Proactive Defense for Evolving Cyber Threats

Richard Colbaugh
Sandia National Laboratories
New Mexico Institute of Mining and Technology
Albuquerque, NM USA
colbaugh@comcast.net

Kristin Glass
New Mexico Institute of Mining and Technology
Socorro, NM USA
kglass@icasa.nmt.edu

***Abstract*—There is significant interest to develop proactive approaches to cyber defense, in which future attack strategies are anticipated and these insights are incorporated into defense designs. This paper considers the problem of protecting computer networks against intrusions and other attacks, and leverages the coevolutionary relationship between attackers and defenders to derive two new methods for proactive network defense. The first method is a bipartite graph-based machine learning algorithm which enables information concerning previous attacks to be "transferred" for application against novel attacks, thereby substantially increasing the rate with which defense systems can successfully respond to new attacks. The second approach involves exploiting basic threat information (e.g., from cyber security analysts) to generate "synthetic" attack data for use in training defense systems, resulting in networks defenses that are effective against both current and (near) future attacks. The utility of the proposed methods is demonstrated by showing that they outperform standard techniques for the task of detecting malicious network activity in two publicly-available cyber datasets.**

***Keywords*—**cyber security, proactive defense, predictive analysis, machine learning, security informatics.

I. INTRODUCTION

Rapidly advancing technologies and evolving operational practices and requirements increasingly drive both private and public sector enterprises toward highly interconnected and technologically convergent information networks. Proprietary information processing solutions and stove-piped databases are giving way to unified, integrated systems, thereby dramatically increasing the potential impact of even a single well-planned network intrusion, data theft, or denial-of-service attack. It is therefore essential that commercial and government organizations develop network defenses which are able to respond rapidly to, or even foresee, new attack strategies and tactics.

Recognizing these trends and challenges, some cyber security researchers and practitioners are focusing their efforts on developing *proactive* methods of cyber defense, in which future attack strategies are anticipated and these insights are incorporated into defense designs [e.g., 1-5]. However, despite this attention, much remains to be done to place the objective of proactive defense on a rigorous and quantitative foundation. Fundamental issues associated with the dynamics and predictability of the coevolutionary "arms race" between attackers and defenders have yet to be resolved. For instance, although recent work has demonstrated that previous attacker actions and defender responses provide predictive information about future attacker behavior [3-5], not much is known about which measurables have predictive power or how to exploit these to form useful predictions. Moreover, even if these predictability and prediction issues were resolved, it is still an open question how to incorporate such predictive analytics into the design of practically-useful cyber defense systems.

This paper considers the problem of protecting enterprise-scale computer networks against intrusions and other attacks, and explicitly leverages the coevolutionary relationship between attackers and defenders to develop two new methods for proactive network defense. Each method formulates the task as one of behavior classification, in which innocent and malicious network activities are to be distinguished, and each assumes that only very limited prior information is available regarding exemplar attacks or attack attributes. The first method models the data as a bipartite graph of *instances* of network activities and the *features* or attributes that characterize these instances. The bipartite graph data model is used to derive a machine learning algorithm which accurately classifies a given instance as either innocent or malicious based upon its behavioral features. The algorithm enables information concerning previous attacks to be "transferred" for use against novel attacks; crucially, it is assumed that previous attacks are drawn from a distribution of attack instances which is related *but not identical* to that associated with the new malicious behaviors. This transfer learning algorithm provides a simple, effective way to extrapolate attacker behavior into the future, and thus significantly increases the rate with which defense systems can successfully respond to new attacks.

The second approach to proactive network defense proposed in this paper represents attacker-defender coevolution as a hybrid dynamical system (HDS) [6,7], with the HDS discrete system modeling the "modes" of attack (e.g., a particular class of DoS or data exfiltration procedures) and the HDS continuous system generating particular attack instances corresponding to the attack mode presently "active". Our algorithm takes as input the mode of attack, obtained for example from the insights of cyber analysts, and generates synthetic attack data for this mode of malicious activity; these data are then combined with actually observed attacks to train a learning-based classifier to be effective against both current and (near) future attacks. The utility of the proposed methods is demonstrated by showing that they outperform standard techniques for the task of distinguishing innocent and malicious network behaviors in analyses of two publicly-available cyber datasets.

