DETECTING CYBER ATTACKS USING ANOMALY DETECTION WITH EXPLANATIONS AND EXPERT FEEDBACK

Md Amran Siddiqui[1]*, *Jack W. Stokes*[2], *Christian Seifert*[3], *Evan Argyle*[3], *Robert McCann*[3], *Joshua Neil*[3] *and Justin Carroll*[3]

[1]School of EECS, Oregon State University, Corvallis, OR 97331 USA

[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA

[3]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052 USA

siddiqmd@oregonstate.edu, jstokes@microsoft.com, chriseif@microsoft.com, evargyle@microsoft.com, robmccan@microsoft.com, Joshua.Neil@microsoft.com and v-juscar@microsoft.com

ABSTRACT

Detecting cyber attacks in large computer networks is crucial for many organizations. To that purpose, different types of detectors capture the important signals resembling a security attack from individual computers and bring that to the attention of a security analyst. Unfortunately, the analyst sometimes has no indications about why the particular computer was identified as being "under attack". In addition, the analyst may have no method to provide feedback to the detector if the computer was actually identified for some benign reason. In this paper, we use a state-of-the-art anomaly detector called an Isolation Forest [1] for attack detection and generate explanations about why the detector identified certain computers as anomalous. These explanations allow the analyst to direct their investigation in order to save time. We then take the feedback from the analyst in the form of true and false positives and update the anomaly detector to capture signals that align better with the given feedback. Our experiments on actual network data show that the explanations give more insight into the detections, and the analyst's feedback increases the attack detection rate.

***Index Terms*—** Anomaly Explanation, Expert Feedback, Cyber Security, Cyber Attack Detection

1. INTRODUCTION

Detecting cyber attacks using machine learning techniques is a promising new field, and a number of supervised methods have been employed for that purpose [2, 3]. However, one of the major problems of these techniques is that they require a large number of labeled attack examples, which is very challenging to obtain in practice. Furthermore, new and novel types of attacks will continue to occur in the future. As a result, unsupervised techniques such as anomaly detection [4, 5, 6, 7, 8] are becoming popular in detecting these new and diverse types of attacks. In general, anomaly detection techniques rely on some generic "statistical measure" which, unfortunately, sometimes selects benign activity as an anomaly that has nothing to do with a security risk. As a result they tend to produce a large number of false positives which require a security expert (*i.e.*, analyst) to manually verify each one through a costly and time consuming investigation. When an analyst is presented with a set of anomalies, they usually have no knowledge of why the items are actually considered anomalous since the underlying detection technique is a black-box which does not reveal what features led to each sample's selection. Some recent work has tried to mitigate this issue by providing explanations for anomaly detection [9, 10, 11, 12]. We use one such simple explanation method called Sequential Feature Explanation [9] described in Section 3. Once the analyst reviews the explanation, they can direct their research only to the associated features to save time from unnecessary investigation.

After each completed investigation, the analyst usually provides a conclusion of the case as a true positive (TP), false positive (FP) or unknown. An example is labeled as unknown if they are unable to decide if it is a TP or FP with certainty due to the lack of information within their investigational time constraints. Some recent works [13, 14, 15, 16, 17] have proposed improving the anomaly detection performance by incorporating this type of user feedback. We employ one such state-of-the-art technique [13] described in Section 4 that is shown to be simple and superior to others.

The anomaly detector we use in our system is called the Isolation Forest [1] (see Section 2). One major benefit of the Isolation Forest is that it is inherently an ensemble tree-based technique which can be implemented efficiently in distributed systems. We leveraged this advantage so that each tree can be built and updated independently in different computers within a distributed system, and also the anomaly score can be com-

*The first author performed the work while at Microsoft Research for an Internship.

puted in parallel. The first contribution of this paper is the implementation of the Isolation Forest anomaly detector for large distributed systems, which is crucial for processing massive amount of data produced by large organizations. Previous research has separately explored explanations for density-based detectors [9] and feedback with Isolation Forest [13]. In this study, we combine explanations and feedback for Isolation Forest-based anomaly detction. Thus, the second contribution is the production of explanations for the Isolation Forest anomaly detector by adapting the technique in [9] that was originally proposed for the density-based anomaly detectors. The third contribution is the evaluation and incorporation of feedback from a professional security analyst on production data using existing methodologies [13] that demonstrates the efficacy and validity of deploying such systems in practice.

