Electronic Journal of SADIO

http://www.dc.uba.ar/sadio/ejs/

vol. 7, no. 1, pp. 31-42 (2007)

Preliminary Results on Masquerader Detection using Compression Based Similarity Metrics

Maximiliano Bertacchini[1] *Pablo I. Fierens*[2]

[1] CITEFA - Instituto de Investigaciones Científicas y Técnicas de las Fuerzas Armadas
mbertacchini@citefa.gov.ar
[2] ITBA - Instituto Tecnológico de Buenos Aires
pfierens@itba.edu.ar

Abstract

This paper extends a series of experiments performed by Schonlau et al. [1] on the detection of computer masqueraders (i.e. illegitimate users trying to impersonate legitimate ones). A compression-based classification algorithm called Normalized Compression Distance or *NCD*, developed by Vitányi et al. [2] is applied on the same data set. It is shown that the *NCD*-based approach performs as well as the methods previously tried by Schonlau et al. Future work, possible enhancements and directions of further research on this topic are presented as well.

Keywords. Kolmogorov complexity, command behavior, masquerade detection, normalized compression distance, similarity metrics.

1 Introduction

This work is based on previous work by Matthias Schonlau on command-line computer masquerader detection, as well as on work by Paul M.B. Vitányi et al. on the development of a novel clustering and classification method called "Normalized Compression Distance" (NCD).

In computer intrusion detection, masqueraders are those who try to hide their identities by impersonating other people using somebody else's computer account [1, 3].

Our goal is to apply the NCD method to detect masquerades in command line data, in an effort to measure the accuracy of this approach in comparison with the methods presented by Matthias Schonlau et al. For this purpose, the same command line data set as well as the same testing and measuring methodologies utilized by Schonlau et al. are used.

This work is part of a more extendend research project of CITEFA SI6 Information Security Labs[3]. The main goal of this project is the development of a computer intruder identification system based on the network behavior of intruders. This project proposes the identification and detection of remote computer intruders by utilizing behavioral biometric techniques. This will be done by means of the analysis of a number of host parameters (keystroke patterns, used commands, language, errors, etc.) as well as network parameters (IP addresses, traversed routers, operating systems, etc.).

This paper is structured as follows: Section 2 introduces previous work which this paper is based on, mainly intruder detection and the Normalized Compression Distance clustering and classification method; Section 3 gives a brief explanation on the theoretical basis of this work, including an introduction to Kolmogorov complexity and compression in general; Section 4 shows the experiments performed and the results obtained; finally, Section 5 presents our conclusions and puts forward future related work.

2 Previous Work

2.1 Intruder Detection

Schonlau et al. [1] analyzed and compared the performance of six different statistical methods to detect masqueraders through anomaly detection on the command line history of each user. The six approaches are: "Uniqueness", "Bayes one-step Markov", "Hybrid multistep Markov", "Compression", "IPAM" and "Sequence-Match".

Of these methods, the most similar to the proposed one is Compression. This approach bases on the premise that data from the same user compresses more readily than mixed data from different users. Intuitively this may be explained by the fact that it is easier to detect patterns in data from only one user, which should permit to achieve a greater level of compression, than in

[3]CITEFA Si6 Labs homepage: `http://www.citefa.gov.ar/SitioSI6_EN/si6.htm`.

data from different users. In [1], results from the Compression method seem discouraging in comparison with the other five methods.

2.2 The Normalized Compression Distance

A distance function is said to be a metric if it verifies the following properties: existence of identity, symmetry and triangular inequality. A distance is admissible if it is computable, symmetric and dense. A normalized admissible distance or similarity distance $d(x, y)$ is an admissible distance in the range $[0, 1]$, such that $d(x, y) \to 0$ when x and y are maximally similar, and $d(x, y) \to 1$ when x and y are maximally dissimilar.

In [4] the Normalized Information Distance ($NID(x, y)$) is introduced, which is based on previous work on the Information Distance $E(x, y)$ by the same authors [5]. The Normalized Information Distance is defined as:

$$NID(x, y) = \frac{max(K(x|y), K(y|x))}{max(K(x), K(y))} \leqslant 1$$

where $K(x|y)$ is the conditional Kolmogorov complexity of x given y, that is, the length of the shortest program which produces x with y as input (see Section 3 for a brief introduction on this subject). The NID is a theoretical universal normalized admissible distance satisfying the metric (in)equalities, i.e. a similarity metric, which relies on the **non-computable** notion of Kolmogorov complexity [6].

