An Internal/Insider Threat Score for Data Loss Prevention and Detection

Kyrre W. Kongsgård, Nils A. Nordbotten, Federico Mancini and Paal E. Engelstad
Norwegian Defence Research Establishment (FFI)
P.O. Box 25, NO-2027 Kjeller, Norway
kyrre-wahl.kongsgard@ffi.no, nils.nordbotten@ffi.no, federico.mancini@ffi.no, paal.engelstad@ffi.no

ABSTRACT

During the recent years there has been an increased focus on preventing and detecting insider attacks and data thefts. A promising approach has been the construction of data loss prevention systems (DLP) that scan outgoing traffic for sensitive data. However, these automated systems are plagued with a high false positive rate. In this paper we introduce the concept of a meta-score that uses the aggregated output from DLP systems to detect and flag behavior indicative of data leakage. The proposed internal/insider threat score is built on the idea of detecting discrepancies between the user-assigned sensitivity level and the sensitivity level inferred by the DLP system, and captures the likelihood that a given entity is leaking data. The practical usefulness of the proposed score is demonstrated on the task of identifying likely internal threats.

1. INTRODUCTION

As the amount of sensitive information increases, so does the number of information leakage incidents attributed to malicious insiders [16, 13]. Instances of data theft involving insiders exfiltrating a large number of sensitive files have in the recent years become a recurring news event in the mainstream media [17]. Even externally originating data thefts very often utilize compromised internal user accounts and/or computers [16].

Efforts to address this growing data loss concern have focused on the application of data driven algorithms (e.g. machine learning) to infer the sensitivity scores of tabular and textual data objects. These scores may later be used to detect and prevent data transfers that violate the governing security policies, e.g., no "classified" content should leave the internal network.

A more concrete example is shown in Figure 1 which depicts the architecture of a DLP system, in which a data guard connects two security domains, and where a machine learning classifier has been constructed using all labeled or known sensitive documents. Whenever a transfer request is made, the DLP algorithm first analyses the textual content of the data object and computes a predicted security label. If the predicted security label is more restricted than the one assigned by the user the transfer can be dropped and/or the security team notified. The method may also be applied in cases where security labels are not used, e.g., assuming a classifier trained on a set of known sensitive company documents, as any document exported from the network may be considered released by the exporting entity (i.e., implicitly labeled as non-sensitive).

It is crucial that the false positive rate of the system is kept manageable. E.g., in the case of a network intrusion detection system (NIDS), where the classifier operates on a packet level, the sheer volume of false positives may overwhelm the security analysts, and has been reported as one of the main obstacles of wide-scale adoption [10]. While DLP algorithms are typically lower-volume, in the sense that they typically operate on a per-document basis [8], there is an increased effort and desire to move towards algorithms that analyze content on a more granular level [12]. This trend increases the number of events to be analyzed, and thus also the number of false alarms. Also, while attempted leakage of known sensitive documents in unmodified form can be detected with high accuracy, detection of transformed data leaks is much more difficult with a higher rate of false positive/negatives [8]. In such cases, sensitive content may for instance have been rewritten or included as part of another document.

In this paper we discuss how we can use an existing DLP classifier to derive the concept of an *Insider/Internal Threat Score* (ITS). For each entity (e.g., user and/or machine) the ITS quantifies the likelihood that the entity is acting in a nefarious manner. The motivating idea is that by aggregating the outcome of the DLP classifiers over longer time periods we are able to compute robust meta-scores that allow us to gain useful insight despite the large number of false alarms that cripple today's systems. The ITS express the likelihood that a given entity is leaking sensitive data. Furthermore, the ITS may be combined with a misusability score, such as the M-Score/TM-score [2], to provide a better estimate of the risk represented by each user/machine by invoking the relationship between risk, probability and consequence:

$$\mathbf{risk} = \overbrace{\mathbf{likelihood}}^{\texttt{Threat Score}} \times \underbrace{\mathbf{consequence}}_{\texttt{Misusability Score}} \quad (1)$$

Intuitively, the misusability score represents the damage an entity can cause if it leaks data, however, it says nothing

IWSPA'17, March 24 2017, Scottsdale, AZ, USA

ISBN 978-1-4503-4909-3/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3041008.3041011

