SAND2016-3373C

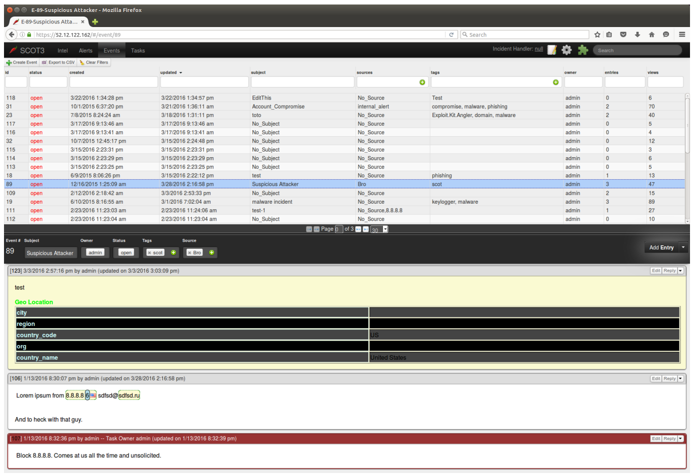
Learning to Rank for Alert Triage

Michael Bierma, Justin E. Doak (JD), and Corey Hudson
*Sandia National Laboratories
Livermore, California
mbierma,jedoak,cmhudso@sandia.gov

***Abstract*—As cyber monitoring capabilities expand and data rates increase, cyber security analysts must filter through an increasing number of alerts in order to identify potential intrusions on the network. This process is often manual and time-consuming, which limits the number of alerts an analyst can process. This generation of a vast number of alerts without any kind of ranking or prioritization is often referred to as alert desensitization [1]. This is the phenomenon where competent analysts become so numbed by the barrage of false positives that they are unable to identify the true positives, leading to unfortunate breaches. Our goal is to alleviate alert desensitization by placing the most important alerts at the front of the queue. With less time and energy expended investigating false positives, critical alerts may not be overlooked allowing timely responses to potential breaches. This paper discusses the use of supervised machine learning to rank these cyber security alerts to ensure that an analyst's time and energy are focused on the most important alerts.**

I. Introduction

The effort to use supervised machine learning to rank cyber security alerts began as an laboratory-directed research and development (LDRD) project at Sandia National Laboratories (SNL) named Active Learning for Alert Triage (ALAT). There was roughly a six-month period where no work was performed on ALAT because the LDRD project had ended and a customer to fund a follow-on project had not been identified. Thus, one of our early goals was to revitalize the codebase, including an extensive refactoring and also the addition of numerous unit tests. We considered this essential to facilitate the addition of novel functionality and to



id	status	created	updated	subject	sources	tags	owner	entries	views
118	open	3/22/2016 1:34:28 pm	3/22/2016 1:34:57 pm	EditThis	No_Source	Test	admin	0	6
31	open	10/1/2015 6:37:20 pm	3/21/2016 1:36:11 am	Account_Compromise	internal_alert	compromise, malware, phishing	admin	2	70
23	open	7/8/2015 8:24:24 am	3/18/2016 1:31:11 pm	toto	No_Source	Exploit.Kit.Angler, domain, malware	admin	2	40
117	open	3/17/2016 9:13:46 am	3/17/2016 9:13:46 am	No_Subject	No_Source		admin	0	5
116	open	3/17/2016 9:13:41 am	3/17/2016 9:13:41 am	No_Subject	No_Source		admin	0	4
32	open	10/7/2015 12:45:17 pm	3/15/2016 2:24:48 pm	No_Subject	No_Source		admin	0	12
115	open	3/15/2016 2:23:31 pm	3/15/2016 2:23:31 pm	No_Subject	No_Source		admin	0	3
114	open	3/15/2016 2:23:29 pm	3/15/2016 2:23:29 pm	No_Subject	No_Source		admin	0	6
113	open	3/15/2016 2:23:25 pm	3/15/2016 2:23:25 pm	No_Subject	No_Source		admin	0	5
18	open	6/9/2015 8:06:26 pm	3/15/2016 2:22:12 pm	test	No_Source	phishing	admin	1	13
89	open	12/16/2015 1:25:09 am	3/28/2016 2:16:58 pm	Suspicious Attacker	Bro	scot	admin	3	47
109	open	2/12/2016 2:18:42 am	3/3/2016 2:53:33 pm	No_Subject	No_Source		admin	2	15
19	open	6/10/2015 8:16:55 am	3/1/2016 7:02:04 am	malware incident	No_Source	keylogger, malware	admin	3	89
111	open	2/23/2016 11:23:03 am	2/23/2016 11:24:06 am	test-1	No_Source,8.8.8.8		admin	1	27
112	open	2/23/2016 11:23:04 am	2/23/2016 11:23:04 am	No_Subject	No_Source		admin	0	10