978-1-61284-4577-0085-9/11/$26.00 ©2011 IEEE

II. PRELIMINARIES

We approach the task of protecting computer networks from attack as a classification problem, in which the objective is to distinguish innocent and malicious network activity. Each instance of network activity is represented as a feature vector $x \in \Re^{|F|}$, where entry x_i of x is the value of feature i for instance x and F is the set of instance features or attributes of interest (x may be normalized in various ways [7]). Instances can belong to one of two classes: positive / innocent and negative / malicious; generalizing to more than two classes is straightforward. We wish to learn a vector $c \in \Re^{|F|}$ such that the classifier orient = sign($c^T x$) accurately estimates the class label of behavior x, returning +1 (−1) for innocent (malicious) activity.

Knowledge-based classifiers leverage prior domain information to construct the vector c. One way to obtain such a classifier is to assemble a "lexicon" of positive / innocent features $F^+ \subseteq F$ and malicious / negative features $F^- \subseteq F$, and to set $c_i = +1$ if feature i belongs to F^+, $c_i = -1$ if i is in F^-, and $c_i = 0$ otherwise; this classifier simply sums the positive and negative feature values in the instance and assigns instance class accordingly. Unfortunately this sort of scheme is unable to improve its performance or adapt to new domains, and consequently is usually not very useful in cyber security applications.

Alternatively, learning-based methods attempt to generate the classifier vector c from examples of positive and negative network activity. To obtain a learning-based classifier, one can begin by assembling a set of n_l *labeled* instances $\{(x_i, d_i)\}$, where $d_i \in \{+1, -1\}$ is the class label for instance i. The vector c is then learned through training with the set $\{(x_i, d_i)\}$, for example by solving the following set of equations for c:

$$[X^T X + \gamma I_{|F|}]\, c = X^T d, \tag{1}$$

where matrix $X \in \Re^{nl \times |F|}$ has instance feature vectors for rows, $d \in \Re^{nl}$ is the vector of instance labels, $I_{|F|}$ denotes the $|F| \times |F|$ identity matrix, and $\gamma \geq 0$ is a constant; this corresponds to regularized least squares (RLS) learning [8]. Many other learning strategies can be used to compute c [8]. Learning-based classifiers have the potential to improve their performance and adapt to new situations, but realizing these capabilities typically requires that large training sets of labeled attacks be obtained. This latter characteristic represents a significant drawback for cyber security applications, where it is desirable to be able to recognize new attacks given only a few (or no) examples.

In what follows we present two new learning-based approaches to cyber defense which are able to perform well with only very modest levels of prior knowledge regarding the attack classes of interest. The basic idea is to leverage "auxiliary" information which is readily available in cyber security applications. More specifically, the first proposed method is a *transfer learning* algorithm [e.g., 9] which permits the knowledge present in data on previous attacks to be transferred for implementation against new attacks. The second approach uses prior knowledge concerning attack "modes" to generate synthetic attack data for use in training defense systems, resulting in networks defenses which are effective against both current and (near) future attacks.

III. METHOD ONE: TRANSFER LEARNING

In this section we first derive a bipartite graph-based transfer learning algorithm for distinguishing innocent and malicious network behaviors, and then demonstrate the algorithm's effectiveness through a case study using publicly-available network intrusion data obtained from the KDD Cup archive [10]. The basic hypothesis is simple and natural: because attacker / defender behavior coevolves, previous activity should provide some indication of future behavior, and transfer learning is one way to quantify and operationalizes this intuition.

