Network Traffic Prediction Models for Near- and Long-Term Predictions

Randall Wald, Taghi M. Khoshgoftaar, Richard Zuech, and Amri Napolitano
Florida Atlantic University, Boca Raton, FL 33431
Email: rwald1@fau.edu, khoshgof@fau.edu, rzuech@fau.edu, amrifau@gmail.com

***Abstract*—The large quantity of data flowing through network equipment demands that effective and efficient models be built to identify whether sessions are healthy or malicious. These models can be complex to build, and may rely on manually-labeled data. As a result, it is desirable to update or rebuild these models as rarely as possible without impairing classification performance. In this work, we consider the Kyoto dataset, training models on a single day's worth of data and testing these models under two circumstances: using 12 datasets gathered between six and twelve months after the training date, and using 9 datasets gathered between 18 and 19 months after the training date. In all cases, we apply three feature rankers (in addition to no feature ranking) and consider four classification models. We find that the results for the "near-term" 12 datasets are similar to those from the "long-term" 9 datasets, demonstrating that once a model has been built, it can potentially be used for over a year afterwards.**

***Keywords*-Network Flow, Intrusion Detection, Long-Term Prediction, Classification**

I. Introduction

Large computer networks make large targets for attackers, and therefore network security is an essential element of maintaining such networks. Although network security has many elements, a core component is monitoring sessions flowing across the network's border (to or from systems which are outside of the network) to determine if they are malicious. There are two well-known constraints on these rules: they must be effective, able to accurately detect malicious sessions while not stopping normal, "healthy" sessions, and they must be efficient, able to perform this analysis extremely quickly despite the vast amount of traffic which may be flowing over the network.

A third constraint which must also be considered when creating such rules is how often they must be re-trained. In general, the process of building a classification model is much more computationally intensive than the process of executing it: once the model's internal schema and parameters have been set, using it to assign labels to future instances is generally much simpler. In addition, acquiring labeled data can be very time-consuming: models which can approximate hand-labeling sessions as malicious or healthy must be training using hand-labeled data. Even with automated labeling, the process of inspecting each sessions to assign a label may be too time-consuming for real-time applications and may require fine-tuning which is not feasible on a regular basis. Thus, it is important to discover how effective models can be when built on one set of data and tested on data from much later, after the attack profiles have potentially changed.

In this paper, we perform such an investigation using data from the Kyoto 2006 dataset [10]. This collection includes network session information from a honeypot network collected on a daily basis for over 2.5 years (from the end of 2006 to the middle of 2009). This makes it unique compared to earlier network security datasets which do not allow for this type of long-range model training and testing. To understand how effective models can be when they are built on one day's worth of training data and tested on a separate day's worth of data, we built all of our models using the 2008-01-01 dataset, but consider two collections of training data: a "near-term" collection which contains two datasets per month from July to December of 2008, and the "far-term" collection which contains one dataset per week for the months of July and August, 2009. More frequent datasets were used from 2009 because the Kyoto dataset collection stops in August of 2009.

Using our training data from the beginning of January, 2008, we built classification models with four learning algorithms: 5-Nearest Neighbor, Naïve Bayes, and two forms of C4.5 Decision Trees. We also considered the use of feature selection, a family of techniques used to reduce the number of features in order to eliminate noise and unnecessary features. Although the Kyoto dataset only has 19 features (after eliminating those which directly predict the class label in order to be fair by not relying on these features), as noted earlier efficiency is a major goal of malicious session prediction approaches, so anything which can help reduce the complexity of the model should be considered, especially as feature selection can also help improve model effectiveness. We tested three forms of feature ranking as well as considering the "no feature selection" case as a baseline.

We found that generally speaking, models built from the 2008-01-01 training data were extremely effective both on the near-term and far-term datasets. Performance was slightly better on the near-term datasets (usually giving Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) values greater than 0.999) compared with the far-term datasets (which were only able to reliably exceed AUC values of 0.99), but in both cases the models had very high performance. Among the learners, 5-Nearest Neighbors usually gave the best performance, although this could depend on the choice of dataset and feature selection strategy: C4.5 (either form) typically came next, with Naïve Bayes giving the worst classification performance. As for the rankers, Signal-To-Noise showed the best performance overall, especially on the near-term datasets, while the Receiver Operating Characteristic ranker was second-best on the near-term datasets (but not on the far-term datasets, where everything other than Signal-To-Noise was tied for second place). Overall, these results show that models can be utilized over 1.5 years after they have been trained, and that feature selection is important not only to improve computational efficiency but to improve classification performance.

