Applying Randomized Projection to aid Prediction Algorithms in Detecting High-Dimensional Rogue Applications

Travis Atkison
Department of Computer Science and Engineering
Mississippi State University
Starkville, MS 39762
tla96@msstate.edu

ABSTRACT
This paper describes a research effort to improve the use of the cosine similarity information retrieval technique to detect unknown, known or variances of known rogue software by applying the feature extraction technique of randomized projection. Document similarity techniques, such as cosine similarity, have been used with great success in several document retrieval applications. By following a standard information retrieval methodology, software, in machine readable format, can be regarded as documents in the corpus. These "documents" may or may not have a known rogue functionality. The query is software, again in machine readable format, which contains a certain type of rogue software. This methodology provides an ability to search the corpus with a query and retrieve/identify potentially rogue software as well as other instances of the same type of vulnerability. This retrieval is based on the similarity of the query to a given document in the corpus. To overcome what is known as the 'the curse of dimensionality' that can occur with the use of this type of information retrieval technique, randomized projections are used to create a low-order embedding of the high-dimensional data. For our experiment, we obtain Microsoft Windows applications, infect a subset of them with several common Trojans and apply our dimensionality and prediction methodology. Preliminary results show promise when applying randomized projections to cosine similarity in both speed of prediction and efficiency of required space when compared with using only cosine similarity.

Categories and Subject Descriptors
D.2.0 [**Software Engineering**]: General – Protection mechanisms

General Terms
Security

Keywords
Rogue software detection, information retrieval, n-gram analysis, cosine similarity, randomized projections

1. INTRODUCTION
In general, a consumer, whether corporate or private, must depend on some other entity to deliver their software needs. This software can come from several different sources ranging from software development firms to downloading freeware from the Internet. These software needs include, but are not limited to, the operating system, virus detection engines, firewall systems, even the latest database software to hold the consumer's music files. Even specialized applications for corporate business needs are often purchased rather than developed "in-house." The outsourcing of application development coupled with globalization of the software development market means that where software is being developed and by whom is becoming more and more abstract. This presents a difficult security problem for the consumer in that developers of a computing application may not have the same philosophies or views as the user of the application. An example scenario might be as follows: a virus detection developer provides a consumer with a detection engine that ignores certain viruses or reports that the machine is free of viruses. Another example could involve a firewall software developer manipulating the consumer's system to report back to an outside entity information regarding network traffic data that passes through the firewall, without the consumer's consent or knowledge.

No one is immune from these malicious attacks; from the corporation to the unsuspecting home user. To combat these attacks on a system, users have turned to anti-virus software which contains virus detection engines. "A large percentage of the security software industry is built on the practice of looking for the digital patterns (signatures) that identify known threats." [21] According to the 2007 CSI Computer Crime and Security Survey, anti-virus software, which is a signature based solution, accounted for 54.3 percent of the total budget for industry software security in 2005. [21] Though good at what they do, virus detection engines rely on a database of signatures to detect known rogue applications. Signature based systems inherently limit the detection of new and previously unknown types of rogue attacks. To that end there have been several research attempts to overcome these limitations. One notable avenue of exploration has been to develop tools and techniques based off the field of information retrieval. Previous attempts [7, 11, 12, 20] to use

methodologies and techniques from information retrieval and data mining have had some success, but all have been subjected to the 'curse of dimensionality'. The 'curse of dimensionality', first referred to by Bellman [4], generally describes the computational issues related to performing mathematical operations within an extremely high dimensional space. A capability to reduce the number of features to a more manageable number is very useful in countering this problem. Making any decisions based on this high-dimensional data will require the construction of a low-dimensional embedding that preserves the underlying "structure" hidden in the data. This research will use a technique called randomized projection [15, 17] to create the low-dimensional embeddings. The goal of our work is to provide a mechanism for reducing high-dimensional data to a more manageable dimension. Information retrieval techniques can, then, be applied with better results in both speed to solution and accuracy of prediction creating a more robust rogue application detection capability.

The following section provides a background description of information retrieval, randomized projection and discusses rogue software vulnerabilities. In Section 3, the experimental design of our work is discussed including the software and data used. In Section 4, results achieved are described. Finally, in Section 5 the conclusion and future directions are presented.

2. BACKGROUND

Evaluating the effectiveness of a potential solution to the rogue software detection problem, in which a low-dimensional embedding is used to reduce the dimensions of an information retrieval technique, is an important direction in host security research. Below, a description of the information retrieval technique, the dimensionality reduction method and rogue software vulnerabilities used in our experiments are described.