A. Proposed Algorithm

The development of the proposed algorithm begins by modeling the problem data as a bipartite graph G_b, in which instances of network activity are connected to their features (see Figure 1). It is easy to see that the adjacency matrix A for graph G_b is given by

$$A = \begin{bmatrix} 0 & X \\ X^T & 0 \end{bmatrix} \tag{2}$$

where matrix $X \in \Re^{n \times |F|}$ is constructed by stacking the n instance feature vectors as rows, and each '0' is a matrix of zeros. In the proposed algorithm, integration of labeled and "auxiliary" data is accomplished by exploiting the relationships between instances and features encoded in the bipartite graph model. The basic idea is to assume that, in G_b, positive / negative instances will tend to be connected to positive / negative features. Note that, as shown below, the learning algorithm can incorporate a lexicon of labeled features (if available). It is assumed that this lexicon is used to build vector $w \in \Re^{|F|}$, where the entries of w are set to +1 (innocent), −1 (malicious), or 0 (unknown) according to the polarity of the corresponding features.

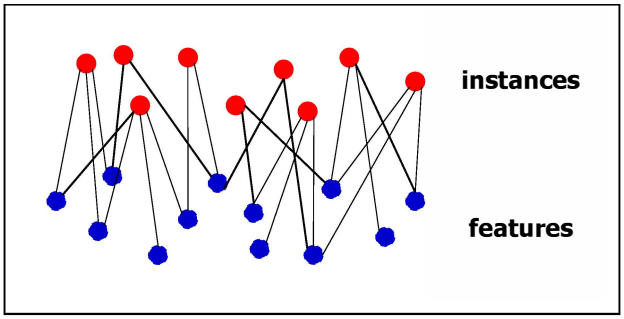

Figure 1. Cartoon of bipartite graph model G_b, in which the instances of network activity (red vertices) are connected to features (blue vertices) they contain, and link weights (black edges) reflect the magnitudes taken by the features in the associated instances.

Many cyber security applications are characterized by the presence of limited labeled data for the attack class of interest but ample labeled information for a related class of malicious activity. For example, an analyst may be interested in detecting a new class of attacks, and may have in hand a large set of la-

beled examples of normal network behavior as well as attacks which have been experienced in the recent past. In this setting it is natural to adopt a transfer learning approach, in which knowledge concerning previously observed instances of innocent / malicious behavior, the so-called *source* data, is transferred to permit classification of new *target* data. In what follows we present a new bipartite graph-based approach to transfer learning that is well-suited to cyber defense applications.

Assume that the initial problem data consists of a collection of $n = n_T + n_S$ network events, where n_T is the (small) number of labeled instances available for the target domain, that is, examples of network activity of current interest, and $n_S >> n_T$ is the number of labeled instances from some related source domain, say reflecting recent activity; suppose also that a modest lexicon F_l of labeled features is known (this set can be empty). Let this label data be used to encode vectors $d_T \in \Re^{nT}$, $d_S \in \Re^{nS}$, and $w \in \Re^{|F|}$, respectively. Denote by $d_{T,est} \in \Re^{nT}$, $d_{S,est} \in \Re^{nS}$, and $c \in \Re^{|F|}$ the vectors of estimated class labels for the target and source instances and the features, and define the *augmented classifier* $c_{aug} = [d_{S,est}^T \;\; d_{T,est}^T \;\; c^T]^T \in \Re^{n+|F|}$. Note that the quantity c_{aug} is introduced for notational convenience in the subsequent development and is not directly employed for classification.

We derive an algorithm for learning c_{aug}, and therefore c, by solving an optimization problem involving the labeled source and target training data, and then use c to estimate the class label of any new instance of network activity via the simple linear classifier orient = sign($c^T x$). This classifier is referred to as *transfer learning-based* because c is learned, in part, by transferring knowledge about the way innocent and malicious network behavior is manifested in a domain which is related to (but need not be identical to) the domain of interest.

We wish to learn an augmented classifier c_{aug} with the following four properties: 1.) if a source instance is labeled, then the corresponding entry of $d_{S,est}$ should be close to this ±1 label; 2.) if a target instance is labeled, then the corresponding entry of $d_{T,est}$ should be close to this ±1 label, and the information encoded in d_T should be emphasized relative to that in the source labels d_S; 3.) if a feature is in the lexicon F_l, then the corresponding entry of c should be close to this ±1 label; and 4.) if there is an edge X_{ij} of G_b which connects an instance i and a feature j, and X_{ij} possesses significant weight, then the estimated class labels for i and j should be similar.