The remainder of this paper is organized as follows: Related works on the topic of feature selection for intrusion detection may be found in Section II. Details of the classification algorithms and feature selection strategies used, as well as our performance evaluation approaches, are provided in Section III. The Kyoto dataset itself is discussed in more detail in Section IV. Section V contains our results and discussion. Finally, Section VI presents our conclusions and ideas for future work.

II. Related Work

A number of machine learning and data mining approaches has been applied in IDSs [5], [6]. Machine learning techniques analyze the input data and find patterns in the known data which can then be used for making decisions on previously unknown (unseen) data. As Lappas et al. [4] explains, there are some specific aspects of data mining that can contribute to an intrusion detection project. Some of these aspects which are more related to detection of attacks are anomaly detection (which removes normal traffic from the network analyst's workflow to help improve the focus on abnormal traffic), classification (which predicts a specific record as being attack or normal), and clustering the data in natural categories. Data mining methods can be differentiated based on their modeling functions, whether they are supervised, unsupervised, or semi-supervised, and their choice of performance criteria and algorithms.

In [7] and [8], Novakov et al. proposed a hybrid anomaly detection method based on two popular statistical and spectral anomaly detection methods. Statistical anomaly detection methods consider the deviations from normal trends in data as anomalies. The normal trend of data is obtained through statistical analysis. Spectral analysis methods, also known as frequency analysis methods, analyse the time series to reveal anomalies. In their proposed hybrid method, Novakov et al. have chosen the popular PCA-subspace anomaly detection method as the statistical method and well-known Haar Wavelet analysis as the spectral method to build their hybrid approach. The PCA-subspace method is based on applying PCA to the data matrix. It detects the anomalies based on the correlation between data metrics (features). Basically, the feature space is transformed to the principal component space. This space is then divided to two normal and residual subspaces. By applying a threshold on the Euclidean magnitude squared of the transformed records to the residual sub-space, one can detect the anomalies. The Haar Wavelet actually measures the amount and magnitude of abrupt changes in the data trend which can lead to detection of anomalies. The proposed combination of these is called Hybrid PCA-Haar Wavelet analysis. First PCA is applied on a modified time shifted matrix of data. The instances are then projected to the residual subspace produced by applying PCA. In the second phase Haar Wavelet analysis is applied on the projected data to detect anomalies. This hybrid approach examines network traffic in time bins. Various aggregate features are extracted for the flows which fall into a specific time bins. They applied their method on the Kyoto 2006 dataset. Based on their results the hybrid approach is able to detect anomalies which neither the PCA nor Haar Wavelet approaches were able to detect.

Beaver et al. [1] propose a near real time machine learning system to provide network intrusion detection by discriminating normal traffic from malicious traffic. In their proposed system architecture a traffic acquisition part captures the network data and aggregates it into network flows. Then the analysis part of the system extracts features from network flows and trains a classifier based on observed network traffic. In their proposed system architecture any machine learning approach can be used as the analysis method. In their paper they have used AdaBoost as a supervised machine learning method, conducting their experiments on simulated data produced with the Lariat tool. The attacks are injected into normal traffic to compose their anomalous network traffic. They mention that their method is able to detect new zero-day attacks (attacks that have not been in the training phase), however there is no description on attacks and their types. To examine their machine learning approach on operational data they applied a semi-supervised variant of it on the Kyoto dataset [11]. The semi-supervised methods use both labeled and unlabeled data in the learning phase. They have shown the ability to generalize well based on a significantly smaller set of labeled samples. Using a semi-supervised approach makes it possible to learn the system in the environment where it is going to be developed (in situ learning). They used the Laplacian Eigenmaps as a Semi-Supervised method to network intrusion detection. Using a small number of labels with a semi-supervised method reduces the costs associated with learning in situ, making it a more viable approach.

In [9], Sallay et al. use data mining for a real-time adaptive intrusion detection alert classifier to discriminate correct alerts from false positives in a high speed network. Their approach is based on self-trained SVM with several learning scenarios. The experiments are done on the Kyoto 2006, ISCX 2012, and KDDCUP 99 datasets.