In [2] a novel method for clustering and classification based on compression is presented. This method is called *Normalized Compression Distance* ($NCD(x, y)$), and represents an implementation of the Normalized Information Distance ($NID(x, y)$) using real-world compressors such as `gzip` or `bzip` in order to approximate Kolmogorov complexity, i.e., $K(x)$ can be approximated "from above" using a real-world compressor C, such that:

$$|K(x)| < |C(x)| < |x|$$

The Normalized Compression Distance is defined as:

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

where C is a *normal lossless compressor*, that is, a compressor that verifies the following conditions (up to an additive term $O(\log n)$): idempotency, monotonicity, symmetry and distributivity.

It is shown in [2] that real-world compressors verify these properties up to some term depending on the compressor's ability to capture patterns in the objects concerned. Moreover, it is also shown that the Normalized Compression Distance is a similarity metric under the assumption of real-world normal compressors.

Clustering based on the NCD is parameter-free in the sense that it does not use any features or background knowledge about the data; it is universal,

since it reveals any underlying similarity between two objects; and it is robust as its results appear independent from the real-world compressor used. The fundamental intuitive idea is that two objects are similar if one of them can be compressed given the information in the other. In [2], evidence of successful application of the NCD is presented, showing the results in areas as diverse as genomics, virology, languages, literature, music, handwritten digits, astronomy, and combinations of objects from completely different domains, using statistical, dictionary, and block sorting compressors.

The "accuracy" of the $NCD(x, y)$ depends heavily on the underlying compressor's ability to detect patterns in x and y.

2.3 Comparison of the NCD and the "Compression" Approaches

The "Compression" score x is defined by Schonlau et al. in [1] as the number of additional bytes needed to compress test data when appended to training data (for further explanation on the structure of command line data, see Subsection 4.1):

$$x = compress(\{D_{Tr}, D_{Te}\}) - compress(D_{Tr})$$

where D_{Tr} is the training data, D_{Te} the test data, $\{D_{Tr}, D_{Te}\}$ is the test data appended to the training data and $compress()$ is a function that gives the number of bytes of the compressed data.

It can be observed that the main difference between the "Compression" approach and the NCD-based approach is the lack of normalization in the former formula. This implies that result values in this case depend on the sizes of input data and thus cannot be taken as a universal absolute measure of alikeness. In the case of NCD, results are in the range $[0, 1]$ and can be used as a universal similarity measure.

3 Backgroud Theory

3.1 Kolmogorov Complexity

Intuitively, the amount of information in a finite string is the size (number of binary digits, or bits) of the shortest program that without additional data, computes the string and terminates [6].

It can be shown that all reasonable choices of programming languages lead to quantification of the amount of "absolute" information in individual objects that is invariant up to an additive constant. This quantity is called the "Kolmogorov complexity" of the object. If an object contains regularities, then it has a shorter description than itself. Such an object is called "compressible" [6].

The Kolmogorov complexity of a string is the length of its shortest description in some fixed language (eg. the code of a Turing Machine):

$$K(x) = min|P| : U(P) = x$$

where P is a program that produces x with no inputs and finishes, and U is the Universal Turing Machine. $K(x)$ represents the minimum information needed to produce x, as well as the maximum compressibility of x. It can be shown that the Kolmogorov complexity of a binary string is upper-bounded by the length of the same string up to an additive constant. There is a constant c such that:

$$K(x) \leqslant |x| + c, \forall x$$

Therefore, the Kolmogorov complexity is an absolute measure of the theoretical maximum compressibility of an object.

Unfortunately, $K()$ is not a computable function.

For a more thorough overview on this subject and its applications, see [6].

3.2 Real-world Compressors

If a string x is uncompressible, then it has no "patterns". Therefore, the compressibility of an object measures the information contained by that object; and compression can be seen as the process of redundancy elimination.

"Real" generic compresors assume input data has no global meaning. They interpret data as a bit string, and search for patterns and statistical deviations. Therefore they do not take into account the "meaning" of data, which should further improve the compression level.