The four objectives listed above may be realized by solving the following minimization problem:

$$\min_{c_{aug}} \; c_{aug}^T L c_{aug} + \beta_1 \left\| d_{S,est} - k_S d_S \right\|^2 + \beta_2 \left\| d_{T,est} - k_T d_T \right\|^2 + \beta_3 \left\| c - w \right\|^2 \qquad (3)$$

where $L = D - A$ is the graph Laplacian matrix for G_b, with D the diagonal degree matrix for A (i.e., $D_{ii} = \Sigma_j A_{ij}$), and β_1, β_2, β_3, k_S, and k_T are nonnegative constants. Minimizing (3) enforces the four properties we seek for c_{aug}. More specifically, the second, third, and fourth terms penalize "errors" in the first three properties, and choosing $\beta_2 > \beta_1$ and $k_T > k_S$ favors target label data over source labels. To see that the first term enforces the fourth property, note that this expression is a sum of components of the form $X_{ij} (d_{T,est,i} - c_j)^2$ and $X_{ij} (d_{S,est,i} - c_j)^2$. The constants β_1, β_2, β_3 can be used to balance the relative importance of the four properties.

The c_{aug} which minimizes the objective function (3) can be obtained by solving the following set of linear equations:

$$\begin{bmatrix} L_{11} + \beta_1 I_{nS} & L_{12} & L_{13} \\ L_{21} & L_{22} + \beta_2 I_{nT} & L_{23} \\ L_{31} & L_{32} & L_{33} + \beta_3 I_{|F|} \end{bmatrix} c_{aug} = \begin{bmatrix} \beta_1 k_S d_S \\ \beta_2 k_T d_T \\ \beta_3 w \end{bmatrix} \qquad (4)$$

where the L_{ij} are matrix blocks of L of appropriate dimension. The system (4) is sparse because the data matrix X is sparse, and therefore large-scale problems can be solved efficiently. Note that in situations where the set of available labeled instances and features is *very* limited, classifier performance can be improved by replacing L in (4) with the normalized Laplacian $L_n = D^{-1/2} L D^{-1/2}$, or with a power of this matrix L_n^k (for k a positive integer).

We summarize the above discussion by sketching an algorithm for constructing the proposed transfer learning classifier:

Algorithm TL (Transfer Learning):

1. Assemble the set of equations (4), possibly by replacing the graph Laplacian L with L_n^k.
2. Solve equations (4) for $c_{aug} = [d_{S,est}^T \;\; d_{T,est}^T \;\; c^T]^T$ (for instance using the Conjugate Gradient method).
3. Estimate the class label (innocent or malicious) of any new activity x of interest as: orient = sign($c^T x$).

B. Algorithm Evaluation

We now examine the performance of Algorithm TL for the problem of distinguishing innocent and malicious network activity in the KDD Cup 99 dataset, a publicly-available collection of network data consisting of both normal activities and attacks of various kinds [10]. For this study we randomly selected 1000 Normal connections (N), 1000 denial-of-service attacks (DoS), and 1000 unauthorized remote-access events (R2L) to serve as our test data. Additionally, small sets of each of these classes of activity were chosen at random from [10] to be used for training Algorithm TL, and a lexicon of four features, two positive and two negative, was constructed manually and employed to form the lexicon vector w.

We defined two tasks with which to explore the utility of Algorithm TL. In the first, the goal is to distinguish N and DoS instances, and it is assumed that the following data is available to train Algorithm TL: 1.) a set of $d_S/2$ labeled N and $d_S/2$ labeled R2L instances (source data), 2.) a set of $d_T/2$ labeled N and $d_T/2$ labeled DoS instances (target data), and 3.) the four lexicon features. Thus the source domain consists of N and R2L activities and the target domain is composed of N and DoS instances. In the second task the situation is reversed – the objective is to distinguish N and R2L activities, the source domain is made up of d_S (total) labeled N and DoS instances, and

the target domain consists of d_T (total) N and R2L instances. In all tests the number of labeled source instances is $d_S = 50$, while the number of target instances d_T is varied to explore the way classifier performance depends on this key parameter. Of particular interest is determining if it is possible to obtain good performance with only limited target data, as this outcome would suggest both that useful information concerning a given attack class is present in other attacks *and* that Algorithm TL is able to extract this information.

