

Machine Learning for Network Intrusion Detection

Jason Bartlett

December 15, 2009

Contents

1	Introduction	1
2	Network Security	1
3	Reviewed Work	2
3.1	Machine Learning in Intrusion Detection	2
3.1.1	Abstract	2
3.1.2	Methods and Results	2
3.2	Active Learning for Network Intrusion Detection	4
3.2.1	Abstract	4
3.2.2	Methods and Results	4
3.3	Language Models for Detection of Unknown Attacks in Network Traffic	5
3.3.1	Abstract	5
3.3.2	Methods and Results	5
4	Conclusions	6

1 Introduction

The field of network security has been growing by leaps and bounds over the past few decades. The laissez-faire attitude of the original network administrators has been replaced by an urgent and pressing need for constant monitoring and quick responses to not only known attacks, but to novel ones that the software designers are unaware of. This paper gives an overview of modern network security in Section 2 first, then reviews several ML-related intrusion detection strategies and algorithms in Section 3. My conclusions follow in Section 4.

I chose this topic because of the seeming neverending reports of computer systems and networks being taken over or exploited. I feel as though there is a vast disconnect between the current state of computer security and where it could be. By researching where the field is at, I hope to find ideas for my own research. Ultimately, I want to be at a point where I can contribute something that will make a deep impact on this field. As far as this assignment goes, that consists mainly of educating myself to recent advances in theory and technology.

2 Network Security

Computers today are interconnected in ways that the earlier users could not have imagined. With that interconnectedness comes an entire array of vulnerabilities that threaten everybody from the everyday user to large corporations and even entire governments. While initial attacks were designed more for fun than for profit, recent years have seen that malicious activity is now being carried out by a sophisticated and somewhat organized class of cybercriminal.[1] This of course necessitates advanced security to protect the

systems and information being targeted by this criminal element. To date, the primary method used by network intrusion detection systems (IDSs) is anomaly detection, where incoming traffic is compared to known patterns of attack traffic. These known patterns are called attack signatures.[1, 2, 3] However, this method is becoming increasingly unable to keep up with the novel attacks and zero-day exploits being discovered by the hacker community.[1] In this paper I will discuss several new algorithms proposed to improve the abilities of IDSs to identify attacks and modify their models to catch future attempts of similar attacks.

3 Reviewed Work

3.1 Machine Learning in Intrusion Detection

3.1.1 Abstract

This Ph.D. dissertation by Yihua Liao at the University of California (Davis) examines several techniques aimed at detecting hostile activity on monitored systems. First, a new algorithm for detecting intrusive process behaviour is introduced and tested. In the algorithm, a process' trace of system calls is converted to a vector and compared to known process behaviour. Using a k -Nearest Neighbor classifier, he is able to "effectively detect intrusive program behavior while a low false positive rate is achieved." [2] The next section introduces an "adaptive anomaly detection framework" that aims to not only improve detection of abnormal program behaviour, but also to adapt to new, normal user behaviour patterns without losing previously gained knowledge. He uses this framework to decrease the false-positive rate of his detection algorithm. The third section deals with "methods to efficiently estimate the generalization performance of an anomaly detector and the training size requirements." [2] Finally, he examines a cost-benefit analysis for an IDS and integrates the "cost-effectiveness and technical performance" of the IDS.

For purposes of this paper, I focused primarily on the theoretical sections; namely the new k -NN algorithm and the adaptive detection framework.

3.1.2 Methods and Results

The first experiment performed in this dissertation deals with a new algorithm for detecting abnormal program behaviour. The difference in this new algorithm from previous related work is that instead of building local system call sequences for all the different processes and comparing users' call traces to the expected normal behaviour, he uses text categorization techniques, namely weighting system calls by their frequency in a given process' normal behaviour, to reduce the computations needed. Once users' processes are converted to similar vectors, he uses a k -Nearest Neighbor classifier to determine if the program is running normally or if there is an anomalous pattern of system calls.

For the experiment, Liao uses process data from the DARPA 1998 dataset. From the connection data, he extracted only the names of system calls to construct the dictionary of words necessary for the text-categorization vector space conversion. Once that preprocessing is complete, he trains the classifier on data that is completely attack-free. Since k -NN is a memory-based model, there really is no training step like what one would have to do for a neural network, for instance. Once the training data is established, the algorithm then tests new data against the training set. If there is an unknown system call, the process is automatically labeled as anomalous. Otherwise the k nearest neighbors are found and compared to the new data. If the new process is above some similarity threshold, the new process is labeled normal and anomalous otherwise.

