A Learning System for Discriminating Variants of Malicious Network Traffic

Justin M. Beaver
Oak Ridge National Lab
(865) 576-0327
beaverjm@ornl.gov

Christopher T. Symons
Oak Ridge National Lab
(865) 241-5952
symonsct@ornl.gov

Robert E. Gillen
Oak Ridge National Lab
(865) 250-9657
gillenre@ornl.gov

ABSTRACT

Modern computer network defense systems rely primarily on signature-based intrusion detection tools, which generate alerts when patterns that are pre-determined to be malicious are encountered in network data streams. Signatures are created reactively, and only after in-depth manual analysis of a network intrusion. There is little ability for signature-based detectors to identify intrusions that are new or even variants of an existing attack, and little ability to adapt the detectors to the patterns unique to a network environment. Due to these limitations, the need exists for network intrusion detection techniques that can more comprehensively address both known and unknown network-based attacks and can be optimized for the target environment.

This work describes a system that leverages machine learning to provide a network intrusion detection capability that analyzes behaviors in channels of communication between individual computers. Using examples of malicious and non-malicious traffic in the target environment, the system can be trained to discriminate between traffic types. The machine learning provides insight that would be difficult for a human to explicitly code as a signature because it evaluates many interdependent metrics simultaneously. With this approach, zero day detection is possible by focusing on similarity to known traffic types rather than mining for specific bit patterns or conditions. This also reduces the burden on organizations to account for all possible attack variant combinations through signatures. The approach is presented along with results from a third-party evaluation of its performance.

Categories and Subject Descriptors

C.2 [Computer-Communication Networks]: General| security and protection; I.2 [Artificial Intelligence]: Applications and Expert Systems

General Terms

Algorithms, Performance, Design, Security.

Keywords

Intrusion Detection, Machine Learning, Computer Network Defense.

1. INTRODUCTION

Modern Computer Network Defense (CND) architectures rely heavily on signature-based intrusion detection systems (IDS), where human analysis is required to encode patterns that are necessary conditions for the detection of network-based attacks. However, the efficacy of the signature-based approach has diminished as the complexity of networks and the quantity of new network attacks has increased. As described in [7], the false positive rate of such systems, combined with the traffic volume, results in a set of alerts that are inadequate as reliable indicators of intrusions. For enterprise-level CND to remain effective, an adaptive system for network intrusion detection, one that can learn system/network behavior, is needed to provide effective decision support to operations personnel. This has been the motivation for research and development of network intrusion detection systems (NIDS) that leverage machine learning methods to discriminate between malicious and benign network traffic.

Machine learning can be roughly characterized as a field of data analysis that involves learning from examples, and attempting to generalize this knowledge to unseen events in the same way a person would. In the cyber domain, machine learning methods have the potential to outperform the human expert, or dramatically augment their effectiveness, due their ability to accommodate the complexity and size of the data. These methods have proven to be extremely effective tools facilitating a powerful combination of data analysis and expert knowledge.

Machine learning is different from anomaly-detection, and even most general definitions of data mining and pattern recognition. While the terms are often used interchangeably in many application domains, machine learning is distinct from the others. Most importantly for cyber security, it involves generalization performance as a foundational principle in the algorithms that it deploys. This approach allows a machine learning method to avoid being fooled by simple patterns, and often allows it to operate well on data that doesn't resemble anything it has seen before. In fact, even if the algorithm and training data are known to the adversary, if the data size and feature set are large, it would still be an incredibly difficult task to formulate ways around such a detector. Rather than making a simple modification to a known attack pattern, the adversary now has to try to make his actions look more like normal traffic than attack traffic while still attempting to perform functions that result in an effective attack. Moreover, this type of defense improves its performance over time, as larger numbers of attack examples are used to train the machine learner.