This study compared the classification accuracy of Algorithm TL with that of a well-tuned version of the RLS algorithm (1) and a standard naïve Bayes (NB) algorithm [11]; as the performance of the RLS and NB methods were quite similar, we report only the RLS results. Algorithm TL is implemented with the following parameter values: $\beta_1 = 1.0$, $\beta_2 = 3.0$, $\beta_3 = 5.0$, $k_S = 0.5$, $k_T = 1.0$, and $k = 5$. We examined training sets which incorporated the following numbers of target instances: $n_T = 2, 5, 10, 20, 30, 40, 50, 60$. As in previous studies (see, for example, [10]), only the 34 "continuous features" were used for learning the classifiers.

Sample results from this study are depicted in Figure 2. Each data point in the plots represents the average of 100 trials. It can be seen that Algorithm TL outperforms the RLS classifier (and also the standard NB algorithm), and that the difference in accuracy of the methods increases substantially as the volume of training data from the target domain becomes small. The performance of Algorithm TL for this task is also superior to that reported for other learning methods tested on these data [e.g., 12]. The ability of Algorithm TL to accurately identify a novel attack after seeing only a very few examples of it, which is a direct consequence of its ability to transfer useful knowledge from related data, is expected to be of considerable value for a range of cyber security applications.

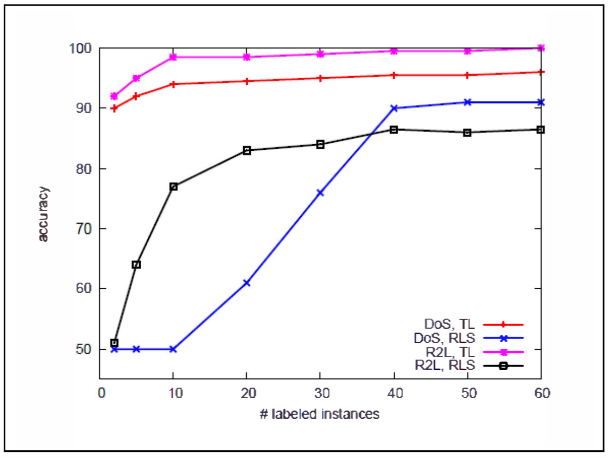

Figure 2. Performance of Algorithm TL with limited labeled data. The plot shows how classifier accuracy (vertical axis) varies with number of available labeled target instances (horizontal axis) for four tasks: distinguish N and DoS using RLS classifier (blue), distinguish N and DoS using Algorithm TL (red), distinguish N and R2L using RLS classifier (black), and distinguish N and R2L using Algorithm TL (magenta),

Finally, it is interesting to observe that the bipartite graph formulation of Algorithm TL permits useful information to be extracted from network data *even if no labeled instances are available*. More specifically, we repeated the above study for the case in which $d_T = d_S = 0$, that is, when no labeled instances are available in either the target or source domains. The knowledge reflected in the lexicon vector w is still made available to Algorithm TL. As shown in Figure 3, employing a "lexicon only" classifier, as described in Section II, yields classification accuracy which is not much better than the 50% baseline achievable with random guessing. However, using this lexicon information together with Algorithm TL enables useful classification accuracy to be obtained (see Figure 3). This somewhat surprising result can be explained as follows: the "clustering" property of Algorithm TL encoded in objective function (3) allows the domain knowledge in the lexicon to leverage latent information present in the *unlabeled* target and source instances, thereby boosting classifier accuracy.