2.1 Information Retrieval

Information retrieval traditionally is the "part of computer science which studies the retrieval of information (not data) from a collection of written documents." [3] These retrieved documents' aim is to "satisfy a user's information need." [3] The process can be thought of as combing through a set of documents, called the corpus, to find a certain piece of information that has a relationship to a given entity, called the query. That piece of information can either be an entire document, set of documents or a subset of a document. Within the information retrieval community several methods exist for finding these pieces of relevant information. These methods include vector space models, latent semantic indexing models and statistical confidence models as well as others. "Vector space models are the first approach to represent a document as a set of terms." [16] As their name implies vector space models represent their data as a vector with each dimension being defined as a term which may or may not have a weight associated with it. [22] One of the most common vector space models is cosine similarity. Cosine similarity determines the similarity between two data vectors by measuring the angular distance between them. "Cosine has the nice property that it is 1.0 for identical vectors and 0.0 for orthogonal vectors." [23] The following is the formula used in our work for computing cosine similarity;

$$\textit{Cosine Similarity (Q, D)} = \frac{\sum_i w_{Q,i} w_{D,i}}{\sqrt{\sum_i w_{Q,i}^2} \sqrt{\sum_i w_{D,i}^2}} \quad (1)$$

This formula computes the similarity between a query *Q* and a document *D*. It does so by summing the individual components of the two entities represented in the formula as *w*. The individual components for this research, *w*, are defined as n-grams. An n-gram is "any substring of length *n*." [3] Here the gram (which will be the composite of the substring) is a byte in hexadecimal form. Therefore, w_{Qi} is the weight of the i^{th} n-gram in the query and w_{Di} is the weight of the i^{th} n-gram in the document.

There have been other efforts [1, 2, 7, 10, 18, 20] to use the information retrieval concept of n-grams as a feature generator. Henchiri et. al. [7] and Abou-Assaleh et. al. [1, 2] both use the Common N-Gram (CNG) analysis method, which uses the most frequent n-grams to represent a class, to detect rogue applications. Henchiri further limits the number of features by imposing a "hierarchical feature selection process". [7] Marceau [18] suggests an interesting modification on the problem of using n-grams as features by having "multiple-length" grams instead of the tradition single n-length gram. Marceau does this by first creating and then compacting a suffix tree to a directed acyclic graph (DAG). [18] Reddy et. al. [20] develop their own unique n-gram feature selection measure called, 'class-wise document frequency.'

2.2 Randomized Projections

Rogue application detection, following the genre of information retrieval, suffers from the problem that the data, once processed, is encoded in extremely high dimensions. This high-dimensional data limits the kind and amount of analysis that can be preformed. One method for dealing with reduction of this type of high-dimensional data is known as feature extraction. Feature extraction transforms, either linearly or non-linearly, the original feature set into a reduced set that retains the most important predictive information. Examples of this type include principle component analysis, latent semantic analysis and randomized projection.

In randomized projection, "the original high-dimensional data is projected onto a lower-dimensional subspace using a random matrix whose columns have unit lengths." [5] This type of projection attempts to retain the maximum amount of information embedded in the original feature set while substantially reducing the number of features required. This feature reduction will allow for greater amounts of analysis to be performed. The core concept has been developed out of the Johnson-Lindenstrauss lemma [8] which states that any set of n points in a Euclidean space can be mapped to $\Re^t$ where $t = O(\frac{\log n}{\varepsilon^2})$ with distortion $\leq 1 + \varepsilon$ in the distances. Such a mapping may be found in random polynomial time. A proof of this lemma can be found in [6].

There have been some efforts [5, 17, 19] that look at using randomized projection techniques for dimensionality reduction. "Randomized projection refers to the technique of projecting a set

of points from a high-dimensional space to a randomly chosen low-dimensional subspace or embedding." [25] Minnila et. al. [17] are using random projection techniques to map sequences of events and find similarities between them. Their specific application is in the telecommunication field looking at how to better handle network alarms. Their goal is to "show the human analyst previous situations that resemble the current one" [17] so that a more informed decision about the current situation can be made. Though their proposed solution is not perfect, it does show the promise of using randomized projections in a similarity based application.

Bingham et. al. [5] applies randomized projections to an image and text retrieval problem. In comparison to this research problem, their dimensions are not as large, 2500 for images and 5000 for text but the results are still significant. The purpose of their work was to show that compared to other more traditional dimensionality reduction techniques, such as principle component analysis or singular value decomposition, randomized projections offered a greater detail of accuracy. The authors were also able to show that there was a significant computation saving by using randomized projections over other feature extraction techniques, such as principle component analysis.