Page 0 of 3 30

Event #	Subject	Owner	Status	Tags	Source
89	Suspicious Attacker	admin	open	scot	Bro

Add Entry

[123] 3/3/2016 2:57:16 pm by admin (updated on 3/3/2016 3:03:09 pm) Edit Reply

test

Geo Location

city	
region	
country_code	US
org	
country_name	United States

[106] 1/13/2016 8:30:07 pm by admin (updated on 3/28/2016 2:16:58 pm) Edit Reply

Lorem ipsum from 8.8.8.8 6 sdfsd@sdfsd.ru

And to heck with that guy.

[107] 1/13/2016 8:32:36 pm by admin -- Task Owner admin (updated on 1/13/2016 8:32:39 pm) Edit Reply

Block 8.8.8.8. Comes at us all the time and unsolicited.

Fig. 1. Screenshot of the SCOT alert management system

ease future deployments. ALAT has primarily been developed for integration with SNL's internal alert management system, Sandia Cyber Omni Tracker (SCOT) [2]. See Fig. 1 for a screenshot of SCOT. (We have also demonstrated ALAT's effectiveness on registry and malware data to show the applicability of ranking and active learning to other cyber security domains [3]). Currently, we are preparing a trial deployment of ALAT and SCOT for the Department of Homeland Security (DHS).

A. Feature Extraction and Evaluation

Alerts ranked by ALAT are pulled from SCOT and are potentially comprised of thousands of features after feature extraction. Some of these features may be useful for predicting the severity of an alert, while others may not be. The extraction of all possible features from every alert is time consuming. Not only does it take time to extract the features, but the more features we use, the longer it takes us to build our model(s). It is important for us to

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

build our models efficiently because we potentially use hundreds of thousands of alerts in the training process and also need to process alerts in near real-time. Using a smaller set of features that correlate well with the prediction task allows us to use a larger portion of the labeled data for training and to rank new alerts efficiently.

In addition to extracting features directly from the alerts, we also query additional data sources for information to augment our feature vectors. For example, the Security IDentifier Database (SIDD) aggregates IP and domain blocklists from multiple departments and agencies. Additional features we extract from SIDD (e.g., Does this IP that we extracted from the alert appear in any of the blocklists?) may improve our model(s) and the output rankings. We also employ model stacking, where the output of a model is treated as a feature to be used by the the ranking model. We give an example of this when discussing the use of a topic model for feature extraction in Section III.

B. *Learning to Rank*

When the purpose of machine learning is to rank items, it is often referred to as Learning to Rank (LTR) [4]. We explored many LTR models for ALAT and are currently using the probability estimates from a binary classifier, implemented using a random forest model, for ranking.

C. *SCOT and DHS Deployments*

The ALAT framework was initially conceived to rank alerts within SCOT for SNL's internal cyber security operations. This provided us with access to labeled data for supervised machine learning. Each week, a given cyber security analyst, denoted the incident handler, has the responsibility of triaging alerts that come into the SCOT system. Some of these are **closed** as false positives; some are **promoted** to an event; and the vast majority are simply viewed and never closed nor promoted. We now have a labeled dataset of closed and promoted alerts and a much larger dataset of unlabeled alerts. It turns out that this is an excellent fit for active learning: there are some labeled alerts, but the vast majority are unlabeled and it is costly to obtain new labels [5]. The active learning part of the project determines which of the unlabeled alerts would most improve the output of the learned models if labels were available for them. There is a certain expense in terms of an analyst's time to explicitly process these queried alerts, and active learning allows us to judiciously use this resource to obtain these valuable, additional labels.