2. ANOMALY DETECTION

Suppose we have collected some network activity data for a certain amount of time from M computers within an organization, and $x_{ij} \in \mathcal{R}^n$ is a set of count-based features representing the activities that occurred in the i^{th} interval from the j^{th} computer. Let, $\mathcal{X} = \{x_{11}, x_{12}, ..., x_{1j}, ..., x_{ij}, ..., x_{iM}\}$ be the collection of all the activity instances across all the computers. $\mathcal{X}$ contains both the benign (*i.e.*, normal) activity as well as the malicious (*i.e.*, attack) events. We assume that the time interval is large enough to capture a malicious event within that period, and if a malicious event is split between two consecutive time intervals, then one of the intervals should have enough data to capture the malicious activity. We also assume that the majority of the data consists of benign activity, while the malicious activity is only a small fraction of the entire dataset. Our goal is to detect all the time intervals containing the malicious events.

We use anomaly detection over the dataset $\mathcal{X}$ to identify the time intervals that are anomalous compared to all other time intervals. We assume that during an attack, the activities of the victim's computer are somewhat different than typical or regular activity. We employ the Isolation Forest [1] as the anomaly detector, which has been shown to be the state-of-the-art by a recent anomaly detection benchmark study [18]. The Isolation Forest builds a collection of trees from the training data, where each internal node is a random threshold test on a random feature sampled uniformly from the input data, and leaf nodes are input instances that followed particular paths from the root node to the leaves based on the threshold tests. The basic intuition is that on average, an anomalous instance should be isolated from the rest of the data with a small number of such random threshold tests and should reside on a shallow leaf on the tree. Each instance falls into an individual leaf node on each tree, and the average depth across all the trees gives the anomaly score $Score(x)$ for the instance x (low score indicates high anomalousness and vice versa). We rank instances based on this anomaly score and report the most anomalous instances to the analyst for further investigation.

3. ANOMALY EXPLANATION

In this section we describe how we compute explanations for the Isolation Forest. We use a feature-based explanation method called Sequential Feature Explanation (SFE) which was proposed in [9]. The goal is to identify a set of salient features, in order of their contribution, which cause a high anomaly score for a particular instance x. We use the greedy technique called Sequential Marginal [9] originally proposed for a joint density function $f(x)$. Sequential marginal greedily chooses the first feature as the one with the lowest density among all the univariate marginal densities. The second feature is chosen from the remaining features that has lowest bivariate density along with the previously chosen feature and so on. Since the Isolation Forest produces a score ($Score(x)$) instead of density estimate, we compute the marginal score for an instance x under a feature subset s as $Score(x_s)$, which is the marginalized score computed by considering the threshold tests involving features only from s across all trees. Essentially, the marginal anomaly score $Score(x_s)$ is obtained by traversing branches of all internal tree nodes whenever the corresponding threshold test on feature F does not belong to the set s, i.e., $F \notin s$. The resultant score is computed recursively by weighting the left and right subtree scores with the fraction of instances that went to left and right branches during training. This method of marginalization is very effective for the Isolation Forest as shown by [19], where the authors call the method "proportional distribution". They considered the set of features not present in the set s as missing value which allows one to compute an anomaly score with any arbitrary set of missing features. Finally, a desired length k explanation $E = (e_1, e_2, ..., e_k)$ can be computed as follows:

$$e_i = \underset{j \notin E_{1:(i-1)}}{\text{argmin}}\ Score(x_{E_{1:(i-1) \cup j}}) \tag{1}$$