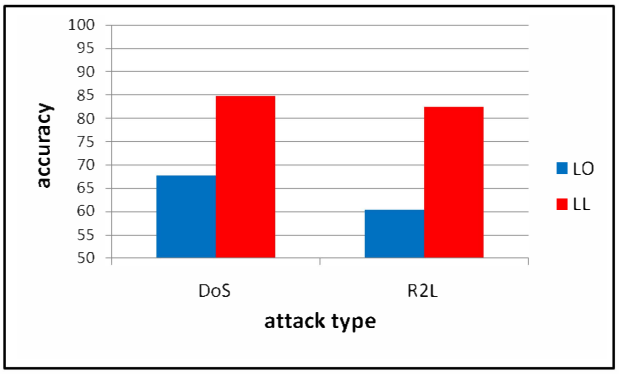

Figure 3. Performance of Algorithm TL with no labeled instance data. The bar graphs depicts classifier accuracy for four tasks: distinguish N and DoS using a lexicon-only (LO) classifier (left, blue bar), distinguish N and DoS using lexicon-learning (LL) via Algorithm TL (left, red bar), distinguish N and R2L using an LO classifier (right, blue bar), and distinguish N and R2L using LL via Algorithm TL (right, red bar).

IV. Method Two: Synthetic Attack Generation

In this section we derive our second algorithm for distinguishing normal and malicious network activity and demonstrate its effectiveness through a case study using the publicly-available Ling-Spam dataset [13]. Again the intuition is that attacker / defender coevolution should make previous activity somewhat indicative of future behavior, and in the present case we exploit this notion by generating "predicted" attack data and using this synthetic data for classifier training.

A. Proposed Algorithm

The development of the second approach to proactive defense begins by modeling attacker / defender interaction as a stochastic hybrid dynamical system (S-HDS). Here we present a brief, intuitive overview of the basic idea; a comprehensive description of the modeling procedure is detailed in [7]. An S-HDS (see Figure 4) is a feedback interconnection of a discrete-state stochastic process, such as a Markov chain, with a family

of continuous-state stochastic dynamical systems [6,14]. Combining discrete and continuous dynamics within a unified, computationally tractable framework offers an expressive, scalable modeling environment that is amenable to formal mathematical analysis. In particular, S-HDS models can be used to efficiently represent dynamical phenomena which evolve on a broad range of time scales, a property of considerable value in the present application [14].

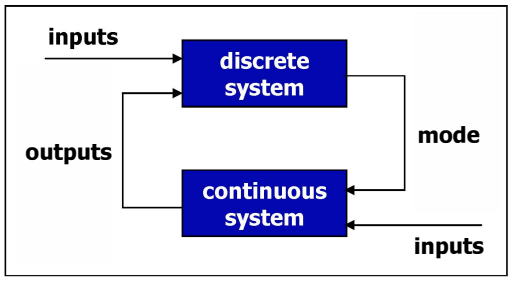

Figure 4. Schematic of basic S-HDS feedback structure. The discrete and continuous systems in this framework model the selection of attack "mode" and resulting adversary behavior, respectively, which arise from the coevolving attacker-defender dynamics.

As a simple illustration of the way the S-HDS formalism enables effective, efficient mathematical representation of cyber phenomena, consider the task of modeling the coevolution of Spam attack methods and Spam filters. At an abstract but still useful level, one can think of Spam-Spam filter dynamics as evolving on two timescales:

- the *slow timescale*, which captures the evolution of attack strategies; as an example, consider the way early Spam filters learned to detect Spam by identifying words that were consistently associated with Spam, and how Spammers responded by systematically modifying the wording of their messages, for instance via "add-word" (AW) and "synonym" attacks [15];
- the *fast timescale*, which corresponds to the generation of particular attack instances for a given "mode" of attack (for example, the synthesis of Spam messages according to a specific AW attack method).

We show in [7] that a range of adversarial behavior can be represented within the S-HDS framework, and derive simple but reasonable models for Spam-Spam filter dynamics and for basic classes of network intrusion attacks.

In [14] we develop a mathematically-rigorous procedure for predictive analysis for general classes of S-HDS. Among other capabilities, this analytic methodology enables the predictability of a given dynamics to be assessed and the predictive measurables (if any) to be identified. Applying this predictability assessment process to the adversarial S-HDS models constructed in [7] reveals that, for many such systems, the most predictive measurable is the *mode* of attack, that is, the state variable for the discrete system component of the S-HDS (see [7] for a detailed description of this analysis). Observe that this result is intuitively sensible.