Example. *The trascendental number π has a low Kolmogorov complexity, as it can be obtained by a simple algorithm up to infinite precision given infinite resources; but it seems completely random and uncompressible to any real-world compressor.*

Two of the most popular real-world compressors are `gzip` and `bzip2`.

`gzip` (`GNU zip`) is a compression utility designed to be a replacement for `compress` [7]. Its main advantages over `compress` are much better compression and freedom from patented algorithms. It has been adopted by the `GNU` project and is now relatively popular on the Internet [8]. `gzip` is based on the DEFLATE algorithm [9], which is a combination of LZ77 and Huffman coding. DEFLATE was intended as a replacement for LZW. The compression library in which `gzip` is built is called `zlib` [10].

`bzip2` is a freely available, open-source, patent free, high-quality data compressor. It typically compresses files to within 10% to 15% of the best available techniques (the PPM family of statistical compressors), whilst being around twice as fast at compression and six times faster at decompression. `bzip2` uses the Burrows-Wheeler transform [11] to convert frequently recurring character sequences into strings of identical letters, and then applies a move-to-front transform [12] and finally Huffman coding [13].

4 Experiments

This work tries to reproduce as exactly as possible the design of the experiments showed in [1]. Specifically, the same data set was used, and the same

considerations and assumptions were made.

4.1 Command-line Data

Command data was downloaded from Matthias Schonlau's Internet homepage[4]. This data set consists of commands collected from UNIX `acct` audit data. Of all fields of audit data provided by `acct` only the username and the command were taken. The command history of 70 users was used. Out of these 70 users, 50 were chosen randomly to represent the user base for testing, and the remaining 20 were used as masqueraders. Data was split into 50 files corresponding to 50 different users. Each file contains 15000 commands, one per line. The first 5000 commands of each file are considered genuine. The remaining 10000 commands of each user are divided into 100 blocks of 100 commands each. These blocks are seeded with masquerading users, i.e. with data of another user not among the 50 users. A block is a masquerader with a probability of 1%. If a block is a masquerader, then the following one is a masquerader too with a probability of 80%. As a result, approximately 5% of the test data contain masquerades. Not all users testing data have masquerades interspersed.

It should be noted that commands have no arguments nor any parameters, as this additional information was not provided by `acct`. It should also be mentioned that due to the way `acct` collects audit data from the system, it is impossible to tell commands typed by human users on the command line prompt from those run automatically from shell scripts. This can lead to some users having a very regular pattern of few commands, due to the repetitive automated execution of shell scripts, such as `.profile` and `make` files.

4.2 The Complearn Toolkit

CompLearn is a suite of simple-to-use utilities that can be used to apply compression techniques to the process of discovering and learning patterns[5]. It is a software implementation of the normalized compression distance (NCD), presented in [2] and developed by some of its authors. It is also the base of most experiments shown in [2]. Therefore, the CompLearn toolkit implements a feature-free, universal clustering method.

The Complearn implements the following built-in compressors:

- `bzip`
- `google` [14]
- `zlib` [10]

It features the possibility to add any custom compressor as well.

The processing of data with CompLearn is modularized into several commands. The main command used for testing in this work is `ncd`. The `ncd`

[4]Matthias Schonlau's homepage: `http://www.schonlau.net`.
[5]The CompLearn Toolkit homepage: `http://www.complearn.org`.

command reads input data, which can be specified in several different formats, consisting of say n objects, and returns a $n \times n$ symmetric square matrix with the pairwise distance between every two objects (the `distmatrix.clb` file). This process is mainly CPU-bound but its complexity is polynomial with respect to the number of input objects.

4.3 Experimental Design

The experiment is constructed as follows. The 5000 commands of training data represent a user profile. For each block of 100 commands of testing data, a score is computed by comparing it with the user profile, as described in [1]. If the score exceeds a threshold, an alarm indicating a potential masquerade is triggered. No updating of the user profiles is performed.

Our approach is based on the "Compression" method described in [1]. The only difference is the use of the NCD for the score calculation instead of the original unnormalized `compress`-based approach. The threshold for each user is obtained in the same fashion as in the "Compression" method.