4. INCORPORATING FEEDBACK

We present the most anomalous instances to the analyst along with their explanations to obtain their labels. The analyst investigates one or more items and labels each as either a true positive or false positive. We incorporate this feedback using a recent work [13] that shows that feedback can improve the anomaly detection performance significantly. Essentially, they introduce a weight on each internal tree node across all the trees which gives a linear representation of the anomaly score function $Score(x) = w^T\phi(x)$, where w is the weight vector, and $\phi(x)$ is a feature function that transforms x into a high-dimensional binary vector based on which threshold test is true for x from the root to the leaf for each tree. If we set all the initial weights to 1, the linear function will replicate the initial Isolation Forest score exactly. Once we have some labeled instances after the analyst's investigation, we can update the weights so that the score computed from the updated weights are more aligned with the feedback. For example, if we have an instance labeled as a false positive, we do not want to see the instance as well as any other similar instances at the top of the ranking again, and for the true positives, we do want to see more similar instances at the top. To that purpose, the authors in [13] introduced some convex loss functions, which allow simple and efficient online weight updates. We use the following linear loss function from [13]:

$$L_t(w_t) = y_t Score(x_t; w_t) \qquad (2)$$

$$y_t = \begin{cases} +1 & if\ x_t\ is\ "true\ positive" \\ -1 & otherwise \end{cases} \qquad (3)$$

Here, y_t is set according to the feedback. If the current instance x_t is a false positive, we set $y_t = -1$, which makes the loss function become negative of the Score. Thus, we need to adjust the weights w_t to make the Score as large as possible to minimize the loss $L_t(w_t)$ and vice versa for true positive feedback. We update the weights according to the Online Mirror Descent (OMD) algorithm described in [13]. Instead of updating the weights for each individual sample's feedback, we update them as a batch. If the analyst is able to investigate 20 instances in one round, we update the weights using feedback from all 20 instances at once. This can save time for the analyst since they do not have to wait for recomputed scores before starting each new investigation.

5. EXPERIMENTS

We collected data from over two millions of computers reporting during a two week period. To protect the identify of the users, we anonymized the user account and the computer name. We divided the data from computers into two classes, servers and clients, based on their operating system. Since servers tend to have significantly different types of activity than client computers, we filtered them out to focus the analyst's time on the majority computer type. For each of these client computers, we collected network activity containing logon events, RDP and SQL connections. The Isolation Forest anomaly detector consumes numeric vectors as input. Thus, we chose a window size with a time resolution of 30 minutes and counted the different types of activity that occurred in that interval. In total, we used 23 count-based features (Table 1) that gave us a numeric vector of size 23 for each 30 minutes interval, *i.e.*, a total of 672 numeric vectors per computer we calculated over the two weeks of activity. We then filtered out the vectors that did not have any activity, *i.e.*, all of the values were 0. After preprocessing, the final dataset contained approximately 300 million vectors for the client computers.

Anomaly Detector Setup. To create the Isolation Forest model we trained 500 trees, where each tree is grown with 10,000 unique instances (vectors), and the tree is grown until each instance is isolated to its own leaf. One major benefit of the Isolation Forest is that each tree can be trained independently. We leveraged this and constructed all the trees in parallel using Cosmos, Microsoft's internal distributed MapReduce platform. It took about 10 minutes in total to train the entire model. We then computed the anomaly score for each instance using the same distributed environment that allowed us to compute anomaly scores for different instances in parallel. For all the instances the scoring process took about 2 hours. We then selected the top 1000 anomalous instances and produced the explanation for each. The explanation creation process took about an hour.