While ALAT has had a very synergistic relationship with SCOT, ALAT can be applied in any context where alert-like entities are being generated and then processed by an analyst. Processed in this context means they are discarded (i.e., I don't care about this.), highlighted (i.e., I might care about this.), or in process (e.g., I don't know what to do with this yet.). It is our belief that DHS cyber security operational environments will generate alerts of some kind and place those in a queue for analysts to process. Thus, it should be possible to extract implicit labels depending on how they manipulate the various alerts. We also anticipate utility in active learning in these environments as there is almost certainly value to be had by obtaining labels on some of the unlabeled alerts to improve model output.

II. Methods

One technique we used to evaluate feature importance was the insertion of a random feature into our model. Any feature that cannot outperform the random feature is unlikely to have any ability to improve the rankings output by our model. All features that are less important than the random feature either need to be improved or removed from our feature vectors for a specific deployment. (Each deployment requires its own feature evaluation to determine what features are useful in that environment.)

To determine the effectiveness of active learning, one approach is to compare it to passive learning where alerts are queried randomly for labels. This is discussed in more detail in Section III.

We developed a method for evaluating our ranking system to account for the time-dependent nature of alerts. While many machine learning evaluation systems use n-fold cross validation, we needed a method that could segregate our data into time series blocks. To solve this problem, we built a model on alerts from time range X, then tested that model on alerts from time range X+1. We then evaluated our model on time ranges farther out (i.e., X+2, X+3, etc.), which may give us insight into how rapidly adversaries evolve and the associated **concept drift**. Concept drift occurs when models become less accurate as they age and no longer accurately model the current tactics, techniques, and procedures (TTPs) of the adversary. Because adversaries do adapt and change their TTPs over time, we needed evaluation metrics that take this into account. This gives us a more representative evaluation of our system in a dynamic environment and allows us to investigate the effect of concept drift on models being used to detect the TTPs of various adversaries.

As TTPs for various adversaries evolve over time, it is important to account for these time-series characteristics in the models we build or in how we build those models. When compromises are detected, network and/or host indicators of compromise (IOCs) [6] for the TTPs used by those adversaries are developed to prevent future exploits from succeeding. In response, adversaries modify their TTPs to evade detection by the IOCs associated with them. The models that we use to rank alerts become stale over time as adversaries improve their TTPs or intentionally modify them to evade detection. Models built at time X may be very effective at ranking alerts at time X+1, but may lose effectiveness as the gap between the data used to build a model and the live data increases (i.e., concept drift).

We quantify concept drift by building our random forest model from three months of labeled data, or approximately 10,000 alerts (months 1-3). Next, we evaluate our model's performance on alerts from the next three-month window (months 4-6) using class-averaged accuracy (CAA)[1] and precision-recall curves [7]. We then slide our window 1 month and evaluate again (months 5-7). This sliding window analysis was continued for the remaining alerts in our dataset ending with months 17-19. Using this approach, we were able to quantify the concept drift of cyber security alerts, which may give us some insight into the speed at which the TTPs of adversaries evolve.

[1]CAA is calculated by determining the accuracy separately for each class and then taking an average across the classes.

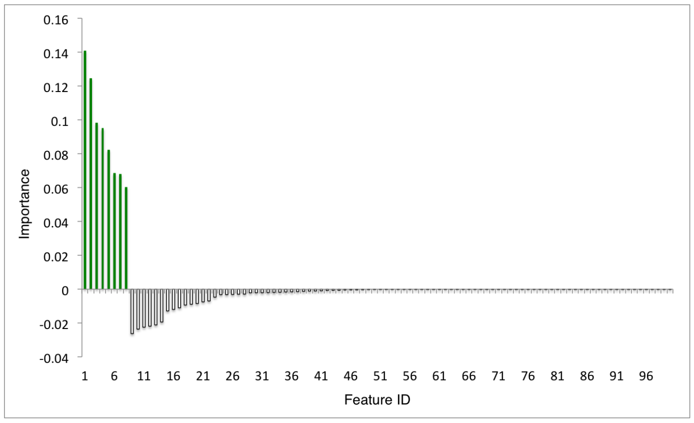


Fig. 2. Plot of feature importances for random forest model

Note that traditional machine learning evaluation techniques, such as n-fold cross validation, are not suitable in this context because they do not consider the time-series nature of alerts, which is critical to our analysis. If alerts are randomly sampled to create the various folds, we can end up building our models with alerts farther out in time than the ones used to evaluate the model. This is unrealistic of real world performance as we cannot build models using knowledge from the future.