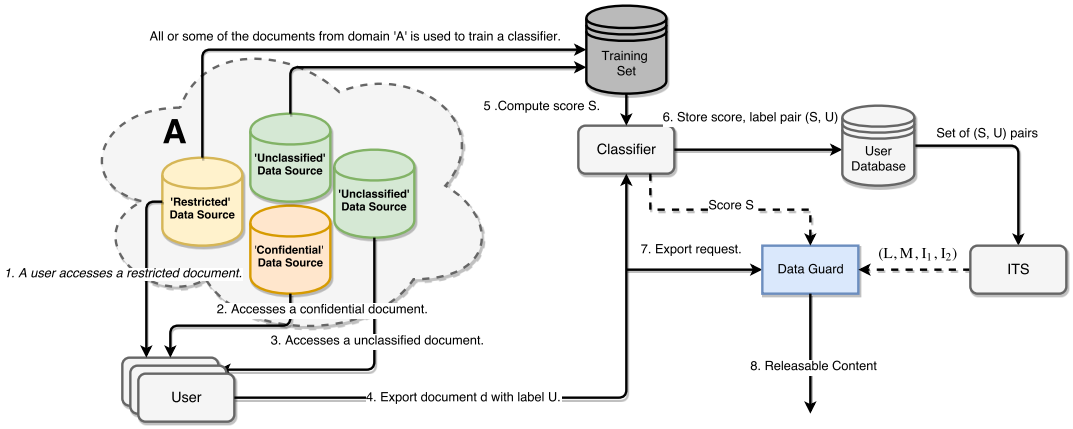

Figure 1: A network environment where users can access sensitive information. Each document exported out of the domain must have a security-label attached, which must satisfy the policy enforced by the data guard (e.g., only *Unclassified* content may be exported). By introducing a machine-learning classifier inferring the sensitivity level using textual contents (e.g. word frequencies), we can detect instances where the user knowingly or unknowingly violates the security policy by exporting classified material. The ITS component uses the set of user assigned labels and predicted scores to infer the misclassification rate for each user.

about the likelihood of the entity actually leaking data. In a context where the data objects are textual documents, the "consequence" factor is represented by a misusability metric such as the TM-Score which can be readily computed [2], while the "likelihood" factor corresponds to the ITS, whose derivation and discussion is the topic of this paper.

The rest of this paper is organized as follows. Section 2 introduces the application domain studied in the experiments and explains how the properties of the underlying labeling classifier influences the design of the ITS. A generative approach in then used to aid the modeling process and we present a Bayesian network model. Experiments is performed using the proposed model, in which its properties and performance are studied. We also analyze attacks for which the model may be vulnerable and propose an outlier detection based defense mechanism. Section 3 demonstrates how the ITS can be utilized in practice. Related work is discussed in Section 4. Section 5 summarizes the work and highlights our contributions.

2. AN INSIDER THREAT SCORE

When deriving an ITS it is instructional to first examine and define the properties it should possess. We assume the existence of an underlying DLP system that infers a sensitivity score $S \in [0, 1]$ for each document, and that each data object is assigned a sensitivity label U by a user. We then base the ITS on the concept of discrepancies, between the user-assigned and inferred sensitivity level. For each user we want to derive a distribution I that captures the propensity to mislabel. From the distribution we can then compute a numeric ITS as $\texttt{ITS} = \mathbb{E}[I]$.

There are other factors that could also be considered, e.g., the amount of imported classified documents (as represented by the TM-Score), sudden bursts in activity or abrupt changes in behavior such as irregular weekend logins and so forth. However, while these properties are known to be indicative of insider behavior [15], they have been studied as part of previous work, e.g., [4], and will not be discussed further in this paper.

Another property of particular importance is that of robustness with respect to manipulation. When applying machine learning algorithms to tasks in information security there are several additional concerns that arise as compared to other application domains. A common underlying assumption for many machine learning algorithms and tasks is that the training and evaluation datasets are both generated from the same unknown, but *stationary* distribution [9]. While reasonable for an OCR problem this assumption is violated for security tasks such as spam detection, automated NIDS and DLP software, where one must take into account the possibility of an attacker actively seeking to thwart detection by attacking and manipulating the security mechanism itself.

We must be particularly wary of manipulation attacks, in which the attacker actively alters the data set in such a way that the classifier mislabels data points [9]. In the context of an insider score the user may influence the computed ITS by carefully choosing the set of documents to export. It is desirable that we define the ITS such that it is robust with respect to manipulation attacks, i.e., it is either very difficult or the risk of exposure for the attacker increases significantly the more he manipulates the system.