Kumar et al. [3] present a multi-objective evolutionary-based approach to generate a pool of noninferior individual solutions and ensemble solutions thereof. The final ensemble method can be used for intrusion detection. Here, the multi-objective genetic algorithm is used to select an optimal model which combines the objectives of attack detection rate, normal traffic detection rate, and diversity of evolved models. There are three phases for the proposed approach. Phase one evolves a set of individual Naive Bayesian solutions by selecting different feature subsets of the original feature space. The second phase generates (evolves) a set of ensembles based on the output models from phase one. In the third phase an integration of the base classifiers is performed to make the prediction of the final ensemble. The major issue in the proposed approach is that it takes a long time to compute fitness functions in various generations.

It is important to note that no previous work has considered the use of machine learning for building models which will be used for long-term prediction. While some IDSs are intended to run without modification for a long period of time, these do not use machine learning to generate their rules, and thus are not as tuned to the specific system they are monitoring. The present work considers models which were built using data from a specific system, and tested using data from the same system between 6 and 19 months later.

III. Methods

In this work we used four classification algorithms and three feature rankers (in addition to the "no feature selection" case) to build our models, along with the Area Under the ROC Curve metric to evaluate out models. These are all discussed in detail below.

A. Classification Algorithms

Four learners were chosen for our analysis: 5-Nearest Neighbor (5-NN), two forms of C4.5 Decision Trees (C4.5D and C4.5N), and Naïve Bayes (NB). These learners were all chosen due to their relative ease of computation and their dissimilarity from one another. Additional learners were explored during our preliminary investigation (Logistic Regression, Multi-Layer Perceptron, and Support Vector Machines), but these were found to take significantly more computational resources while also giving worse classification results. All models were built using the Weka machine learning toolkit [2], using default parameters unless changes (which will be discussed) were deemed appropriate based on previous research. For all classification models, the "malicious session" class was considered as the positive class, with the "normal, healthy session" class as the negative class.

The k-nearest neighbors, or k-NN, learner is an example of an instance based and lazy learning algorithm, which uses only the training data for building its hypothesis (without creating statistics or models from that data). The k-NN learner does this by calculating the distance from a given test sample to every training instance, and the predicted class is derived from the k nearest neighbors. Specifically, when there is a test sample that needs to be classified, the classes for each of the k closest training samples (for this paper, we chose $k = 5$, and thus the algorithm as a whole is called "5-NN") are found, and the weighted sums for each class are calculated (using $\frac{1}{distance}$ as the weighting term). The prediction will be the class with the largest cumulative weight [13].

C4.5 Decision Trees (implemented as J48 within Weka) are a form of tree-based learner which builds a decision tree where each node divides instances into two or more branches based on the values of one of their features. The C4.5 algorithm uses a normalized version of Information Gain to decide which feature is most useful for dividing the instances into groups which correspond to the class values. This feature is used for the root node. Following this, the algorithm iterates: a new node is created under each branch, finding the most-useful feature when applied to the remaining instances (those which would flow to this part of the tree). Finally, the algorithm halts when a stopping criterion is met, such as when each leaf node contains only instances from a single class. When using a decision tree to classify an unknown instance, its values are tested to determine its path through the tree, and the leaf node it finishes at will determine its class. In the present work, we employed both a version of C4.5 using the default parameter values from Weka (denoted"C4.5D") and a version where Laplace smoothing was turned on and tree pruning was turned off (denoted "C4.5N").

Naïve Bayes (NB) uses Bayes' Theorem to deduce the posterior probability that the instance is the member of a specific class. Ideally, the classifier would find the posterior probability directly and then assign each instance to the class for which it has the highest posterior. Unfortunately, it is very difficult to calculate the posterior directly. Therefore it is necessary to use Bayes' Theorem, which states that the posterior equals the ratio of the prior multiplied by the likelihood over the evidence. This process can be further simplified by certain assumptions. The evidence is always constant for the specific data set and therefore can be ignored for the purposes of classification. The likelihood formula, $p(F_1, \ldots, F_n|C)$ can be simplified to $\prod_i p(F_i|C)$ by making the naïve assumption that all of the features are conditionally independent from all of the other features. This naive assumption with the removal of the evidence parameter creates the Naïve Bayes classifier [13].