In another text retrieval application, Kaski [9] successfully applied randomized projections in his text retrieval application that used WEBSOM, a graphical self-organizing map. Again Kaski turned to randomized projection as a method to overcome the computation expense that made other dimensionality reduction techniques infeasible when handling high-dimensional data sets. After incorporating randomized projection into their tool the authors gained an additional 5% increase in classification and topic separation than in previous methods used. [9]

The following efforts [13, 14, 19] use randomized projection in conjunction with latent semantic indexing. Papadimitriou et. al. [19] looking at another information retrieval technique shows positive results in using randomized projections as a pre-processor to the computationally expensive Latent Semantic Indexing. By simply applying randomized projection to their data before computing the Latent Semantic Indexing, their asymptotic running time for the overall system improved from $O(mnc)$ to $O(m(\log^2 n + c\log n))$, where *m* and *n* are the matrix size, *c* is the average number of terms per document. [19]

2.3 Rogue Software Vulnerabilities

Today rogue software vulnerabilities come in all "shapes and sizes," from buffer overflows to injection attacks to information leakage attacks. As noted above there are several instances of these vulnerabilities. Our concentration is on information leakage vulnerability attacks.

Information leakage can be defined as when "non-public" information is released (or leaked) without the information owner's knowledge or consent. An information leakage vulnerability can be introduced within an application at design time through malice or through poor programming practices (intentional versus accidental). It can also be introduced by a rogue attacker after deployment by being bundled with, or concealed within, a seemingly non-threatening application. Symantec reported in their bi-annual threat report for the first half of 2005, that "six of the top ten spyware (information leakage) programs were delivered to their victim by being bundled with some other program." [24]

There are several methods by which a rogue attack can induce a victim's computer to leak information without the knowledge or consent of the user. A notable example of this is the introduction of key stroke loggers into an application. A key stoke logger is software that will record every key that is typed on the user's computer. Our research not only looks at key stroke loggers but also CD key stealers and password stealers. CD key stealers browse through the victim's computer registry looking for serial numbers for any CD's that the victim may have installed and registered. Password stealers work in a similar way but are geared specifically for detecting and extracting account passwords, such as AOL, Yahoo and MSN. Each of these stealers leaks its illicitly gathered information by packaging and sending it to the attacker through email or directly to an FTP server.

Our research concentrates on detecting rogue applications before execution while still packaged in their transporter. This transporter is often called a Trojan horse and the rogue package is referred to as a Trojan. A Trojan horse, similar to the myth, may provide a useful service (for example, a calculator or Notepad) but once executed performs harmful actions. We investigate a specific kind of Trojan horse known as a *binder* or *dropper*. Binders are applications that have the ability to combine (or bind) two or more applications together, yet allow them to run autonomously when executed. This autonomous nature allows the attacker to place a non-threatening, useful service together with one or more rogue applications. The unsuspecting user then executes the application expecting only the useful application; however, unbeknown to them the rogue application(s) are also executed.

3. EXPERIMENT

The following provides a description of the components of the experimental methodology we use. Details of the software application that we developed as well as a description of the data set that was used in the experiments are described below. This section concludes with an overall experimental design description that provides a description of how the experiments were conducted.

3.1 Similarity Software

The software created for this experiment provides functionality to ingest Windows formatted binary executables and then creates an *m*-dimensional data space that contains vectors representing those applications. In these experiments, *m* is the number of total possible n-grams that can be extracted from the ingested applications, one dimension for each possible n-gram. The information stored in each of the dimensions can take on one of several possible values: the absolute total number of occurrences of the particular n-gram in the application, the normalized value of the total number of occurrences of the particular n-gram in the application, or finally, a 1 if the application contained the particular n-gram or a 0 if it did not. Once the *m*-dimensional vectors have been created, the randomized projection matrix algorithm is then applied. The random matrix is populated by selecting vectors that are normally distributed, random variables

with a mean of 0.0 and a standard deviation of 1.0. The result is a low-dimensional embedding of the original high-dimensional features. Then the cosine similarity algorithm is applied to the query application's vector and the corpus applications' vectors. The cosine similarity algorithm followed is the same as shown in Equation 1 above. A special feature of this software is that it has the ability to shift the n-gram window not only by the more traditional byte offsets but also by bit offsets. This allows for a more fine grain tuning of the vector values, e.g., if the rogue adversary performs bit shifting on the rogue applications. It also provides for more accurate similarity result calculations.