Table 1. List of Features

SuccessfulLogonRDPPortCount
UnsuccessfulLogonRDPPortCount
RDPOutboundSuccessfulCount
RDPOutboundFailedCount
RDPInboundCount
SuccessfulLogonSQLPortCount
UnsuccessfulLogonSQLPortCount
SQLOutboundSuccessfulCount
SQLOutboundFailedCount
SQLInboundCount
NtlmCount
SuccessfulLogonTypeInteractiveCount
SuccessfulLogonTypeNetworkCount
SuccessfulLogonTypeUnlockCount
SuccessfulLogonTypeRemoteInteractiveCount
SuccessfulLogonTypeOtherCount
UnsuccessfulLogonTypeInteractiveCount
UnsuccessfulLogonTypeNetworkCount
UnsuccessfulLogonTypeUnlockCount
UnsuccessfulLogonTypeRemoteInteractiveCount
UnsuccessfulLogonTypeOtherCount
DistinctSourceIPCount
DistinctDestinationIPCount

Explanation Results. For each instance, we get the anomaly score from the Isolation Forest along with the sequential feature explanation computed as described in Section 3. Table 2 shows some of the example explanations, *i.e.*, SFE with length 3. Note that the raw anomaly scores are normalized according to [1] to make higher scores indicate more anomalousness. The explanations tell the analyst what features were most responsible for the high anomaly score. Also, we report the individual marginal anomaly score along with each accumulated feature, which gives a rough estimation of how much each additional feature contributes to the overall anomaly score. Note that the individual marginal scores are each independent since they have a different set of features, and hence it should not be expected that the overall anomaly score somehow decomposes into them. The last two rows in Table 2 show examples of some actual RDP brute force attempts that correlate with the features shown in the explanations. These examples indicate that the explanations are inherently capturing some cause of the attack.

Incorporating Analyst's Feedback. Once we have the

Table 2. Explanation examples along with anomaly scores

Anomaly Score	Top 3 features Explanation
0.839	UnsuccessfulLogonTypeOtherCount = 40.0 is unusual with score 0.56 SuccessfulLogonTypeNetworkCount = 800.0 is unusual with score 0.70 SuccessfulLogonTypeInteractiveCount = 120.0 is unusual with score 0.77
0.836	SQLOutboundSuccessfulCount = 44.0 is unusual with score 0.54 UnsuccessfulLogonTypeInteractiveCount = 21.0 is unusual with score 0.70 SQLInboundCount = 1568.0 is unusual with score 0.77
0.834	UnsuccessfulLogonTypeOtherCount = 40.0 is unusual with score 0.56 SuccessfulLogonTypeInteractiveCount = 69.0 is unusual with score 0.70 SuccessfulLogonTypeNetworkCount = 547.0 is unusual with score 0.76
1.24	UnsuccessfulLogonTypeNetworkCount = 59.0 is unusual with score 0.54 RDPInboundCount = 368.0 is unusual with score 1.13 DistinctDestinationIPCount = 11.0 is unusual with score 1.17
1.24	UnsuccessfulLogonTypeNetworkCount = 79.0 is unusual with score 0.54 RDPInboundCount = 145.0 is unusual with score 1.13 DistinctDestinationIPCount = 1.0 is unusual with score 1.16

top ranked outliers along with their explanations, we present this ranked list to the analyst. The analysts time is an extremely valuable resource, and we only had access to one professional analyst who could investigate about 20 instances per week in addition to their standard responsibilities. Analysts have access to much richer data logs than is available in the features in our dataset which allows them to make a confident assessment as to whether or not the anomalous activity is malicious. After each investigation the analyst give us their verdict as either "True Positive" or "False Positive". In some cases, the analyst is unable to decide if there is enough information to make a certain decision, so the analyst just labels that instance as "unknown". We only incorporated "True Positives" or "False Positives" in our feedback and ignored the unknowns.