$$p(C|F_1, \ldots, F_n) = p(C) \prod_i p(F_i|C)$$

B. Feature Selection Techniques

Although feature selection encompasses a wide range of techniques, including feature ranking (which evaluates the quality of each feature individually and ranks the results), filter-based subset evaluation (which applies a statistical metric to various feature subsets to discover the best subset), and wrapper-based subset evaluation (which builds classification models with different feature subsets to discover which will produce the best model), in this work we consider three forms of feature ranking. These were chosen to represent three broad families of feature ranking technique: Chi-Squared (CS), representing commonly-used rankers often found in the machine learning literature; Receiver Operating Characteristic (ROC), representing threshold-based feature selection (TBFS); and Signal-To-Noise (S2N), representing First Order Statistics (FOS) based feature selection. CS utilizes the χ^2 statistic to measure the strength of the relationship between each independent variable and the class. ROC, as with other TBFS techniques, considers each feature individually with the class value and pretends that the feature's value (normalized to lie between 0 and 1) is a posterior probability prediction of the class value: the ROC value is calculated (as discussed in Section III-C) using this value. S2N, like all FOS rankers, uses first order statistics like mean and standard deviation to score the features; in particular, S2N finds the ratio of the difference between each class's mean value for a given feature divided by the sum of the standard deviations of the class's values.

Based on preliminary evaluation, we chose the top six features from each ranked list when building models with that feature selection approach. We also considered the "no feature selection" strategy, employing all features from the dataset for model-building.

C. *Performance Evaluation*

To evaluate the quality of our classification models, we used Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) as our metric. AUC builds a graph of the True Positive Rate vs. False Positive Rate as the classifier decision threshold is varied, and then uses the area under this graph as the performance across all decision thresholds. Note that this is the same metric as is used in the ROC ranker, but it differs in one important way: when used as a ranker, ROC operates solely on the normalized values of a single attribute, pretending that these are the output of a classifier. When AUC is used to evaluate our classification models, however, it is based on the actual output of those classification models. The different acronyms are used to highlight this distinction.

Because all models (including both feature selection and classification) were trained on the 2008-01-01 data prior to being tested on separate datasets, no cross-validation was necessary.

IV. Case Study

While a number of intrusion detection datasets exist, one major flaw with many such datasets is their artificial nature. The popular KDDCUP 99 dataset [12] has been used by many studies, but is entirely artificial and was generated over a decade ago. To resolve these problems, Song et al. [10] developed what they refer to as the Kyoto 2006 dataset. Unlike previous datasets, this is not composed solely of artificial data: although the "normal" traffic was generated artificially, this was done on honeypots, collections of real or virtual servers which appear to be valuable targets from the perspective of potential attackers. Song et al. employed a number of different types of honeypots, including a range of Windows, MacOS, and Solaris servers with different security settings and some embedded devices such as printers and TV sets. Once any malicious traffic was detected on a given honeypot, it was fully wiped and rebooted into its clean state, to give future attacks the same system profile. In addition to utilizing real (and not artificial) attack data, the Kyoto 2006 dataset is unique in containing a long duration of data: the honeypot systems were left exposed to the internet for almost three years (November 2006 through August 2009). For all captured sessions, 24 features were collected: 14 network session features similar to those used in the KDDCUP 99 dataset, and 10 features which included security analysis of each session and IP/port numbers for each session.

Date	Total # of Instances	# Normal	# Attack
2008-01-01	111,589	61,462	50,127
2008-07-10	92,371	72,863	19,508
2008-07-25	123,275	78,519	44,756
2008-08-10	124,393	64,759	59,634
2008-08-25	124,736	87,253	37,483
2008-09-10	114,301	75,749	38,552
2008-09-23	76,538	53,703	22,835
2008-10-10	98,705	79,298	19,407
2008-10-25	75,734	61,647	14,087
2008-11-10	101,462	78,875	22,587
2008-11-25	80,508	43,716	36,792
2008-12-10	125,187	66,459	58,728
2008-12-25	124,406	79,537	44,869
2009-07-01	125,198	65,054	60,144
2009-07-08	124,537	73,350	51,187
2009-07-15	125,688	70,065	55,623
2009-07-22	125,442	66,182	59,260
2009-07-29	124,279	68,446	55,833
2009-08-05	128,347	72,953	55,394
2009-08-11	127,173	67,823	59,350
2009-08-19	126,461	72,800	53,661
2009-08-26	125,937	75,801	50,136

TABLE I
Details of the Datasets

Due to the large quantity of data collected in the Kyoto 2006 dataset (over 93,076,270 sessions in total, 50,033,015 normal and 43,043,255 attack), in the present work we consider a reduced version of this data. All of our feature selection and model-building employs only the data from January 1st, 2008, which contains 111,589 instances total (61,462 normal, 50,127 malicious). Both "near-term" and "far-term" datasets were utilized to test these models: the near-term collection contains datasets from July to December of 2009, while the far-term collection contains datasets from July and August of 2009. Table I contains all of the dates we used, with the number of instances and number of normal/attack instances. In the near-term (2008) block, we used data from the 10th and 25th of each month, except for September 2008, as no data was available from September 24th through the end of the month. For the far-term (2009) data, each date is exactly one week after the previous date, except for the 8/11/2009 data. We chose 8/11/2009 as opposed to 8/12/2009 because the dataset for 8/12/2009 had an abnormally small number of instances (only 10,716), and thus we were concerned that this dataset had problems which might make it inappropriate for our analysis.