2.1 DLP Algorithm

The ITS concept is generic, in the sense that it is applicable to any numerical DLP procedure[1]. As a basis for further experimentation we rely on the dataset and pre-processing done by by Hammer et al. [5]. Using this dataset derived from the Digital National Security Archive we train a binary class logistic regression classifier to distinguish between "classified" and "unclassified" documents. The trained classifier achieved an accuracy of 0.84 on the test set.

2.1.1 *Error Sources*

There are two main sources of errors that negatively influence the accuracy of the underlying binary classifier:

[1]However, the score must be mapped to the interval $[0, 1]$.

1. **Incorrect labeled data points**. It is likely that a subset of DNSA documents were originally assigned an incorrect label. The mislabeled data points will confuse the classifier during the training phase, and they will give a distorted estimate of the performance when the classifier is evaluated, and 2. **Inability to generalize.** Even for machine learning algorithms that satisfy the conditions for the Universal approximation theorem there are no guarantees that we will be able to actually recover the correct function from the learning process.

The combined effect of these errors is reflected in the two score distributions: $P(Y_1)$ (unclassified documents) and $P(Y_2)$ (classified documents), which can be fitted using Beta distributions.

2.2 A Generative Approach

In an operational setting the two key observed DLP quantities are: the security labels assigned by the users and their predicted counterparts, which are expressed in terms of confidence scores.

We can approach the problem by introducing and examining a simplified model of the process of users assigning security labels to data objects. Given a document X to be classified and a user, let $S \in [0, 1]$ denote the random variable whose value represents the probability that the document is classified (as estimated by the machine learning classifier), $U \in \{u, c\}$ the user assigned label, $L \in \{u, c\}$ (i.e., unclassified or classified) the correct label (unknown), M a variable representing whether or not the document was mislabeled (unknown) and I the overall mislabel propensity. Figure 2 visualizes the dependency structure of these variables in terms of a directed graphical model. The links and their directions reflects a generative model of how a user performs the labeling process.

If we a take a closer look at the generative process, we can interpret the variable I as a mixture of the two variables I_1 and I_2 representing the mislabel propensities for unclassified and classified documents respectively. Likewise, the score variable S is a mixture of the two variables Y_1 (scores for 'unclassified' documents) and Y_2 (scores for 'classified' documents) discussed in Section 2.1.1. The distribution of I is known to be skewed in favor of over-classification in the defense industry [14]. A malicious user on the other hand is more likely to have a higher misclassification rate for classified documents.

From the graphical model representation and the set of observed data we can specify the model families of the random variables and infer the values of the hidden structure, thus recovering the distribution of the mislabeling propensities I_1 and I_2 for each user.

2.2.1 *Bayesian Network Model*

If we specify the conditional distribution of each random variable we get the Bayesian network shown in Figure 2. From the observed values of U and S we can infer the posterior density for I_2 and use its mean as the ITS.

In order to perform the fitting we utilized the software package Infer.NET and its implementation of the expectation propagation algorithm [11], while the data was generated using instantiations of the corresponding models implemented in PyMC3 together with the NDSA corpus [7]. By using a probabilistic programming framework like Infer.NET we were able to quickly implement and experiment

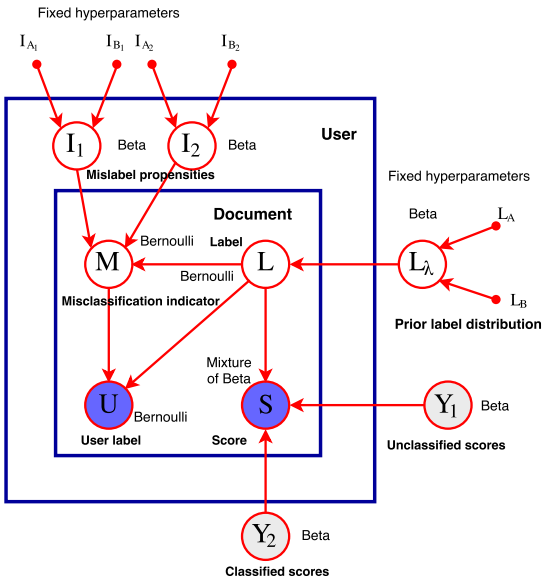

Figure 2: Graphical model expressing the dependency structure of the Bayesian network, and the distributions for each of its nodes. The node U denotes the user-assigned labels, S the predicted scores, M the document mislabel mislabel indicator, L the true label and I the mislabeling propensity. The two variables shaded gray Y_1 and Y_2 are the known score distributions for classified and unclassified documents respectively. Observed variables are shaded blue.

with different models. The alternative, i.e., manually implementing each model, would have been a huge undertaking. However, by relying on Infer.NET we ran into a few restrictions when designing the model[2], e.g., it was not possible to create priors for beta distributions and it proved difficult to learn common Y_1 and Y_2 distributions directly from the complete set of observed user scores. In order to prevent situations in which a skewed distribution would completely dominate the model all users were made to share a common distribution L_λ.