This analytic finding suggests the following *synthetic data learning* (SDL) approach to proactive defense. First, identify the mode(s) of attack of interest. For attacks which are already underway, [7] offers an S-HDS discrete-system state estimation method that allows the mode to inferred using only modest amounts of measured data. Alternatively, and of more interest in the present application, it is often possible to identify likely future attack modes through analysis of auxiliary information sources (e.g., the subject matter knowledge possessed by domain experts or "non-cyber" data such as that found in social media [16,17]).

Once a candidate attack mode has been identified, synthetic attack data corresponding to the mode can be generated by employing one of the S-HDS models derived in [7]. The synthetic data take the form of a set of K network attack instance vectors, denoted $A_S = \{x_{S1}, \ldots, x_{SK}\}$. The set A_S can then be combined with (actual) measurements of L normal network activity instances, $N_M = \{x_{NM1}, \ldots, x_{NML}\}$, and P (recently) observed attacks, $A_M = \{x_{M1}, \ldots, x_{MP}\}$, yielding the training dataset $TR = N_M \cup A_M \cup A_S$ of real and synthetic data. It is hypothesized that training classifiers with the augmented set TR may offer a mechanism for deriving defenses which are effective against both current and near future malicious activity.

We summarize the above discussion by sketching a procedure for constructing the new SDL classifier:

Algorithm SDL:

1. Identify the mode(s) of attack of interest (e.g., via domain experts or auxiliary data).
2. Generate a set of synthetic attack instances A_S corresponding to the attack mode identified in Step 1.
3. Assemble sets of normal network activity N and measured attack activity A_M for the network under study.
4. Train a classifier (e.g., RLS, NB) using the training data $TR = N_M \cup A_M \cup A_S$. Estimate the class label (innocent or malicious) of any network activity x with the formula: orient(x) = sign($c^T x$).

B. Algorithm Evaluation

We now examine the performance of Algorithm SDL for the problem of distinguishing legitimate and Spam emails in the Ling-Spam dataset [13], a corpus of 2412 non-Spam emails collected from a linguistics mailing list and 481 Spam emails received by the list. After data cleaning and random subsampling of the non-Spam messages we are left with 468 Spam and 526 non-Spam messages for training and testing purposes; this set of 994 emails will be referred to as the *nominal Spam* corpus. (Note that all email was preprocessed using the *ifile* tool [18].)

We considered three scenarios in this study:

1. NB classifier / nominal Spam: for each of ten runs, the nominal Spam corpus was randomly divided into equal-sized training and testing sets and the class label for each message in the test set was estimated with a trained naïve Bayes (NB) algorithm [11];

2. NB classifier / nominal plus attack Spam: for each of ten runs, the nominal Spam corpus was randomly divided into equal-sized training and testing sets and the test set was then augmented with 263 additional non-Spam messages (taken from the Ling-Spam dataset) and 234 Spam messages generated via a standard add-word (AW) attack methodology [15]; the class labels for the test messages were estimated with the NB algorithm [11] trained on the nominal Spam training set;
3. Algorithm SDL / nominal plus attack Spam: for each of ten runs, the training and test corpora were constructed exactly as in Scenario 2 and the class labels for the test messages were estimated with Algorithm SDL.

In generating the AW attacks in Scenarios 2. and 3., we assume that the attacker knows to construct AW Spam to defeat an NB filter but does not have knowledge of the specific filter involved [15]. Analogously, the synthetic AW attacks generated in Scenario 3 (using Step 2 of Algorithm SDL) are computed with no knowledge of the attacker's methodology beyond the mode of attack (i.e., AW).

NB Algorithm: Nominal Spam

class\truth	non-Spam	Spam
non-Spam	262	19
Spam	1	215

NB Algorithm: Nominal and Attack Spam

class\truth	non-Spam	Spam
non-Spam	524	253
Spam	2	215

Algorithm SDL: Nominal and Attack Spam

class\truth	non-Spam	Spam
non-Spam	524	40
Spam	2	428

Figure 5. Performance of Algorithm SDL on Spam dataset. Each confusion matrix shows number of non-Spam messages classified as non-Spam and Spam (left column) and number of Spam messages classified as non-Spam and Spam (right column). The three matrices, from top to bottom, report the results for: NB against nominal Spam, NB against Spam which contains add-word attacks, and Algorithm SDL against Spam which contains add-word attacks.