In addition, although the original data had 24 features, we excluded the security analysis of each session (because we wanted to test if our models could work effectively without this analysis), the Destination_IP_Address feature (as the honeypots which were targeted by outside attacks had different IP addresses from those which only received simulated healthy traffic, making this feature also unrealistic of real-world problems), and the class label (which was counted among the original 24), giving us 19 independent features.

V. Results

The results for models built using the dataset from 2008-01-01 are presented in Tables II and III, representing testing the models on the near-term (6–12 months later, from 2008-07-01 to 2008-12-25) and far-term (18–19 months later, from 2009-07-01 to 2009-08-26) data, respectively. In total 15 models were trained on the 2008-01-01 data, for different combinations of the four learners and four forms of feature selection (three ranking techniques and the "no feature selection" option for comparison), with the combination of CS feature ranking and C4.5D classification omitted for computational reasons. For all choices of test dataset and feature selection approach, the results from the best learner are printed in **bold** while those from the worst learner are printed in *italics*.

As can be seen from Table II, the results on the near-term data are extremely good: for all dates between 2008-07-01 and 2008-10-25, the model built using S2N and the top-performing learner (either 5-NN or C4.5N) will have an AUC performance value above 0.999, and combining the ROC ranker with the 5-NN learner will always give an AUC value above 0.99 for all 12 of these datasets. Generally speaking, the 5-NN learner performs best, especially towards the middle of the date range (where it performs best for three of the four feature selection strategies), but even elsewhere it will usually tie or exceed the next-best learner. Unfortunately, as 5-NN can be computationally intensive to process, it is not as feasible in practice: fortunately, a well-built C4.5N model may be applied more easily to large datasets, and these results are also very good, always having at least one ranker with an AUC value greater than 0.99 (and in fact, with the exception of the 2008-11-10 dataset's value of 0.98977, the combination of S2N and C4.5N always exceeds an AUC value of 0.99). Although the NB model is even simpler to perform on a large dataset, its results suffer more than any other learner: while the CS and S2N rankers often give good results with this learner, they still will frequently have AUC values below 0.99, sometimes even below 0.95. Whether or not this performance is acceptable when considering their ease of computation will depend on the specific hardware constraints of a particular practitioner. Finally, the C4.5D learner only is the best choice when used without any feature selection, and sometimes not even then. Because feature selection is shown to improve performance here, we therefore do not recommend the use of C4.5D.

Considering the feature selection approaches, we find that "no feature selection" is almost never the best choice, at least when considering the best learner for each feature selection strategy. However, it is also rarely the worst approach: more often, CS gives the worst performance (again, considering the best learner for each feature selection strategy). Between ROC and S2N, S2N will usually be the best ranker, although for certain datasets ROC is best. Overall, across all of the datasets, we would recommend the use of one of these two rankers (most likely S2N) along with a learner chosen based on the computational resources available: 5-NN if computation does not pose a meaningful restriction and maximal performance is desired, NB for extremely computationally-constrained environments, and C4.5N when neither of these constraints dominate the problem.