There are several advantages to machine learning NIDS [2], including minimizing reliance on human analysis and adaptation to local network environments. The machine learning provides an insight that would be difficult for a human to explicitly describe as a rule or signature because it can potentially evaluate thousands of interdependent metrics simultaneously. Unlike signature-based systems, zero-day (previously unseen) attack detection is possible because traffic is classified based on its similarity to known types rather than using patterns specified in signatures. These systems are also particularly effective in detecting variants of attacks [4], which are small changes in a known attack vector created to bypass signature-based sensors.

Prior work in network traffic data analysis includes the application of various machine learning methods in order to discriminate traffic types. In some applications, this approach is used to classify traffic for the purpose of discovering the type of service being used, as in [3]. However, most applications are focused on discriminating malicious network traffic from non-malicious traffic. An extensive review of these approaches is described in [1], including both supervised and unsupervised methods such as support vector machines, naive Bayes, clustering, decision trees and random forests. The state-of-the-art studies follow a familiar evaluation pattern where the method is applied in isolation to a network traffic data set and cross-validated or compared with competing methods. While technically sound, these studies struggle to be practical because their evaluations are based largely on academic data.

In this work, we describe a machine learning network intrusion detection system designed for operational deployment. The contribution of this system is the use of the AdaBoost ensemble learner as a reliable discriminator of malicious traffic in realistic network settings. We describe an architecture that functions in near real-time, provides a tractable learning process, and is evaluated on a high fidelity simulation that closely mimics an existing operational network.

2. TECHNICAL APPROACH

We developed a machine learning system to discriminate malicious network traffic from benign network traffic. The analysis approach is shown in Figure 1.

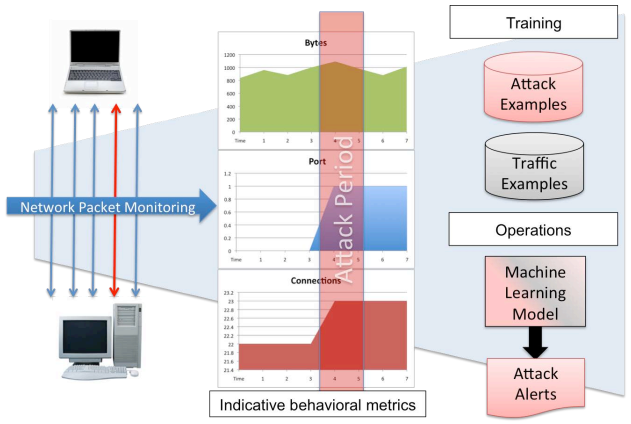


Figure 1 Analysis Approach to Discriminate Malicious Network Traffic

Channels of communication between computers, characterized by network flows, are tracked and monitored by the system in near real-time. Metrics, also called *features*, that are significant behavioral indicators, are analyzed simultaneously. During training of the machine learning models, the system uses collections of metrics (features) that have been labeled a priori as either malicious or benign and stores them as examples of those traffic types. From these examples, models that generalize the encoded knowledge are built to characterize and discriminate between the various types of learned network traffic. In operations, these models are used to classify previously unseen sets of features encountered in the real-time traffic. The architecture design is detailed in the subsections below.