III. Results

To determine the importance of each feature in our random forest model, we use the relative depth of a feature in the various paths to decision nodes. Features used in making decisions towards the top of the tree impact the final prediction for a larger fraction of the input samples. Thus, the expected number of input samples whose predictions they might influce can be used as an estimate of the importance of the feature [8]. Figure 2 shows how the extracted features compare to the random feature. Features that performed better than the random feature are shaded green, while features that performed more poorly are left unshaded. The importance values for unshaded features have been multiplied by negative one in order to easily differentiate them in the figure. In our entire feature set, the most useful features were associated with topic membership or time. To determine topic membership, we first generated a Latent Dirichlet Allocation (LDA) [9] model using a corpus consisting of the raw text from the various alerts. For each alert, the model

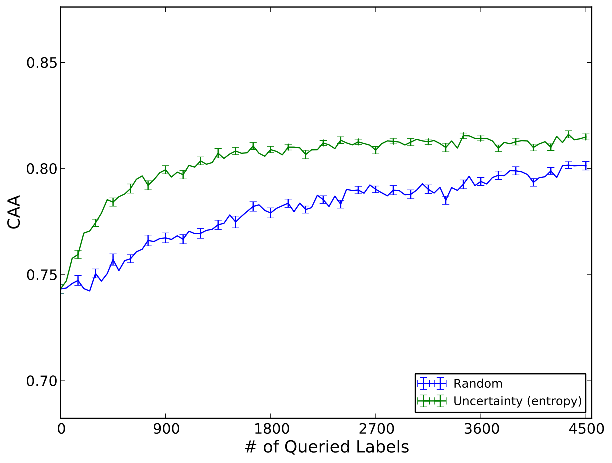


Fig. 3. Comparison of active learning to passive learning

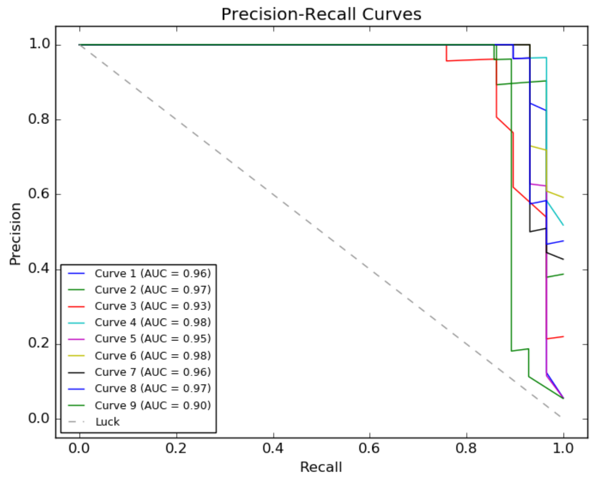


Fig. 4. Precision-recall curves for the models tested against the first 1,000 alerts

assigned probabilities that the various topics were associated with that alert. The number of topics in the LDA model was set a priori, i.e., the model did not determine heuristically how many topics were appropriate for the corpus represented by the alerts. LDA is based on the bag-of-words model where the ordering of words and grammar do not impact the algorithm, i.e., it only uses counts of the various words in each document (e.g., an alert). Four of the seven features that were more important than the random feature were LDA topics. Three time-based features were more important than the random feature including day of the month, month of the year, and the day of the week.