The first step of his experiment is to find an optimal value for k such that the classifier is detecting 100% of attacks with the lowest false-positive rate. This data is presented as ROC curves and is shown on page 30 of his paper.[2] His finding was that for $k=10$, the detection accuracy reached 100% with the lowest false-positive rate (0.44%). He also compared the accuracy of the model with two different weighting strategies for the vector conversion. He found that term frequency-inverse document frequency ($tf \cdot idf$) weighting had a lower false-positive rate for 100% detection than frequency weighting.

After establishing that k -NN can effectively detect anomalous program behaviour, his next step is to add signature verification to the classifier. He does this by including class data in the training set, and changing

the rules of classification to immediately label a process malicious if its nearest neighbors are malicious. Otherwise, it uses the same algorithm as his initial experiment to determine a process' legitimacy. In this algorithm, a process' system calls are first compared to known malicious processes. If the similarity is 1, the process is classified as malicious. Otherwise, the previous algorithm is applied. Using this new algorithm on the DARPA 1998 data, he was able to correctly identify 91.7% of all attacks with a false-positive rate of either 0.59% or 0.89% for two different weightings of the system call data. The attacks that the classifier missed were denial-of-service (DoS) attacks that repeatedly call a legal process until the system is overloaded. Since the process being used is legal, this classifier could not distinguish it, but Liao notes that because of the high frequency of the process call, this attack could be detected by a different model.

The next chapter of Liao's dissertation presents two new hurdles when it comes to intrusion detection: concept drift and the need for online learning. The idea of concept drift states that the source of data itself is not static. In most machine learning algorithms, an underlying assumption about the data is made. For instance, in QDA or LDA, the data is assumed to come from different normal distributions with more or less set means and standard deviations. For process data, a user's normal behaviour can change over time, and so a classifier trained at one point in time may not correctly classify new patterns in behaviour. Liao's idea on this topic is to use an evolving connectionist system, which is a form of artificial neural network. In this type of network, the inputs are fed into the network in an online fashion and assigned to one of several patterns known by the network. In a general sense, the input layer is the same as the hidden layer that our research has used, and the pattern layer represents the output layer. In a system such as this, however, the weight vectors on the pattern layer can be exported and used to identify just what the network is learning. If a new input vector is close enough to an existing pattern (determined by some distance measure), then the new sample is assigned to that pattern and the weight vector for that pattern is updated to accommodate the new data. If the new data is not close enough to an existing pattern, a new pattern is created for the new data.

This is how Liao's system accounts for changes in user behaviour while still remembering the old behaviour patterns. When a new vector is passed into his system, it is assigned to the *normal* class if it is above some similarity threshold to an existing *normal* pattern. Otherwise it is labeled *uncertain* and is assigned to either an existing *uncertain* pattern or a new pattern is created for it. The system also generates a Level 1 alarm to notify a system administrator of a potential attack. To determine whether the new data is actually malicious or the beginning of a new normal behaviour pattern, the next few data samples are monitored. If enough of them correspond to the new, uncertain pattern, then the pattern and all the member vectors are changed to *normal*. If there are not enough subsequent vectors to satisfy that threshold, however, the pattern is destroyed and all the members of it are reclassified as *anomalous*. When that happens, Liao's system dispatches a Level 2 alarm as well to receive further actions from the system administrators.

Liao then performs experiments to tune the parameters of the algorithms Fuzzy Adaptive Resonance Theory (Fuzzy ART) and Evolving Fuzzy Neural Networks (EFuNN) after adapting them to the framework defined above. He also compares these evolving connectionist systems to static learning with Support Vector Machines (SVMs). In the first experiment, Liao uses a subset of the KDD Cup 1999 dataset to test the effectiveness of the evolving connectionist systems. The data did come with training labels, but they were not used since the systems themselves learn the patterns on-the-fly. First, Liao varies the vigilance threshold, that is the parameter that determines how close to existing patterns a new vector must be in order to be classified normal. Using the cost function $Cost = (1 - hitRate) + \gamma(falsepositiverate)$ to establish a uniform measure of performance, he found that Fuzzy ART minimized its cost at a vigilance of 0.93 and EFuNN reached its lowest cost at 0.96. At these values, the two systems reached the optimal balance of correct anomaly detections (86.3%*FuzzyART*, 90%*EFuNN*) and lowest false positive rate (2.35%*FuzzyART*, 1.97%*EFuNN*). Next he varied the learning rate, which controls how much the neural networks change their weight vectors to accommodate new data in a given pattern. He found that a learning rate of 0.1 gave optimal results for both Fuzzy ART (86.3%*hitrate*, 2.35%*FP*) and EFuNN (90%*hitrate*, 1.97%*FP*). The third and fourth parameters Liao varies are the number of new data vectors to watch after a new *uncertain* pattern is created (N_{watch}) and the number of new data vectors that must be classified in that pattern to reclassify it as *normal* (Min_{count}). He found that Fuzzy ART achieved an 86.3% hit rate and 2.35% false-positive rate with $N_{watch} = 8$ and $Min_{count} = 4$. EFuNN made a 88.9% hit rate and 1.53% false-positive rate for $N_{watch} = 4$ and $Min_{count} = 2$. The final experiment in this section compared the results of Fuzzy ART and EFuNN with an SVM model, and found that the SVM system returned over a 12% false-positive rate. Fuzzy ART returned the best hit