2.2.2 *Manipulation*

An assumption underpinning this Bayesian network, is that we are dealing with unknown yet *stationary* distributions, which as explained in Section 2.1, is not valid in a context in which the user actively seeks to manipulate the system. For the model in question, the attacker has a particular easy task of manipulating the distribution of I_2: Any incorrect classifications, harmful to the operational security of the insider, could later be mitigated by a sequence of confident and correct classifications. It should be emphasized that this attack is only applicable in a scenario in which the guard is either: 1. connected to multiple domains with differing security levels/policies, i.e., it can export both unclassified and classified documents to two domains, or 2. if the ITS is computed when the label is applied as opposed

[2]The field of probabilistic programming is currently undergoing a second renaissance with inference for increasingly complex models becoming feasible [3].

to when the document is exported. Although the task of reducing the ITS score is straightforward, the act of exporting a large number of documents increases the exposure for the insider.

For a setting in which only documents with the user assigned label of 'unclassified' are allowed to be exported it is not possible to perform this attack. However, in this case we must ensure that we are operating with meaningful priors for the L distribution, as the more skewed the L prior is towards the 'unclassified' category, the higher score for a misclassified document must be in order for it to be correctly detected as being mislabeled.

As the scenario in which the ITS operates on both 'classified' or 'unclassified' user assigned sensitivity levels is the most difficult, we will restrict the experiments to this setting.

2.2.3 Evaluation

In order to assess the ability of the Bayesian network in reducing the number of incorrect classifications and alarms as well as detecting insiders, we performed a series of experiments in which 1000 regular users (low misclassification) and 1 insider (high misclassification) were simulated. This experiment was repeated 100 times with a different set of users, with mislabeling rates ranging of from 0.01 to 0.23, and with observations sizes (number of exported/labelled documents) ranging from 5 to 1000. It should be noted that the optimal choice of prior distributions is application specific and has a direct impact on performance. For all experiments we used the weak prior $I_2 \sim \texttt{Beta}(1, 10)$ and a weak uniform prior for L_λ.

Per-document performance: We want to compare the difference in performance, for a per-document classifier, for the three strategies in which we use either: 1. the inferred label L, 2. the user assigned label U or 3. the score S (with the threshold value 0.5) as the predicted document label. There is a notable positive change in accuracy that is achieved by using the two first approaches. This can be attributed to the fact that we are providing the algorithm with the (U, S) pairs, in which the user assigned label U is for the most of the part correct, and thus that the performance is inversely proportional to the mislabeling rate.

It was also observed that using the L variables from the ITS yields a better performance than relying on the user's label, and thus that the model is correctly capturing instances of misclassification.

Error analysis revealed that for many of the incorrectly classified documents the associated L and M values frequently took on non-zero values that were nonetheless smaller than the threshold needed for a correct classification. We can interpret this as although there was not sufficient evidence to completely sway the classification, the model was still able to register that it was uncertainty connected to the assigned/predicted label. This uncertainty was then accounted for by the latent distributions I_1 and I_2 in the form of updated parameter values. This means, as expected, that the inferred I_1 and I_2 distributions are of more use in practice than the per-document M and L values.

Sample size: We also measure how many samples, i.e., the number of exported documents, that is needed to establish the true misclassification rate for a user. Figure 3 (left) displays the relative error of the estimated value of $\mathbb{E}[I_2]$ as a function of export size. For the depicted graph the insider had a misclassification rate of 0.10. As expected, the higher the misclassification rate is the less samples are needed. Although a significant number of documents (> 600) is required to completely determine the true value, even with a small number of documents the estimated mean is larger than those of regular users.

Insider detection: In order to test how effective the value $\mathbb{E}[I_2]$ is in detecting insiders we generate the set of users, compute their latent distributions and then rank them according to the size of their sample means. From this ranked list of users we can compute the mean average precision at k (MAP@k)[3].