Sample results from this study are displayed in Figure 5. In each case the "confusion matrix" [8] reports the (rounded) average performance over the ten runs. It can be seen that, as expected, the NB filter does well against the nominal Spam but poorly against the AW Spam (in fact, the NB filter does not detect a single instance of AW Spam). In contrast, Algorithm SDL performs well against both nominal Spam and AW Spam, achieving ~96% classification accuracy with a low false positive rate. It is emphasized that this result is obtained using only the (synthetic) estimate of AW Spam generated in Step 2 of Algorithm SDL.

ACKNOWLEDGEMENTS

This work was supported by the Laboratory Directed Research and Development Program at Sandia National Laboratories. We thank Chip Willard of the U.S. Department of Defense for numerous helpful discussions on aspects of this research.

REFERENCES

[1] Byers, S. and S. Yang, "Real-time fusion and projection of network intrusion activity", *Proc. ISIF/IEEE Intern. Conference on Information Fusion*, Cologne, Germany, July 2008.

[2] Armstrong, R., J. Mayo, and F. Siebenlist, "Complexity science challenges in cybersecurity", Sandia National Laboratories SAND Report, March 2009.

[3] Colbaugh, R., "Does coevolution in malware adaptation enable predictive analysis?", *IFA Workshop: Exploring Malware Adaptation Patterns*, San Francisco, CA, May 2010.

[4] Mashevsky, Y., Y. Namestnikov, N. Denishchenko, and P. Zelensky, "Method and system for detection and prediction of computer virus-related epidemics", US Patent 7,743,419, June 2010.

[5] Bozorgi, M., L. Saul, S. Savage, and G. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits", *Proc. ACM SIGKDD Conference*, Washington DC, July 2010.

[6] Majumdar, R. and P. Tabuada, *Hybrid Systems: Computation and Control*, LNCS 5469, Springer, Berlin, 2009.

[7] Colbaugh, R. and K. Glass, "Proactive defense for evolving cyber threats", Sandia National Laboratories SAND Report, March 2011.

[8] Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Second Edition, Springer, New York, 2009.

[9] Pan, S. and Q. Yang, "A survey on transfer learning", *IEEE Trans. Knowledge and Data Engineering*, Vol. 22, pp. 1345-1359, 2010.

[10] http://kdd.ics.uci.edu/databases/kddcup99/; accessed Dec. 2010.

[11] http://www.borgelt.net/bayes.html; accessed July 2010.

[12] He, J., Y. Liu, and R. Lawrence, "Graph-based transfer learning", *Proc. 18th ACM Conference on Information and Knowledge Management*, Hong Kong, November 2009.

[13] http://labs-repos.iit.demokritos.gr/skel/i-config/downloads/; accessed July 2010.

[14] Colbaugh, R. and K. Glass, "Predictive analysis for dynamical processes I: Multi-scale hybrid system modeling, and II: Predictability and warning analysis", *Proc. 2009 IEEE Intern. Multi-Conference on Systems and Control*, Saint Petersburg, Russia, July 2009.

[15] Lowd, D. and C. Meeks, "Good word attacks on statistical Spam filters", *Proc. Second Conference on Email and Anti-Spam*, Palo Alto, CA, July 2005.

[16] Cao, L., P. Yu, C. Zhang, H. Zhang, F. Tsai, and K. Chan, "Blog data mining for cyber security threats", *Data Mining for Business Applications*, Springer US, 2009.

[17] Colbaugh, R. and K. Glass, "Emerging topic detection for business intelligence via predictive analysis of 'meme' dynamics", *Proc. AAAI 2011 Spring Symposium*, Palo Alto, CA, March 2011.

[18] http://www.nongnu.org/ifile/; accessed July 2010.