Figure 3 shows how an active learning algorithm, Uncertainty Sampling, compares to passive learning. While the details of how this comparison are performed are somewhat complex, essentially each algorithm (active learning and passive learning) gets to choose a certain number of points to query at each iteration. After labels are obtained, the models are rebuilt and the CAA is obtained. (CAA is a better metric than accuracy for data sets where there is skew in the data. In our data, promoted alerts are far less frequent than closed alerts.) Point 0 on the x-axis represents the performance of the initial model built with 10% of the labeled data before any queries. The figure shows that Uncertainty Sampling has a higher CAA than passive learning for all the experiments once the algorithms were allowed to query the unlabeled alerts.

We next ran a series of experiments to explore the potential presence of concept drift in our data. In Figure 4, we see the performance of the models on the first 1,000 alerts. These alerts were the newest alerts that the models could rank, resulting in excellent area under the curve scores. For this initial test, we built a model nine different times using a different (random) sample of alerts from our training set.

As we moved the sliding window for the time-series evaluation, we continued to use the same nine models described above so that we could see the impact of concept drift, if any, across multiple models. In Figure 5, we see noticeable drops in CAA when the data is 5 and 6 months older than the training data. We also note that the performace overall is inferior to the initial evaluation on the first 1,000 alerts. This performance drop, especially around the 5-6 month mark, is potentially caused by concept drift. Thus, as our models grow older, we can expect them to be less effective against the evolving TTPs of our adversaries.

IV. Conclusions

Our experiments with feature evaluation showed that only the features based on LDA or time were more important than the random feature. Thus, additional feature extraction and evaluation is required to identify new, more predictive features.

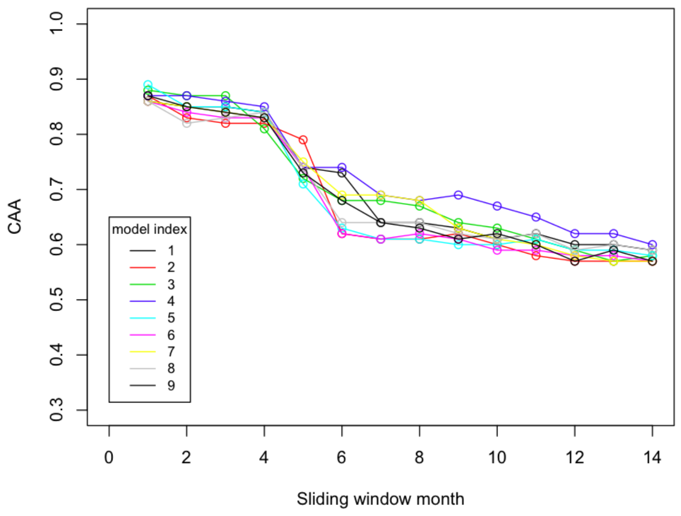


Fig. 5. CAA for sliding window evaluation

We showed that, under our experimental conditions, active learning outperforms passive learning and it thus may be an effective technique at making full use of the unlabeled portion of alert datasets. The time-series evaluation appears to indicate that the TTPs of various adversaries significantly evolve slightly more often than every 6 months. It is important to consider this evolution when developing techniques to defend against adversaries.

V. Future Work

We have expanded the scope of our project to include a trial deployment for DHS. This will allow us to evaluate the performance of our ranking system in a much broader context and at a larger scale. Additionally, we would like to explore techniques to improve our rankings. By using the Laplace correction to smooth probability estimates generated by our binary classifier [10], we hope to address complications that might arise with ties in our ranking system. These ties can be observed when multiple alerts are given the same rank.

References

[1] J. Goldfarb. (2014) 7 Tips To Improve 'Signal-to-Noise' In The SOC. [Online]. Available: http://www.darkreading.com/analytics/7-tips-to-improve-signal-to-noise-in-the-soc/d/d-id/1204605

[2] T. Bruner. (2014) SCOT - Sandia Cyber Omni Tracker. [Online]. Available: http://getscot.sandia.gov/

[3] J. Doak, J. Ingram, and J. Johnson, "Active learning for alert triage," Sandia National Laboratories, SAND report SAND2015-20773, December 2014, Official Use Only.

[4] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[5] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[6] Cyber Security Community. OpenIOC. [Online]. Available: http://www.openioc.org

[7] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[8] scikit-learn developers. Ensemble methods. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[10] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learning*, vol. 52, no. 3, pp. 199–215, 2003.