress function [2].)
- Example: DLL files from a Windows/System32 directory.

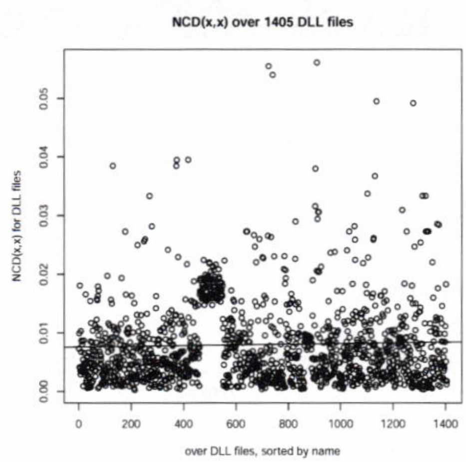

Figure: (U) DLL files are represented in alphabetical order on the X axis. Note the least-squares fit line, and the clusters.

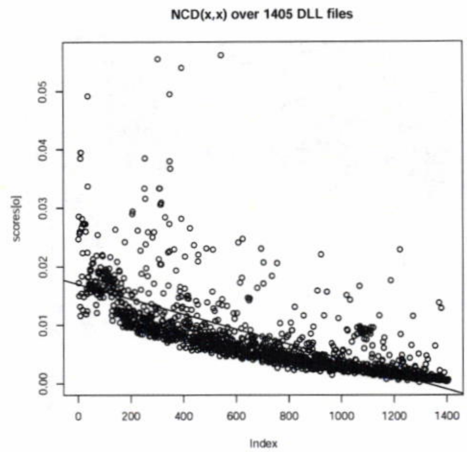

Figure: (U) NCD as a function of file length. The longer the file x, the closer $NCD(x, x)$ is to zero.

More About NCD

- For most x, $NCD(x, x) = 0$ is almost but not exactly true.
- For most x, y, $NCD(x, y) = NCD(y, x)$ is almost but not exactly true.
- The triangle inequality holds, in part *because* of the compression overhead.
- NCD is useful for comparing binaries, but computing NCD requires us to create some (possibly big) objects only to measure their length when compressed, and compression is relatively slow: $O(n\ log\ n)$.

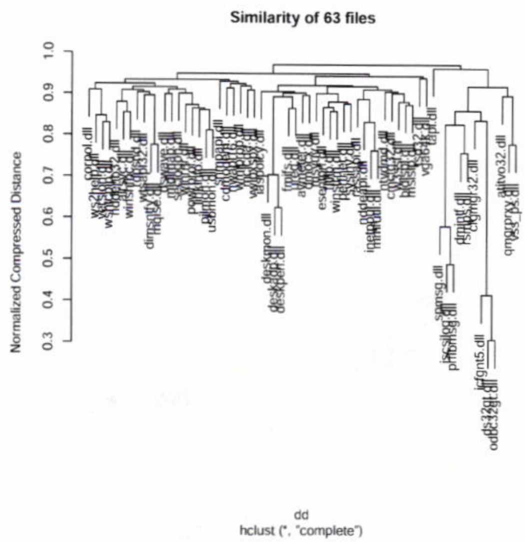

Figure: (U) We can use NCD to compare binaries, and performance is reasonable for small sets.

xz compression, over-simplified

- R's xz compression function implements Lempel-Ziv compression by finding strings in an object that occur more than once, and replacing them with shorter strings [3]
- The dictionary of strings and their shorter "stand-ins" is attached to the compressed file, which imposes some necessary overhead
- Such a compression dictionary can be created *without* doing any compression, in $O(n \log n)$ time.

The *dzd* similarity metric

- Substrings that occur in both files will also appear in both compression dictionaries
- Let $d(x)$ be the set of dictionary entries generated when x is compressed, and measure the overlap between the compression dictionaries, as Jaccard might suggest:

$$dzd(x, y) = 1 - \frac{|d(x) \cap d(y)|}{|d(x) \cup d(y)|}$$

- The range of *dzd* is [0,1]
- Reflexive, Symmetry and Triangularity properties follow from elementary set theory

dzd is easy to implement

- A given object's compression dictionary can be built once, sorted, saved, and used in subsequent calculations. (About 30 lines of Perl.)
- Since R has suitable built-in set operations, and having stored the compression dictionaries, we can compute dzd in $O(n)$ time, vs. $O(n \log n)$ for NCD.
- No need to build a global set of terms, as would be necessary with (for example) the vector space model.

The *dzdW* similarity metric

- The compression dictionaries also have string frequencies, that is, how many times was a given string "emitted"?
- Intuition: if objects x and y share many strings that occur a lot, that tells us more than if they share strings that occur only rarely.
- Compute normalized frequencies of strings in a document, and add up the products of matching string frequencies

$$dzdW(x, y) = \sum_j f_{x,j} \times f_{y,j}$$

where $f_{x,j}$ is the normalized frequency of term j in document x
- Again, the distance metric properties hold

Using dzd and dzdW in a Malware Collection

- We have a private collection of many thousands of malware objects, of various kinds
- Executable binaries are of particular interest, so we built compression dictionaries for those
- We then compared $NCD(x, y)$ with $dzd(x, y)$ for 1,000 random pairs of executable binaries
- NCD took 505 seconds to do those comparisons, versus 195 seconds for *dzd*

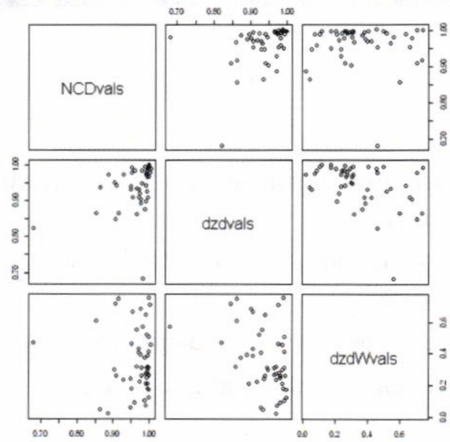

Figure: (U) Comparing $NCD(x, y)$, $dzd(x, y)$ and $dzdW(x, y)$ for 1,000 random pairs of "malware" files.

When Malware Files are Similar

- NCD, *dzd*, and *dzdW* have different distributions, hence different critical values. For example, the "1%" critical value of *dzd* is 0.57, versus 0.85 for NCD
- We noticed a pair of files x, y with $dzd(x, y) = 0.60$ which happens by chance less than 5% of the time. These two executables had little in common except for a particular form of obfuscation.

Conclusions and Future Work

- We have proposed and implemented versatile distance metrics for files called *dzd* and *dzdW*
- *dzd* and *dzdW* seem consistent with NCD, but seem faster (after one-time pre-processing)
- Our effort to use these metrics to cluster malware continues.
- POC: Charles Nicholas `nicholas@umbc.edu`

References

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul M. B. Vitanyi.
The similarity metric.
IEEE Transactions on Information Theory, 50(12):3250–3264, December 2004.

R Development Core Team.
R: A Language and Environment for Statistical Computing.
R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

Terry Welch.
A technique for high-performance data compression.
IEEE Computer, 17(6):8–19, June 1984.