Detection Results. We performed two rounds of feedback with the help of a professional analyst and repeated three iterations each time. Table 3 shows the number of true positives and false positives detected after each iteration of the feedback round. In the first feedback round at iteration 1 we only detect 1 TP and 11 FPs. After incorporating them we discovered 9 TPs and 4 FPs in the next iteration. After we included these new TPs in the third iteration the system became really good at detecting TPs and detected 20 TPs without any FPs. After further investigation, we found that all the TPs discovered in this round belong to the same type of attack known as RDP Brute Force Attempt. Next, we shifted our focus to detecting examples from a different type of attack. To that end, we started a second feedback round with the very first model and intentionally incorporated all the previous TPs and also all the FPs discovered so far as FPs. This strategy forced the model to ignore the RDP Brute Force Attempts. We continue this round for another three iterations and detected 1 and 2 TPs in iteration 2 and 3 respectively. During this round, we uncovered an additional attack behavior, that of network port

Table 3. Detection result after each feedback round

	Feedback Round 1			Feedback Round 2		
Iteration #	1	2	3	1	2	3
# TPs	1	9	20	0	1	2
# FPs	11	4	0	10	17	23

scanning. These results show that in just a few iterations, the system can become very effective at discovering attacks with a small false positive rate for the top K results. We also found that providing TPs as feedback seems to be more effective at discovering similar type of attacks, and FPs as feedback helps to focus on finding something different than the type what the FPs belong to.

6. CONCLUSION

We developed a human-in-the-loop security attack detector using an anomaly detection technique that can provide explanation about the detected anomaly and can improve its detection capabilities using feedback from security experts. In the future, it would be interesting to see how this type of system can detect diverse attack examples with small supervision from analysts.

Acknowledgments. We thank Tommy Blizard for helping us in the cumbersome process of grading the top ranked instances.

7. REFERENCES

[1] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[2] Anna L Buczak and Erhan Guven, "A survey of data

mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[3] Sumeet Dua and Xian Du, *Data mining and machine learning in cybersecurity*, Auerbach Publications, 2016.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 15, 2009.

[5] Martin Grill and Tomáš Pevnỳ, "Learning combination of anomaly detectors for security domain," *Computer Networks*, vol. 107, pp. 55–63, 2016.

[6] Debin Gao, Michael K Reiter, and Dawn Song, "Gray-box extraction of execution graphs for anomaly detection," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 318–329.

[7] R Sekar, Mugdha Bendre, Dinakar Dhurjati, and Pradeep Bollineni, "A fast automaton-based method for detecting anomalous program behaviors," in *sp*. IEEE, 2001, p. 0144.

[8] Xiaokui Shu, Danfeng Yao, and Naren Ramakrishnan, "Unearthing stealthy program attacks buried in extremely long execution paths," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 401–413.

[9] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, and Weng-Keen Wong, "Sequential feature explanations for anomaly detection," *arXiv preprint arXiv:1503.00038*, 2015.

[10] Barbora Micenková, Xuan-Hong Dang, Ira Assent, and Raymond T Ng, "Explaining outliers by subspace separability," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 518–527.

[11] Nguyen Xuan Vinh, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei, "Scalable outlying-inlying aspects discovery via feature ranking," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 422–434.

[12] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang, "Mining outlying aspects on numeric data," *Data Mining and Knowledge Discovery*, vol. 29, no. 5, pp. 1116–1151, 2015.

[13] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer, "Feedback-guided anomaly discovery via online optimization," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018.

[14] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott, "Incorporating expert feedback into active anomaly discovery," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 853–858.

[15] Shubhomoy Das, Weng-Keen Wong, Alan Fern, Thomas G Dietterich, and Md Amran Siddiqui, "Incorporating feedback into tree-based anomaly detection," *arXiv preprint arXiv:1708.09441*, 2017.

[16] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld, "Toward supervised anomaly detection," *Journal of Artificial Intelligence Research*, vol. 46, pp. 235–262, 2013.

[17] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li, "Aiˆ 2: training a big data machine to defend," in *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*. IEEE, 2016, pp. 49–54.

[18] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong, "Systematic construction of anomaly detection benchmarks from real data," in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*. ACM, 2013, pp. 16–21.

[19] Thomas G Dietterich and Tadesse Zemicheal, "Anomaly detection in the presence of missing values," *arXiv preprint arXiv:1809.01605*, 2018.