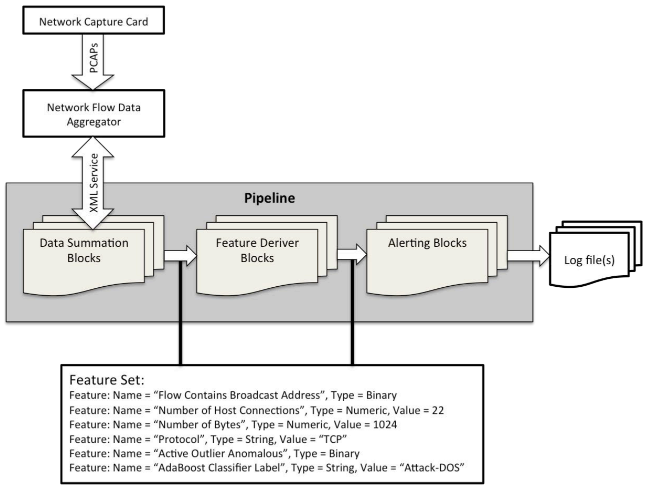
2.1 Feature Set Design

The features selected for this system are intended to characterize network traffic flows, and are an extension to the run-time calculable features proposed in the DARPA/1999 KDD Cup data set [13]. At the unique flow level, the addresses, ports, and protocol are used to track and manage flow data, but features that indicate the presence of broadcast/routing addresses and whether the information is exchanged with a regular periodicity are examples of the factors used for classification. To capture exchanges between hosts and across the network, there are features that focus on the volume and frequency of transactions and the degree of connectivity of each host, as well as the presence of statistical outliers in numeric metrics such as byte counts and packet counts. Also included in the system's feature sets are features that describe the classification results of upstream learners in the pipeline architecture, as described in Section 2.3.

This system's feature set is comprised of approximately 40 features to characterize unique flows, all of which can be captured or derived from protocol header data. No deep packet inspection is performed, nor is service-specific data used. In addition, while address, port, and protocol features are used to track data associated with unique network flows, we intentionally exclude them from the feature set when making the classification decision. This is to avoid learning environment-specific conditions such as attacks originating from the same host, which is a common occurrence in range evaluations.

2.2 System Architecture

The architecture of the developed system is shown in Figure 2 and has two primary components: a data acquisition piece that handles the high-speed translation of network data into sets of features, and the analysis pipeline, which processes the features sets resulting in a classification of the observed network traffic.



Feature Set:
Feature: Name = "Flow Contains Broadcast Address", Type = Binary
Feature: Name = "Number of Host Connections", Type = Numeric, Value = 22
Feature: Name = "Number of Bytes", Type = Numeric, Value = 1024
Feature: Name = "Protocol", Type = String, Value = "TCP"
Feature: Name = "Active Outlier Anomalous", Type = Binary
Feature: Name = "AdaBoost Classifier Label", Type = String, Value = "Attack-DOS"

Figure 2 Malicious Traffic Classification System Architecture

The data acquisition portion of the system collects the raw network traffic, using a commercial network capture card, and stores it on disk in the form of packet capture (PCAP) files. As applied to our solution, the card is not configured to perform any aggregation or analytics other than to ensure that the full data rate is being captured and persisted to disk. Once captured, data is aggregated at the flow level (represented by a unique 5-item tuple: source/destination address, source/destination port, and protocol) for a given time window (i.e. 2 seconds). This data is then made available via web service for use by the analysis pipeline.

The core analysis of the data occurs in the analysis pipeline – a software construct with the flexibility to plug in/out various components using a configuration specification. Examples of different pipeline components include data summation blocks, machine-learning based feature extraction/derivation blocks, and alerting blocks. These can be arranged in most any order, although normal operations would dictate a certain flow that generally includes feature extraction and summation blocks proceed alerting blocks. The pipeline is structured such that each stage in its progression builds on the analysis applied in prior stages. Downstream components are informed of the decisions made upstream through of a common medium of exchange – essentially a common message structure that is an extension of a feature set comprised of feature names and associated values. The current implementation of the system utilizes a log-file style of alerts with the assumption that a log-file aggregation system is in place.

2.3 Machine Learning Methods

The pipeline architecture described above is designed to support nearly any type of machine learning algorithm. In this evaluation, the primary model applied relies on the method of Adaptive Boosting (AdaBoost) [8] [9], where many weak classifiers are combined into a committee of models that form an ensemble. As new models are added to the committee during the training process, the AdaBoost method uses importance weighting to focus on examples that remain difficult to classify. This approach is well known for its ability to avoid over fitting the training data, and AdaBoost has been shown to generalize well to unseen data even when using a very complex model based on thousands of underlying models.

The specific implementation that was tested has four levels of decision-making. The top-level model is a wrapper that imposes a cap on the false positive rate on the training data of the underlying model, with the assumption that such a cap will translate at least partially to results on unseen data. Essentially, this approach finds a model that fits the data according to the internal model building process (in this case AdaBoost), but then it adapts the decision threshold of the underlying model in order to reduce false positives at a potential cost of reducing recall. For our runs, we impose a maximum false positive rate of 0.01.