3.2 Data

The data used for this experiment consisted of 267 Windows formatted binary executable files that were randomly chosen from a Windows XP operating system. These files ranged in size from 50KB to 500KB. Integrated within the corpus were 24 files that had been infected with rogue code using the F.B.I. (Finding, Binding and Infecting) binder and six standalone rogue applications for a total of 30 rogue applications. The Windows applications infected for this experiment were Microsoft Calculator, MS-DOS Command Prompt, Microsoft Notepad and Microsoft 3D Pinball for Windows. The rogue applications used were the CDKey Harvester v0.9, Fearless KeySpy v2.0, LttLogger v2.0, HermanAgent v1.0, ProAgent v2.0 and Recon v2.0. Each docile application was infected with each of the rogue applications using the F.B.I. (Finding, Binding and Infecting) binder to create 24 infected files. The binder and all rogue applications are freely available for download from the following website, http://www.trojanfrance.com. Table 1 contains short descriptions of the rogue applications used in this experiment.

Table 1. Descriptions of rogue applications

CDKey Harvester v0.9	searches victim's registry for Online Game CD Keys and sends them to the attacker through email
Fearless KeySpy v2.0	keystroke logger
LttLogger v2.0	keystroke logger that can completely remove itself at a specified time or after a specific amount of collection
HermanAgent v1.0	password stealer where information is passed back to the attacker through email
ProAgent v2.0	monitoring and surveillance tool that captures data from webcams, screenshots and microphone usage
Recon v2.0	keystroke logger that can disable anti-virus and firewall software

3.3 Design

This section describes the overall design of our experiment. The size of the n-grams was limited to a 4-byte window. For the dimensionality reduction stage a random matrix was used and projected upon the original high-dimensional data set to produce a new low-dimensional embedding that contained 500 features. The random matrix for the projection was created by randomly selecting values to populate the vectors of the matrix. These values were normally distributed random variables with a mean of 0.0 and a standard deviation of 1.0. The results of these experiments are presented below.

4. RESULTS

To make a valid comparison and to show the value of applying our dimensionality reduction techniques of randomized projection, we compare our results to that of applying the same prediction technique of cosine similarity to the data without any dimensionality reduction. The non-dimensionality reduced data set contains over 7 million features as compared to the dimensionality reduced data set that contains only 500 features.

Tables 2 and 3 depict the performance values for the entire data set without using dimensionality reduction and using random matrix projections respectively. These values include true positive rate (TPR), false positive rate (FPR), accuracy and precision. TPR, as known as recall, is the ratio of positive instances that were correctly identified. FPR is the ratio of negative instances that were incorrectly identified. Accuracy is the ratio of the number of positive instances, either true positive or false positive, that were correct. Precision is the ratio of predicted true positive instances that were identified correctly.

Table 2. Performance Values without Dimensionality Reduction

Performance Metric	**Threshold Values**		
	0.5	**0.3**	**0.1**
TPR	0.44	0.7	0.72
FPR	0	0.002	0.01
Accuracy	0.94	0.97	0.96
Precision	1	0.98	0.9

Table 3. Performance Values using Randomized Matrix Projection

Performance Metric	**Threshold Values**		
	0.55	**0.6**	**0.65**
TPR	0.8	0.8	0.63
FPR	0.05	0	0
Accuracy	0.93	0.98	0.96
Precision	0.6	1	1

Note that the accuracy results shown in Table 3, when compared with the results from the data set that did not have dimensionality reduction applied (Table 2), fall within the described error term mentioned in Section 2.2. Also, comparing the best results from each data set (threshold of 0.3 from Table 2 and 0.6 from Table 3) show that our method out performs the non-reduced data set. This can be attributed to the 'curse of dimensionality' complicating the prediction method. Significant gains were made from a computational performance standpoint. The addition of

computing the matrix multiplication to acquire the reduced dimensional data set was minimal and can be improved with further refinements and taking advantage of advances in fast matrix multiplication. Furthermore, obtaining a prediction result for an individual application saw an over 100-time increase. Over a small number of predictions, the minimal time to compute the matrix was absorbed. The data space required to contain the non-reduced feature vectors was a factor of 3 greater than that required to hold the reduced data set.

These results of applying the randomized matrix projection algorithm are significant suggesting that we can maintain a high precision without sacrificing accuracy or TPR. It is important to note that most methods used in previous research, report only accuracy value ratings. However, a high accuracy rate may not tell the entire story. For example, consider Table 2. For a threshold value of 0.5, the accuracy value is high at 0.94 but the TPR rate is only 0.5. The accuracy values reported in the literature range from 93% to 98%, so the results presented with this effort are very comparable. Looking at just the accuracy values of Table 3, one can conclude that this method is successful. More importantly, we can conclude that our results are successful by comparing them with the results from the non-reduced feature vectors.