Test Dataset	Feature Selection	Learner			
		5-NN	C4.5D	C4.5N	NB
2008-07-10	No FS	0.99820	**0.99836**	0.99654	*0.98764*
	CS	**0.98956**	—	*0.98520*	0.98552
	ROC	0.99909	0.99863	**0.99914**	*0.98816*
	S2N	0.99947	0.99912	**0.99950**	*0.99649*
2008-07-25	No FS	0.99874	**0.99952**	0.99682	*0.98129*
	CS	**0.99926**	—	0.99583	*0.99233*
	ROC	**0.99939**	0.99638	0.99897	*0.96832*
	S2N	0.99960	0.99920	**0.99964**	*0.98927*
2008-08-10	No FS	0.99722	**0.99782**	0.99745	*0.98862*
	CS	*0.97955*	—	**0.99555**	0.99225
	ROC	**0.99928**	0.99739	0.99885	*0.98605*
	S2N	**0.99943**	0.99895	0.99938	*0.99645*
2008-08-25	No FS	0.99088	**0.99708**	0.99458	*0.98700*
	CS	**0.99893**	—	0.99660	*0.99452*
	ROC	0.99403	0.99762	**0.99913**	*0.98877*
	S2N	0.99962	0.99943	**0.99965**	*0.99692*
2008-09-10	No FS	**0.99508**	0.99187	0.99394	*0.98659*
	CS	0.99458	—	**0.99535**	*0.98608*
	ROC	0.99816	0.99709	**0.99898**	*0.98451*
	S2N	0.99807	0.99895	**0.99958**	*0.99757*
2008-09-23	No FS	**0.99809**	0.99252	0.98971	*0.97441*
	CS	*0.93165*	—	**0.98604**	0.97542
	ROC	**0.99881**	0.99554	0.99830	*0.97342*
	S2N	**0.99943**	*0.94104*	0.99645	0.99910
2008-10-10	No FS	**0.99715**	0.98938	0.98526	*0.97826*
	CS	**0.99316**	—	*0.97713*	0.98024
	ROC	**0.99922**	0.99200	0.99413	*0.98504*
	S2N	**0.99936**	*0.95132*	0.99501	0.99646
2008-10-25	No FS	0.99932	**0.99947**	0.99541	*0.98350*
	CS	**0.99042**	—	0.98990	*0.97289*
	ROC	**0.99969**	0.99292	0.99577	*0.97895*
	S2N	**0.99962**	0.99670	*0.99660*	0.99911
2008-11-10	No FS	**0.99469**	0.98841	*0.97940*	0.98238
	CS	*0.97970*	—	0.98477	**0.98543**
	ROC	**0.99687**	0.98847	0.99354	*0.98055*
	S2N	0.99578	*0.95695*	0.98977	**0.99670**
2008-11-25	No FS	0.98143	**0.99774**	0.99470	*0.94754*
	CS	**0.99127**	—	0.98993	*0.94808*
	ROC	**0.99432**	*0.87831*	0.99008	0.95099
	S2N	0.98762	0.99562	**0.99683**	*0.97578*
2008-12-10	No FS	0.99016	**0.99724**	0.99078	*0.97426*
	CS	**0.99750**	—	*0.98846*	0.98859
	ROC	**0.99252**	0.98390	0.98416	*0.96760*
	S2N	0.99257	*0.94983*	**0.99292**	0.98094
2008-12-25	No FS	0.96021	**0.99072**	0.98547	*0.88475*
	CS	*0.83107*	—	**0.99839**	0.96755
	ROC	**0.99121**	*0.73633*	0.95580	0.89229
	S2N	0.99633	0.99591	**0.99657**	*0.93357*

TABLE II
AUC Results for Near-Term (6–12 months) Test Datasets

Across the 12 datasets in Table II, there are few clear trends that vary with time. We do find that 5-NN is particularly effective towards the middle of this period (the 2008-09-23, 2008-10-10, and 2008-10-25 datasets), while C4.5N is more effective towards the beginning (before the area where 5-NN predominates) and very slightly towards the end. As for rankers, while S2N is the best ranker for the first seven of the 12 datasets, for the later results it is not always the best: for the 2008-10-25 and 2008-11-10 dates, ROC is better than S2N, and for the final three dates, either "no feature selection,"