rate (94%), while EFuNN returned the best false-positive rate (0.884%). Liao concludes that an adaptive, online learning system for anomaly detection is vastly preferable to a static learning model because of the variability in data over time. Liao conducts one more experiment on a different data set, user process data collected from a real-world government agency. In this data set, he chooses some of the users as normal users and then inserts data from the other users into the normal users' sessions. The goal is to see if the algorithms can distinguish the masquerades from the legitimate users' data. He found that his algorithms based on Fuzzy ART and EFuNN both gave far better results than an SVM model, and that EFuNN returned both a higher hit rate (81.8%) and lower false positive rate (3.42%).[2]

3.2 Active Learning for Network Intrusion Detection

3.2.1 Abstract

Anomaly detection for network intrusion detection is usually considered an unsupervised task. Prominent techniques, such as one-class support vector machines, learn a hypersphere enclosing network data, mapped to a vector space, such that points outside of the ball are considered anomalous. However, this setup ignores relevant information such as expert and background knowledge. In this paper, we rephrase anomaly detection as an active learning task. We propose an effective active learning strategy to query low confidence observations and to expand the data basis with minimal labeling effort. Our empirical evaluation on network intrusion detection shows that our approach consistently outperforms existing methods in relevant scenarios.

[1]

3.2.2 Methods and Results

The primary goal of this paper is to redefine anomaly detection as an active learning task. In most systems that use anomaly detection, normal network traffic is used to train the system and then any deviation from that learned model is anomalous. This paper first introduces basic anomaly detection, then extends it with strategies for active and semi-supervised learning. The overall strategy consists of combining both margin strategy and k -NN to query the network security team for labels for data that lies both close to the hypersphere decision boundary *and* within potentially anomalous clusters. By using both techniques rather than one or the other, the model gains the greatest refinement possible while minimizing the workload on the network team. They add this refinement to the support vector domain description (SVDD) and call the new system ActiveSVDD.

The first experiment this group conducts aims to directly compare the effectiveness of the ActiveSVDD model against the vanilla SVDD algorithm. The group uses actual network data mapped to a vector space defined by 3-grams, that is, the set of all 3-character "words" present in the network packets' payload. To establish a baseline, the group compares ActiveSVDD and vanilla SVDD on data with a varying amount of initially labeled data. They found that both algorithms performed very well, detecting most of the attacks with an extremely low false-positive rate.

To demonstrate the superiority of ActiveSVDD, the group next sets up an experiment similar to the first one, except that in this trial, the malicious payloads are cloaked, that is, normal-looking network headers are included at the top of the payloads to mask the malicious nature of the payload itself. When the two techniques are applied to this new data set, the detection rate of SVDD drops to about 70%, whereas ActiveSVDD takes advantage of the randomly labeled data and quickly improves its accuracy. At about 15% randomly-labeled data, ActiveSVDD was detecting over 95% of the attacks. Furthermore, when ActiveSVDD was allowed to query about borderline data, it reduced the number of necessary labels to achieve the same accuracy. When the active learning strategy was included, only about 3% of the data being labeled was enough to achieve over 95% detection accuracy.

The group's next experiment intends to incorporate an online learning aspect to the ActiveSVDD model. They take a random subset of their network data and break it into chunks, only labeling a handful of each samples in each chunk. They then train the ActiveSVDD on increasing numbers of these data chunks and compare it to vanilla SVDD trained on the same chunks. They find that because vanilla SVDD does not take into account the labeled data, its accuracy drops as the data pool grows, and also that by using the

active learning approach and only labeling 1.5% of the incoming data, ActiveSVDD correctly identifies 96% of the cloaked attacks with a false-positive rate of only 0.15%.