As mentioned above, the next level of the model consists of an AdaBoost [9] ensemble, and we use the version implemented in the Minorthird machine learning library [10]. Since AdaBoost is an ensemble method, there are many options for the models that make up the committee. For the internal models, we use the default implementation of a Decision Tree in the Minorthird library. We selected one thousand rounds of boosting in order to take advantage of the AdaBoost algorithm's ability to increase performance even at high levels of complexity. Based on prior experience, using a number larger than this rarely improves accuracy, but increases training and testing times significantly.

The lowest level of decision-making relies upon an Active Outlier anomaly detection algorithm [11] that has only seen the normal (non-attack) training data. This model provides its judgment of normality as an additional feature input to the higher order Decision-Tree models.

2.4 Training Process

The machine learning model generation and training process is implemented as an extension of the analysis pipeline. Training data are stored in PCAP files that provide examples of known benign traffic in the network environment, and classes of known malicious traffic. Training PCAPs are loaded and processed as they would be during a live system run until the analysis pipeline has produced representative feature sets that characterize all of the benign and malicious traffic processed.

The machine learning model for each learner is then built from these labeled examples. In cases where multiple learners are staged, the training process iterates through the pipeline, creating models for upstream learners and repeating the training process for downstream learners until all required models are built. Of particular importance is the tractability of the learning process. We employ a selective sampling approach that allows for the training process to complete in about 4 hours, allowing it to fit within the concept of operations for most CND systems.

3. EXPERIMENTATION

The system described in Section 2 was evaluated as part of a third-party formal experimentation series. There were three objectives for the machine learning NIDS we developed:

1. Demonstrate an ability to complement the detection capabilities of signature-based IDS systems,
2. Demonstrate an ability to detect zero day attacks (attack vectors previously unseen by the system), and
3. Demonstrate a high detection and low false positive rate.

The experiment was conducted on a controlled range where network traffic was simulated using the Lariat tool developed by MIT Lincoln Labs [12]. The Lariat system emulates complex and varied user actions on actual virtual machines, providing a high fidelity simulation of actual network operations. The CND system defending the simulated network was comprised of a high-performing signature-based sensor and the machine learning NIDS described in this paper. The CND sensors were positioned as boundary protection devices privy to both inbound and outbound information flows.

In situ training of the machine learning system was accomplished with the cooperation of the penetration testing team involved in the experiment. A subset of attacks was used against the defending network as examples of malicious traffic, and then was recorded as PCAP files. Similarly, benign network traffic was recorded during periods where the penetration team was not active. Attacks were performed on top of normal traffic and training samples were tightly trimmed to include the attack and just a few seconds of data on either side. The resultant PCAP libraries were used to generate examples and models as described in Section 2.4.

During experiment runs, the penetration testing team randomly levied attacks against the simulated network. Although a variety of vectors were used, the attack approaches were staged to employ common strategies, including reconnaissance, vulnerability exploitation, and access misuse. At the conclusion of each experiment run, the alert logs produced by the CND components were analyzed and evaluated against the experiment objectives.

The experimental results highlight the promise of machine learning NIDS sensors. The described system was measured to detect 82% of the attacks that were missed by the high-performing signature-based sensor, demonstrating the machine learning sensor's ability to complement existing sensors in a CND system. Furthermore, the machine learning sensor detected 89% of the attacks on which it had not been trained (zero day attacks), yet were levied during the experiment. The reliable detection of previously unseen attacks represents a new capability for CND. Finally, over the course of 18 different experiment runs, the machine learning sensor was measured to have a malicious traffic detection rate of 94% and a false alert rate of 1.8%, which meets the objective of high detection rate and low false positive rate, and shows a significant performance improvement when compared to a recent study where a classifier ensemble was applied to operationally derived feature data [14], and a detection rate of 81% and a false positive rate of 5.9% was reported.