Test Dataset	Feature Selection	Learner			
		5-NN	C4.5D	C4.5N	NB
2009-07-01	No FS	0.98416	**0.99517**	0.98998	*0.93560*
	CS	**0.99594**	—	0.98776	*0.93972*
	ROC	**0.98813**	0.97659	0.97500	*0.92236*
	S2N	*0.98759*	**0.99812**	0.99655	0.99263
2009-07-08	No FS	0.97484	*0.94669*	**0.98375**	0.96057
	CS	**0.99514**	—	*0.93943*	0.97557
	ROC	**0.99752**	0.99435	0.99600	*0.97577*
	S2N	0.99762	0.99862	*0.99687*	**0.99864**
2009-07-15	No FS	**0.98066**	*0.92207*	0.97940	0.95124
	CS	**0.99815**	—	*0.94032*	0.95684
	ROC	**0.99792**	0.99000	0.99032	*0.95876*
	S2N	0.99796	0.99631	*0.99208*	**0.99847**
2009-07-22	No FS	0.98984	**0.99158**	0.98621	*0.93282*
	CS	**0.99222**	—	0.97976	*0.94226*
	ROC	**0.99082**	0.98647	0.98659	*0.92770*
	S2N	0.98779	0.98910	**0.98974**	*0.97237*
2009-07-29	No FS	*0.87439*	**0.99909**	0.99869	0.93344
	CS	**0.99852**	—	0.99743	*0.94051*
	ROC	**0.92009**	*0.57019*	0.89387	0.90711
	S2N	*0.88255*	**0.99929**	0.99915	0.96439
2009-08-05	No FS	0.99175	**0.99900**	0.99748	*0.94751*
	CS	0.98548	—	**0.99390**	*0.93008*
	ROC	0.99146	0.99204	**0.99251**	*0.94158*
	S2N	0.99717	**0.99880**	0.99847	*0.99142*
2009-08-11	No FS	0.99296	**0.99853**	0.99616	*0.97056*
	CS	**0.99575**	—	0.99084	*0.96532*
	ROC	**0.99908**	0.99399	0.99698	*0.97089*
	S2N	**0.99903**	0.99837	*0.99775*	0.99797
2009-08-19	No FS	0.93471	**0.99885**	0.99764	*0.91773*
	CS	0.95607	—	**0.99632**	*0.89347*
	ROC	0.94256	0.91848	**0.96257**	*0.90635*
	S2N	0.99234	0.99823	**0.99854**	*0.96936*
2009-08-26	No FS	0.99623	**0.99836**	0.99587	*0.96964*
	CS	**0.99770**	—	0.99369	*0.96028*
	ROC	**0.99928**	0.99578	0.99656	*0.94544*
	S2N	**0.99932**	0.99790	0.99744	*0.97674*

TABLE III
AUC RESULTS FOR FAR-TERM (18–19 MONTHS) TEST DATASETS

CS, or both are better than S2N and ROC (although for these three, S2N is always better than ROC). Thus the reliability of S2N as the most effective feature selection strategy may only hold for earlier test datasets which are closer to the training dataset.

The results for the far-term test datasets (found in Table III) are relatively similar to those from the near-term test datasets: although performance is somewhat lower, it is still quite high overall. For each dataset, there is at least one combination of learner and feature selection approach which gives an AUC value greater than 0.99, and using the S2N ranker with the C4.5N learner exceeds this threshold in all but one dataset (for the 2009-07-22 dataset, it only has an AUC value of 0.98974). As with the earlier datasets, the results using the 5-NN learner tend to be as good or better than the other choices of learner, although the pattern here is less clear: rather than a clear temporal trend of one learner dominating for a time until another learner overtakes it, the optimal learner switches from one dataset to the next. In fact, C4.5D is sometimes the best learner when used with the S2N feature selection algorithm, which never occurred with the earlier datasets. Nonetheless, the pattern of 5-NN being the best learner, C4.5 (either D or N) being moderate, and NB being the worst remains; the relative computational constraints of these models also remain, so the optimal choice will depend on the specific application.

Similar to comparing the learners, the rankers show much the same pattern here as with the earlier datasets, albeit with a higher degree of random variation. S2N is still generally the best ranker overall, but is only the best ranker for five of the nine datasets, with "no ranking" being best for two datasets and CS and ROC each being best once. In all but one of these cases (when CS is best), however, S2N is second-best. The second-place ranker is also less clear on these far-term datasets compared with the near-term results: CS, ROC, and "no feature selection" are all second-place two times, in addition to the aforementioned three cases where S2N was second-place. CS is third-place (compared to fourth-place) more often than ROC, and likewise ROC is fourth-place more often than CS, but it is hard to make a fair comparison between these two rankers: beyond S2N giving the best results, there is no clear pattern among the remaining three feature selection strategies.

As the nine far-term datasets are much more closely packed with one another in terms of time, it is more difficult to find patterns that change over the course of this dataset: the optimal choices for learner and ranker show no broad patterns, and even the overall performance doesn't change significantly: there is a slight increase in performance towards the end of the time period, but this could be random fluctuation. If anything, the biggest difference between the near-term datasets and the far-term datasets is this very lack of pattern; however, because the second collection has roughly twice as many datasets per unit time as the first collection, it is hard to say if this pattern comes from genuine differences or merely examining minuet more carefully in the second, far-term collection. Nonetheless, for both collections the models built on the 2008-01-01 dataset perform quite well, especially given that new attack types were developed and performed in this dataset over the months and years following the 2008-01-01 dataset.