The final experiment the group performs does not involve their ActiveSVDD, but simply extends their initial active learning strategy and applies it to the vanilla SVDD. Their goal is to demonstrate the preferability of an active learning strategy over random sampling with respect to the labeled data. They also use SVDD as a baseline. They find that even small amounts of labeled data will produce at least reasonable results (about 67% accuracy with about 0% false positive rate), whereas a random sampling approach gives a 50% false-positive rate and SVDD completely fails with a 100% false-positive rate.[1]

3.3 Language Models for Detection of Unknown Attacks in Network Traffic

3.3.1 Abstract

In this paper we propose a method for network intrusion detection based on language models. Our method proceeds by extracting language features such as n-grams and words from connection payloads and applying unsupervised anomaly detection without prior learning phase or presence of labeled data. The essential part of this procedure is linear-time computation of similarity measures between language models of connection payloads. Particular patterns in these models decisive for differentiation of attacks and normal data can be traced back to attack semantics and utilized for automatic generation of attack signatures. Results of experiments conducted on two datasets of network traffic demonstrate the importance of higher-order n-grams and variable-length language models for detection of unknown network attacks. An implementation of our system achieved detection accuracy of over 80% with no false positives on instances of recent remote-to-local attacks in HTTP, FTP and SMTP traffic.

[3]

3.3.2 Methods and Results

The techniques presented in this paper differ from previous attempts to use language-based models on network traffic in that Rieck and Laskov want to use actual byte sequences instead of the characters inside the packets themselves. The theory is that because network protocols are subject to a variety of "dialects" that come from different implementations and extensions, attempts to find common rules that encompass all legitimate traffic are stymied in the same fashion as trying to recover the underlying grammatical structure of a human language.[3] Therefore, using the raw bytes and applying principles of language processing could give a more robust way to detect malicious data packets. Specifically, Rieck and Laskov propose to represent n -grams and words drawn from network payloads as tries and they also present a linear-time way to compare tries.

For their experiments, Rieck and Laskov used the DARPA 1999 dataset and created a second data set, which they call PESIM 2005. This data was generated by a server bank running virtual machine instances of Windows, Linux, and Solaris. Their lab teams generated the traffic, and the servers were configured to mirror news sites on HTTP, offer file sharing on FTP, and they artificially injected SMTP traffic from the staff's personal communication and mailing lists as well as spam from several team members. Attacks were generated by a penetration testing expert using the Metasploit framework.

To create testing data for the trie-based algorithms, they converted the TCP connection data into a trie that represents a specific language model, thus generating a set of tries computed over the connection payloads. The test sets were mixed to contain both normal and attack data with no previous learning step. To tune the parameters of the algorithms being compared, the team used a second, independent data subset.

The first experiment intends to find a combination of a similarity measure and anomaly detector that provides the best results, as reported by the area under a ROC curve on the interval $[0,0.01]$. They do not report much on this experiment, except to inform the reader that the Kulczynski coefficient is their best similarity measure and their Zeta anomaly score (an extension of k -NN) proves to be the best algorithm to use as their unsupervised anomaly detector.

Having established the tools they will use for the future experiments, Rieck and Laskov's next experiment is intended to extend a previous result that has been demonstrated for other IDSs, namely that the optimal

n -gram length varies among applications and datasets. They first find the best value of n for each of the three protocols being examined (HTTP, FTP, and SMTP), and then plot ROC curves for each of them. As in the previous work, each protocol does in fact have an optimal n . In addition, they found that their approach in general is promising. On their more recent PESIM 2005 data, the experimental classification approach achieved over 80% on all three protocols with no false positives.

To better understand why there was no optimal n in the previous experiment, the next experiment goes deeper and evaluates the detection accuracy of the algorithm on individual types of attacks within the three protocols. They found that 18 of the 27 attacks were perfectly identified with no false positives, and that for the simpler attacks, the length of n was irrelevant. However, more complicated exploits required larger and larger n , and Rieck and Laskov conclude that setting a fixed n could limit the system's ability to detect novel attacks.

To get around the need for n , the next step in their experiments is to use the variable-length word model instead of n -grams. Because of the syntax required by the HTTP, FTP, and SMTP protocols, they were able to come up with a set of delimiter symbols that they could use to define words in the connection payloads. To determine if this is a better approach than n -grams, they also used multiple anomaly detectors for different values of n and combined the scores returned from it. For further comparison, this experiment also pitted the algorithms against the open-source IDS Snort, which uses attack signatures to detect attacks. They found that, while the multiple n -gram anomaly detectors gave the best result, the word model gave an acceptably high detection rate for a large reduction in overall computations. Both algorithms outperformed Snort, usually by wide margins. This indicates that, despite being a recent release and the developers being aware of the attacks being tested, signature-based IDSs are limited by how up-to-date their signature repositories are.