The graph of Figure 3 (right) shows the MAP@k (for $k \in [1, 10]$) when the insider user has a misclassification rate of 0.10 and 100 exported documents. From a set of a 1001 users it is not unreasonable for a security team to investigate and audit a set of $k > 10$ users, in which case the performance is high (≈ 0.95).

2.2.4 Counter-measures against Manipulation

As explained in Section 2.2.2, manipulation attacks are feasible only in the less common deployment scenario, in which documents of both classes can be exported. However, it is still of theoretical interest to investigate ways to detect attempts at manipulation. In order to combat these threats we propose an auxiliary anomaly based detection mechanism complementing the computation of the ITS.

Discrepancies between the assigned labels and the predicted scores is reflected in the score distributions of a user. For an attacker to avoid detection he needs to shape the distributions such as to mimic those generated by his non-malicious colleagues thereby blending in with the regular activity of the system. Ideally, we would like to have a single unified graphical model, in which score distributions of the users were also inferred, that we could use to directly construct manipulation counter-measures. As explained in Section 2.2.1 this was not possible due to technical limitations. Instead, we decided to construct a preliminary proof-of-concept algorithm, where we represent each user's score distribution by a feature vector, so that we can subsequently compare a user's current representation with those computed in the past as well as those belonging to his peers. This comparison operation can be used to find instances in which the distribution deviates from regular behavior, which could indicate a manipulation attack.

For each user we construct two multidimensional vector representations for unclassified and classified score distributions respectively. The entries in these vectors belong to one of two classes: 1. summary statistics derived from the score distribution that represent the degree to which the user assigned and predicted security labels deviate and 2. metrics that mitigates the attempt by the attacker to manipulate one or more corresponding score components, e.g., the number of exported and imported documents. As the number of entries increases so does the effort required for a malicious user to blend in with the regular ones. Ideally, the vector entries would be connected or paired in the sense that if a user was to manipulate one of them it would lead to anomalous values in the connected component.

The list of proposed summary statistics used in our exper-

[3]As we operated with a single insider for each performed experiment the insider was either among the top k users or he was not.

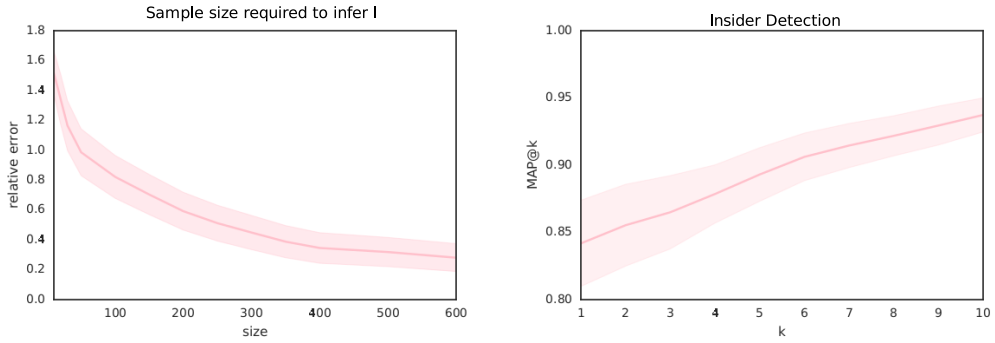

Figure 3: Left: Relative error of the inferred estimate $\mathbb{E}[I_2]$ plotted as a function of the sample size. Right: Mean average precision at k for detecting the insider user among a set of regular users. Y-axis corresponds to the shape parameter $I_2 \sim \texttt{Beta}(a, 10.0)$.

	MAP@1	MAP@5	MAP@10
$\mathbb{E}[I]$=0.07	.19	.49	.61
$\mathbb{E}[I]$=0.13	.61	.73	.77
$\mathbb{E}[I]$=0.23	.91	.99	1.0

Table 1: Manipulation Detection. Performance measured in terms of "Mean Average Precision at k" (MAP@k), for a set of 1001 users. Regular users characterized by $\mathbb{E}[I]\approx 0.01$ while the insider user has $\mathbb{E}[I]\approx 0.07, 0.13, 0.23$.

iments includes the: *mean*, *median*, *skew*, *kurtosis*, *standard deviation*, *variance*, *interquartile range (iqr)* as well as the *min* the *max* values of the observed scores. If we desire sub-scores which are monotonically non-decreasing, as a deterrent to manipulation, then a simple way to establish a score with this property is to aggregate the probability estimates for N documents and include these as separate entries.