Different compressors were used in turn as the underlying compressor for the NCD with the purpose of comparing each one's ability to detect similarities and patterns in the command line data.

Tests were run using the following underlying compressors for the NCD:

- bzip
- compress
- zlib
- raw zlib with a custom dictionary

`bzip` and `zlib` support is built-in into CompLearn. `compress` version 4.2.4 was used. The latter compressor generates raw zlib streams, i.e. without any header and checksum value, using a fixed custom dictionary, consisting only of those commands found in the training data, ordered by the number of times it was executed. In all cases the maximum compression level for each compressor was selected. `libcomplearn` version 0.9.2 was used.

4.4 Results

Tables 1 and 2 show the False Alarm and Missing Alarm Rates obtained when targeting a fixed False Alarm Rate corresponding to the `compress`-based and the `bzip`-based NCD methods, which are the best performing ones.

Results from the NCD-based approach are compared with those obtained by Schonlau et al. in Table 3. In all cases a false alarm rate of 1% was targeted. It can be observed that the NCD-based method's missing alarm rates are relatively high in comparison with the other methods. Particularly, the NCD based on `compress` performs better than all the other methods except for Bayes one-step Markov and Hybrid multistep Markov. On the other hand, the false alarm

Table 1: *NCD* (`compress`): False Alarm Rates and Missing Alarm Rates

Targeted False Alarms (%)	False Alarms (%)	Missing Alarms (%)
1	2.9	57.5
2	4.46	51.9
6	9.8	40.2
10	12.4	34.1
50	53.1	9.9

Table 2: *NCD* (`bzip`): False Alarm Rates and Missing Alarm Rates

Targeted False Alarms (%)	False Alarms (%)	Missing Alarms (%)
1	0.6	69.6
2	0.8	65.8
6	3.0	61.4
10	4.7	57.1
50	42.4	23.8

Table 3: Comparison of all methods (targeting a false alarm rate of 1%)

Method	False Alarms (%)	Missing Alarms (%)
Uniqueness	1.4	60.6
Bayes one-step Markov	6.7	30.7
Hybrid multistep Markov	3.2	50.7
Compression	5.0	65.8
Sequence-Match	3.7	63.2
IPAM	2.7	58.9
NCD (*bzip*)	**0.6**	**69.6**
NCD (*compress*)	**2.9**	**57.5**
NCD (*zlib*)	**1.2**	**77.9**
NCD (*raw zlib with custom dictionary*)	**1.3**	**82.6**

rates are low, especially for `bzip` and `zlib`, which are the lowest ones among all shown methods. The `zlib` dictionary-based approach performs poorly. It can also be observed that the NCD based methods are closer to the target false alarm rate.

Receiver operating characteristic curves, or ROC curves, are often utilized to show the trade-off between the missing alarm rate and the false alarm rate, by varying the masquerade detection threshold. The ROC curves for the NCD-based method using `zlib`, `bzip`, `compress` and raw zlib with a custom dictionary are shown in Figure 1.

It can be observed that the `compress` compressor performs better than `bzip`. On the other hand, the `bzip` compressor yields slightly better results than `zlib`. The raw zlib compressor with a custom dictionary fares poorly, though similarly to `zlib`.

The original ROC curves obtained from Schonlau et al's experiments are shown in Figure 2. Figure 3 shows a comparison between some of these curves and those of the two best performing NCD-based approaches. The NCD ROC curves perform better for low False Alarm Rates. Particularly, the `compress`-based NCD ROC curve is very similar to that of Compression, as expected.

5 Conclusions and future work

We have shown that the Normalized Compression Distance can be applied on the detection of computer masqueraders. Its accuracy is comparable to that of traditional methods, having no prior knowledge about the command line data. This is due to the universality and parameter independence of the NCD.

We should bear in mind that the command data used has no arguments or parameters. It should also be noticed that presently the analysis of data is purely syntactical, i.e. the semantical "meaning" of commands and their context of execution are not taken into account (eg. "`rm -fr /`" may be different than "`rm -fr /tmp/*`"). Commands are not divided into sessions. Should session data be available, it could be utilized in our calculations. The use of all these additional data should be explored. We believe that the accuracy of the presented method could be enhanced by taking into account these topics.