VI. CONCLUSION

In this work, we considered whether malicious session detection models trained on one set of data could be used on subsequent data, taken from either the "near-term" (6–12 months later) or the "far-term" (18–19 months later). To perform this analysis, we used data from the Kyoto 2006 dataset, built models with four different learners, and used three forms of feature selection in addition to no feature selection.

We discovered that our models were quite effective on both collections of data. Although the near-term test datasets gave slightly higher results than the far-term datasets (AUC values above 0.999, not just above 0.99), in both cases the performance was quite high. The learners showed a fairly consistent pattern across all test datasets, with the most time-consuming learner (5-NN) giving the best performance, and the least time-consuming learner (NB) giving the worst performance. The C4.5D and C4.5N learners fall in between these two extremes both in terms of classification performance and time complexity. As for the rankers, S2N was the best choice for both collections of datasets, but although ROC was second-best for the near-term data, on the far-term data all three approaches other than S2N tied for second-best. Based on these results, we would recommend the use of S2N, most likely with the C4.5N learner (which provides a good balance of performance and efficiency), when building malicious session detection models which should remain effective for over 1.5 years after being built.

Future experiments can consider even longer-term models (e.g., starting with training data earlier than 2008-01-01), as well as explore a wider range of feature selection approaches. Additional datasets may be considered, but only those which have at least two years of consistent daily network intrusion data from a single experimental source.

References

[1] J. M. Beaver, C. T. Symons, and R. E. Gillen, "A learning system for discriminating variants of malicious network traffic," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, ser. CSIIRW '13. New York, NY, USA: ACM, 2013, pp. 23:1–23:4. [Online]. Available: http://doi.acm.org/10.1145/2459976.2460003

[2] M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[3] G. Kumar and K. Kumar, "Design of an evolutionary approach for intrusion detection," *The Scientific World Journal*, vol. 2013, pp. 1–14, 2013. [Online]. Available: http://dx.doi.org/10.1155/2013/962185

[4] T. Lappas and K. Pelechrinis, "Data mining techniques for (network) intrusion detection systems," Department of Computer Science and Engineering UC Riverside, Riverside CA 92521, Tech. Rep., 2007. [Online]. Available: http://trac.assembla.com/odinIDS/export/12/Egio/artigos/datamining/dataIDS.pdf

[5] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real time data mining-based intrusion detection," in *Proceedings of DARPA Information Survivability Conference & Exposition II (DISCEX '01)*, vol. 1, 2001, pp. 89–100 vol.1.

[6] H. A. Nguyen and D. Choi, "Application of data mining to network intrusion detection: Classifier selection model," in *Challenges for Next Generation Network Operations and Service Management*, ser. Lecture Notes in Computer Science, Y. Ma, D. Choi, and S. Ata, Eds. Springer Berlin Heidelberg, 2008, vol. 5297, pp. 399–408. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88623-5_41

[7] S. Novakov, C.-H. Lung, I. Lambadaris, and N. Seddigh, "Combining statistical and spectral analysis techniques in network traffic anomaly detection," in *Next Generation Networks and Services (NGNS 2012)*, Dec 2012, pp. 94–101.

[8] ——, "Studies in applying pca and wavelet algorithms for network traffic anomaly detection," in *14th IEEE International Conference on High Performance Switching and Routing (HPSR 2013)*, July 2013, pp. 185–190.

[9] H. Sallay, A. Ammar, M. Ben Saad, and S. Bourouis, "A real time adaptive intrusion detection alert classifier for high speed networks," in *12th IEEE International Symposium on Network Computing and Applications (NCA 2013)*, August 2013, pp. 73–80.

[10] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "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 for Security*, ser. BADGERS '11. New York, NY, USA: ACM, 2011, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/1978672.1978676

[11] C. T. Symons and J. M. Beaver, "Nonparametric semi-supervised learning for network intrusion detection: Combining performance improvements with realistic in-situ training," in *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '12. New York, NY, USA: ACM, 2012, pp. 49–58. [Online]. Available: http://doi.acm.org/10.1145/2381896.2381905

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 53–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=1736481.1736489

[13] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical machine learning tools and techniques*, 3rd ed. Burlington, MA: Morgan Kaufmann, January 2011.