5. CONCLUSION

These results support our hypothesis that applying the technique of random matrix projection as a dimensionality reduction method for the cosine similarity metric has merit in determining if an application may contain a rogue application. We have shown that our solution falls within a suitable error range and that the results generated are accurate, comparable, and in some cases better than other suggested solutions in the literature as well as comparable to results generated without using the reduction technique.

There is no claim that this is a complete solution, rather a tool designed to fit into the security administrator's toolbox as a data point or first pass to help reduce the number of applications needing review. This potential reduction in number of applications to sort through can provide an administrator or analyst with valuable time saving by not having to analyze applications that clearly do not contain rogue software. With more and more applications not being developed "in-house" this is a positive result for those responsible for providing secure solutions.

Future efforts for this research are to expand it with the addition of prediction algorithms from the data mining realm, for example decision trees. Also the author plans to investigate additional dimensionality reduction methods and techniques in order to further expand and enhance the analysis capability. Additional research is also planned into determining the threshold values for the similarity algorithm. Determining the key factors in choosing an optimal threshold value is crucial, as can be seen above, to gaining high confidence and to the success rate of the algorithm.

6. ACKNOWLEDGMENTS

The author would like to thank Dr. Ray Vaughn for his insight and thoughtful review. The author would also like to thank Rebekah Atkison for reviewing this document for its grammatical content. This work was partially supported by the National Science Foundation under grant SCI0430354 04090852.

7. REFERENCES

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "Detection of New Malicious Code Using N-grams Signatures," in *Proceedings of the Second Annual Conference on Privacy, Security and Trust*, New Brunswick, Canada, 2004, pp. 193 - 196.

[2] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based Detection of New Malicious Code," in *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.* vol. 2, 2004.

[3] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Harlow, England: Addison Wesley, 1999.

[4] R. Bellman, *Adaptive Control Processes: A Guided Tour*: Princeton University Press, 1961.

[5] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245-250, 2001.

[6] S. Dasgupta and A. Gupta, "An elementary proof of the Johnson-Lindenstrauss Lemma," Interantional Computer Science Institute, Berkley, California, USA 1999.

[7] O. Henchiri and N. Japkowicz, "A Feature Selection and Evaluation Scheme for Computer Virus Detection," *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 891-895, 2006.

[8] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189-206, 1984.

[9] S. Kaski, "Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering," *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, vol. 1, 1998.

[10] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. J. Tesauro, and S. R. White, "Biologically inspired defenses against computer viruses," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intellgence*, San Francisco, CA, 1995, pp. 985 - 996.

[11] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," *The Journal of Machince Learning Research*, vol. 7, pp. 2721-2744, 2006.

[12] J. Z. Kolter and M. A. Maloof, "Learning to Detect Malicious Executables in the Wild," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* Seattle, WA, USA: ACM Press, 2004, pp. 470-478.

[13] M. Kurimo, "Indexing Audio Documents by using Latent Semantic Analysis and SOM," *Kohonen Maps*, pp. 363-374, 1999.

[14] J. Lin and D. Gunopulos, "Dimensionality reduction by random projection and latent semantic indexing," *Proceedings of the Text Mining Workshop, at the 3rd*

SIAM International Conference on Data Mining, May, 2003.

[15] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215-245, 1995.

[16] N. Liu, B. Zhang, J. Yan, Q. Yang, S. Yan, Z. Chen, F. Bai, and W.-Y. Ma, "Learning Similarity Measures in Non-Orthogonal Space," in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management* Washington, D.C., USA: ACM Press, 2004, pp. 334 - 341.

[17] H. Mannila and J. K. Seppänen, "Finding similar situations in sequences of events," *First SIAM International Conference on Data Mining*, 2001.

[18] C. Marceau, "Characterizing the Behavior of a Program Using Multiple-Length N-grams," in *Proceedings of the 2000 workshop on New security paradigms* Ballycotton, County Cork, Ireland: ACM, 2000.

[19] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent Semantic Indexing: A Probabilistic Analysis," *Journal of Computer and System Sciences*, vol. 61, pp. 217-235, 2000.

[20] D. K. S. Reddy and A. K. Pujari, "N-gram analysis for computer virus detection," *Journal in Computer Virology*, vol. 2, pp. 231 - 239, 2006.

[21] R. Richardson, "2007 CSI Computer Crime and Security Survey," Computer Security Institute 2007.

[22] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, pp. 613-620, 1975.

[23] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bulletin of the Technical Committee on Data Engineering*, vol. 24, pp. 35 - 43, 2001.

[24] Symantec, "Symantec Internet Security Threat Report: Trends for January 05 - June 05," September 2005.

[25] S. S. Vempala, *The Random Projection Method*: American Mathematical Society, 2004.