Intruder command line data, such as that captured by keyloggers, may be much shorter than 100 commands. The use of block lengths different from 100 commands should be analized more thoroughly, as it may have a significant impact on the accuracy of this method.

Presently the user profiles are built statically. The dynamic update of user profiles should also be explored further.

Finally, a thorough analysis of the impact of the underlying compressor used is due, as it could have a significant impact on the NCD ability to discern legitimate users from masqueraders. This could lead to the construction of a compressor specialized in command line data. A first approach on this direction was the construction of the raw `zlib` compressor with a custom dictionary, though its results were disappointing.

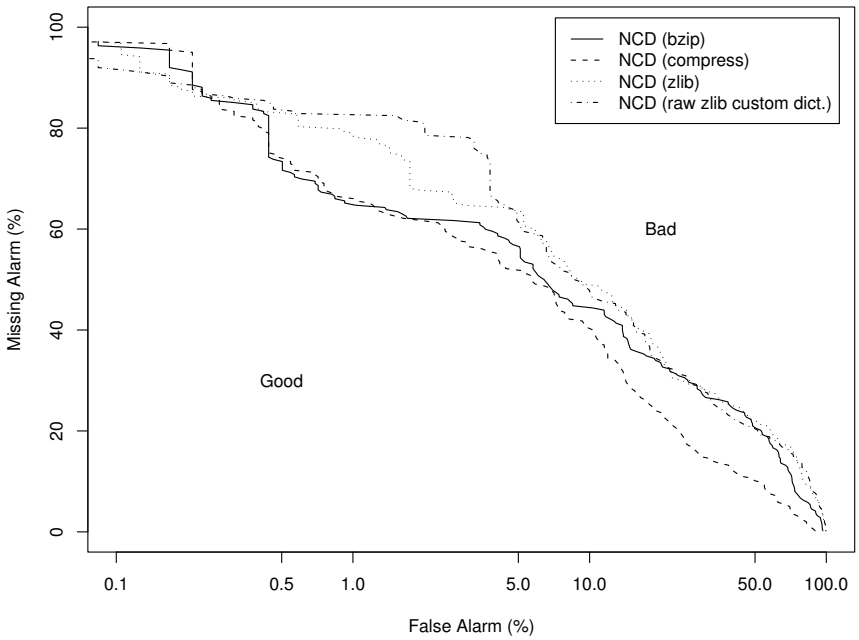

Figure 1: ROC curves for the *NCD* method using `zlib`, `bzip`, `compress` and raw `zlib` with a custom dictionary as the underlying compressor (no profile updating)

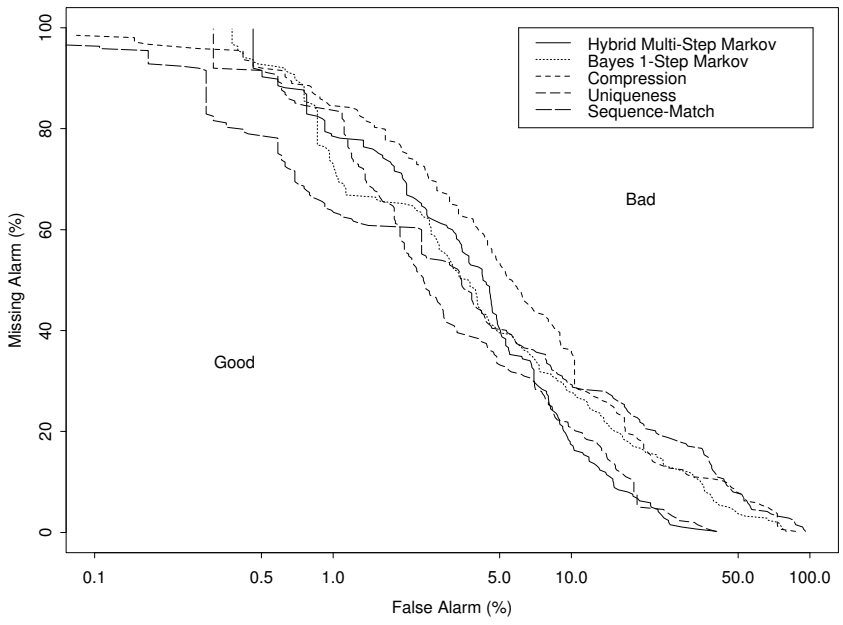

Figure 2: Original ROC curves obtained by Schonlau et al. (no profile updating). *(This figure is freely downloadable from Matthias Schonlau's homepage)*