After demonstrating the validity of their ideas, Rieck and Vaskov then show how the difference in anomalous traffic and normal traffic can be represented as a frequency difference plot. Since n -grams or words are the basis of how traffic is compared, n -grams or words that do not show up in normal traffic are more likely to be part of an attack, and so can be flagged for future dissection by the network security analysts.

Finally, Rieck and Laskov present a language-based model for generating attack signatures for novel attacks. With enough instances of the same type of attack, one can combine the tries for the different attack instances and prune off any branches that only occur in single instances to generate a unique trie that represents what they call an A-signature. One can also do a similar procedure for normal network traffic, and then merge the attack and normal tries to create an AN-signature that will indicate either normal or malicious data, but not both.

They tested these derived signatures on the PESIM 2005 dataset. They found that the A-signatures alone only captured roughly 30-60% of the attacks, whereas AN-signatures detected 80-100% of the attacks with no false positives. Rieck and Laskov caution that because generation of these signatures are somewhat dependent on the local network environment, this model should be applied in a semi-automatic manner, where a network security analyst is looking over the signatures to make sure the IDS is not overfitting.[3]

4 Conclusions

All three of these papers presented valuable insight into the field of computer security for me. Liao's dissertation was the most dissimilar of the three, since his work focused mostly on process data and not network payloads. However, the language basis of the work is similar across all three papers. The deconstruction of whatever data set being examined, whether process data or network data, into n -grams or words seems to be the prevalent theory in intrusion detection these days. What seems to be changing the most is the automation of the process. In [3], the role of the human security analyst is presented a couple of times as still having the final say in what the IDS is looking for. In [1], on the other hand, the system is being taught to ask questions, not just take direction. Incidentally, I discovered that Rieck and Laskov's paper [3] is cited by [1], and that Rieck worked on both. Obviously the research is still ongoing. I want to know how well Rieck and Laskov's approach would work on other network protocols. Attacks are available in far more avenues than just HTTP, FTP, or SMTP.

Liao's paper presented a very interesting approach to the bag-of-words representation. Instead of looking at the data itself, or the order that the system calls come in, he instead just weighted each call by its

frequency in the overall process. That seems a little counterintuitive to me at first glance, because it would seem like the order of system calls is just as important as the calls themselves. If an attack could be derived that had the same proportions of system calls in a different enough order to cause a problem, then his approach would miss it.

I see some similarity between the adaptive anomaly detection framework presented by Liao and the ActiveSVDD algorithm. The neural network that Liao bases his evolving connectionist system on modifies the weights of a given pattern when a new data vector is classified as belonging to that vector. The ActiveSVDD algorithm, on the other hand, simply creates a hypersphere around the normal data and tweaks the boundaries of it when it sees a data vector near its boundary. I think that if the hypersphere and the pattern boundaries of Liao's system were overlaid, they would likely overlap. Liao's system is designed to handle drift in normal behaviour by adding new patterns to his neural network, but the hypersphere in ActiveSVDD can do the same thing. If ActiveSVDD were to get input from a new protocol near the decision boundary, then it would ask the operator if the new data is normal or not. When the new data is labeled normal, then, the hypersphere will expand there to accomodate the new data. In this way, both algorithms can be taught to handle unknown, but normal, data. The drawback to Liao's system is the set manner in which it decides that a new pattern is normal. With sufficient concurrent attacks, I think Liao's system could be fooled into accepting an attack pattern as normal, whereas even if ActiveSVDD gets a bunch of concurrent attacks, the operator is still going to label them anomalous.

The one main problem I could see in Rieck and Laskov's language-based signature generation is the need for multiple instances of the same attack to create the A-signature. If the attack is not widely known, then it is possible that there will not be enough tries that contain that attack's n -grams to generate a signature. On the other hand, as soon as ActiveSVDD sees the new attack, even if there are no other instances of it readily available, either it will be far enough from the hypersphere to be immediately classified anomalous or the operator will have a chance to look at the data and make that call without needing more attempts against his network to know he's looking at an attack.

References

- [1] Nico Gornitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Active learning for network intrusion detection. Oct 2009.
- [2] Yihua Liao. *Machine Learning in Intrusion Detection*. PhD thesis, University of California (Davis), 2005.
- [3] Konrad Rieck and Pavel Laskov. Language models for detection of unknown attacks in network traffic. *Journal of Computer Virology*, 2:243–256, Dec 2006.