4. DISCUSSION

The results cited above demonstrate tremendous potential for training a useful machine learning network intrusion detector. This approach addresses the weaknesses of the signature-based approach, including reliable detection performance and zero day detection. Training of the learner is tractably performed at the deployment site, which is critical given the idiosyncrasies of individual networks. The learning system was evaluated as part of a real-time network intrusion detection experiment and shown to meet all experiment objectives.

One potential item of concern with the evaluation is that the simulation itself is learned, and thus the operational nature of the levied attack is more easily observed. While this remains a possibility, we have applied a variant of this machine learning approach to operational data from the University of Tokyo [5] with similar success. See [6] for an account of these results.

Another potential concern is the real-time performance of the system relative to network throughput. The data rate of the traffic produced using Lariat intentionally hovered around 1.5-2 Mb/s in order to appropriately mimic a specific operational network. Some cursory tests were performed with higher data rates with similar results in terms of the classification performance. However, a parallelized implementation of the pipeline will need to be explored in order for this system to be applicable to a more diverse set of real-world networks.

5. FUTURE WORK

Our future work will focus on scaling the performance of both the learning system and system architecture to provide a real-time machine learning NIDS capability at the 1 Gb/s scale. This work will require introducing more parallelism into the software architecture, and also to augment the existing learning approaches to scale to larger volumes of training data, yet still remain computationally tractable. We expect this work to culminate in the operational deployment of these sensors on enterprise-level government networks.

6. ACKNOWLEDGMENTS

Prepared by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285; managed by UT Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR2225. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

7. REFERENCES

[1] Dua, S. and Du, X. 2011. *Data Mining and Machine Learning in Cybersecurity*. Boca Raton, FL, Taylor and Francis Group, LLC.

[2] Sommer, R. and Paxson, V. 2010. Outside the closed world: on using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, IEEE.

[3] Zander, S., T. T. T. Nguyen, et al. 2005. Automated traffic classification and application identification using machine learning. In *IEEE Conference on Local Computer Networks 30th Anniversary (LCN '05)*, Sydney, Australia.

[4] Sharma, O., M. Girolami, et al. 2007. Detecting worm variants using machine learning. In *2007 ACM Conference on emerging Network EXperiments and Technologies (CoNEXT)*. New York, NY, U.S.A., ACM.

[5] Song, J., Takakura, H., et al. 2011. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns*. (2011) Salzburg, Austria.

[6] Symons, C. T. and Beaver, J. M. 2012. Nonparametric semi-supervised learning for network intrusion detection: combining performance improvements with realistic in-situ training. In *5th ACM Workshop on Artificial Intelligence and Security*. (2012). Raleigh, NC.

[7] Axelsson, S. 2000. The base-rate fallacy and the difficulty of intrusion detection. In *ACM Trans. Information and Sys. Security*. 3, 3 (Aug. 2000), 186 – 205. DOI=http://doi.acm.org/10.1145/357830.357849.

[8] Schapire, R. E. and Freund, Y. 2012. *Boosting: Foundations and Algorithms.* MIT Press.

[9] Schapire, R. E. and Singer, Y. 1999. Improved boosting algorithms Using confidence-related predictions. In *Machine Learning*. 37, 3 (1999), 297 – 336.

[10] Cohen, W. W. 2004. *MinorThird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data*. (2004). Available at: http://minorthird.sourceforge.net.

[11] Abe, N., Zadrozny, B. and Langford, J. 2006. Outlier detection by active learning. In *ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*. (2006). Philadelphia, PA.

[12] Rossey, L. M. et al. 2002. LARIAT: Lincoln adaptable real-time information assurance testbed. In *Proceedings of the IEEE Aerospace Conf.* (2002).

[13] Lippmann, R. P. et al. 2000. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. *Proc. the 2000 DARPA Information Survivability Conference and Exposition*. (2000)

[14] Kishimoto, K. et al. 2011. Improving performance of anomaly-based IDS by combining multiple classifiers. *Proc. IEEE/IPSJ Intl. Symposium on Applications and the Internet.* (2011), 366-371.