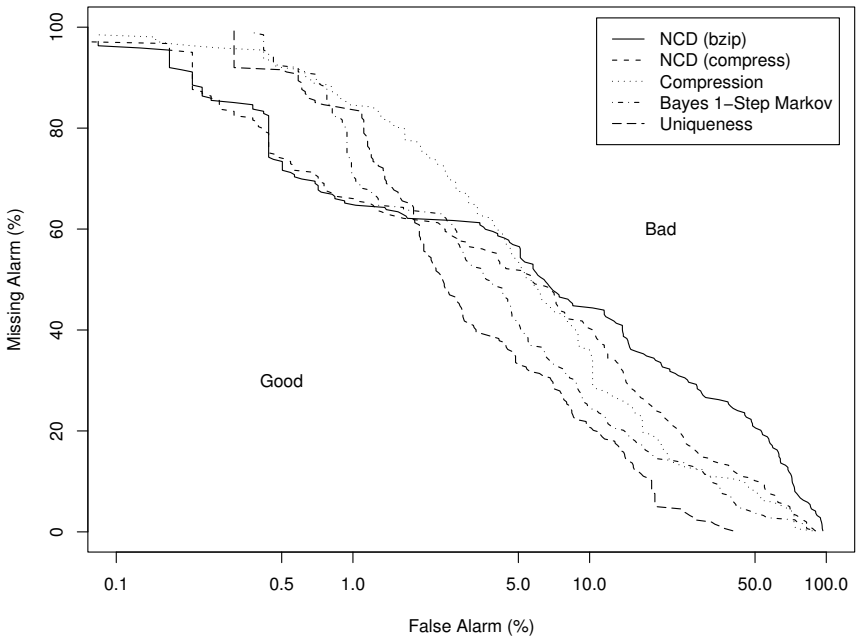

Figure 3: Comparison between NCD-based and some of the original ROC curves

6 Acknowledgements

We would like to thank all members of CITEFA's Si6 Lab for their support; and particularly Carlos Benítez, Sebastián García and the peer reviewers for their suggestions and corrections.

References

[1] Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y.: Computer Intrusion: Detecting Masquerades (2001) Statistical Science (submitted).

[2] Cilibrasi, R., Vitányi, P.: Clustering By Compression. IEEE Transactions on Information Theory **51**(4) (2005) 1523– 1545

[3] Schonlau, M., Theus, M.: Detecting Masquerades in Intrusion Detection Based on Unpopular Commands. Inf. Process. Lett. **76**(1-2) (2000) 33–38

[4] Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.: The Similarity Metric. IEEE Transactions on Information Theory **50**(12) (2004) 3250 – 3264

[5] Bennett, C., Gacs, P., Li, M., Zurek, W., Vitányi, P.: Information Distance. IEEE Transactions on Information Theory **44**(4) (1998) 1407–1423

[6] Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Second edn. (1997)

[7] Welch, T.A.: A technique for high-performance data compression. IEEE Computer **17**(6) (1984) 8–19

[8] Deutsch, L.P.: RFC 1952: GZIP file format specification version 4.3 (1996) Status: INFORMATIONAL.

[9] Deutsch, L.P.: RFC 1951: DEFLATE compressed data format specification version 1.3 (1996) Status: INFORMATIONAL.

[10] Deutsch, L.P., Gailly, J.L.: RFC 1950: ZLIB compressed data format specification version 3.3 (1996) Status: INFORMATIONAL.

[11] Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Technical Report 124 (1994)

[12] Bentley, J.L., Sleator, D.D., Tarjan, R.E., Wei, V.K.: A locally adaptive data compression scheme. Commun. ACM **29**(4) (1986) 320–330

[13] Huffman, D.A.: A method for the construction of minimum redundancy codes. Proceedings of the IERE **40** (1952) 1098–1101

[14] Cilibrasi, R., Vitanyi, P.: Automatic meaning discovery using google (2004)