In the experiments we used a simple outlier detection algorithm in which we: 1. standardize features (remove the mean and scale to unit variance), 2. compute a common baseline vector for all users by taking the mean, and then 3. compute the euclidean distance between each user and the baseline. The set of users can then be ranked according to distance.

Evaluation was performed analogously to how the main set of experiments were carried out (refer to Section 2.2.3). Users who manipulate the ITS were generated by letting the insider first sample 50 documents with a high misclassification rate and then by alternating between: 1. sampling new documents from Y_2 (labelled correctly) and 2. re-computing the ITS. This two-step procedure was repeated until the sample mean $\mathbb{E}[I_2]$ had fallen into the range of those associated with regular users.

Table 1 displays the performance of the PoC manipulation detection method. The results in the table were achieved without using the monotonically non-decreasing (summation) class of sub-scores. As expected, it shows that the higher the intitial misclassification rate is, then the the easier detection becomes, and that at least certain forms (see description above) of manipulation attacks can be detected through anomalies in the observed user score distribution(s). Future work should seek to incorporate these defense mechanism within a single graphical model.

3. IDENTIFYING INTERNAL THREATS

Having introduced a method of computing internal/insider threat scores; we will in this section demonstrate how they can be applied in practice. We primarily focus on how the ITS allows us to equip IT security personnel with visualizations/overviews that help ease the cognitive load of day-to-day operations by providing situational overviews of the risk each user poses to the organization,

Equation 1 shows how the ITS and misusability weight for a user can be combined to compute the data leakage risk he poses to the organization. These risk estimates facilitates the construction of interactive visualizations providing an overview of the risk each user poses to the organization.

We assume a scenario akin to the one presented in Figure 1, and a user behavior model in which malicious users are characterized by the following set of traits: 1. A high misclassification rate (captured by the ITS), 2. Sudden bursts in activity (periods of increased volume) and 3. Irregular activity, e.g., a user that suddenly starts coming into work on weekend mornings or outside of regular business working hours in general. For each user and time period we can then compute the associated risk-estimate or ITS and display it in terms of interactive visualizations. Figure 4 displays a "calendar heatmap" of the daily ITS for three users in the period January-October. A darker shade of green signifies a higher ITS value. The underlying data was generated from experiments implemented and performed using the discrete-event simulation framework SimPy, in which the inter-arrival time between events (exported documents) is modeled by an inhomogeneous Poisson point process. The three users depicted represents a malicious insider, a regular user and an incompetent user.

Visualizations such as the "calendar heatmap" can be utilized by security operators to detect potential anomalous behavior with respect to both a high degree of mislabeling and other irregular activity. Further analysis can performed by drilling down and querying which and when a user accessed and exported the documents in question.

4. RELATED WORK

To the best of our knowledge no prior research has discussed the concept of an ITS built on the output of existing DLP systems. However, our work is influenced by related concepts such as misusability weights. The M-Score [1] and

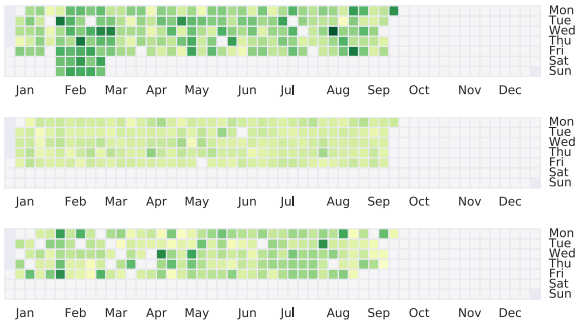

Figure 4: Top: ***A malicious user that has a very high baseline misclassification rate.*** **Middle:** ***A regular user with a low misclassification rate. Inactive during weekends.*** **Bottom:** ***Incompetant user with a high misclassification rate.***

the TM-Score [2] algorithms provide a means of quantifying the damage an organization would incur if the data a user has accessed would leak to 3rd parties. In order to assign a score, domain experts are consulted to manually assess the sensitivity on a per record or document level for a subset of the corpora. These metrics allow us to measure and analyze the consequence of potential data loss on a per user level, and can be further applied to enhance existing anomaly detection algorithms or access control mechanisms.

Legg [15] uses the Carnegie Mellon University CERT Insider Threat Dataset [6] to create an interactive model visualization tool that allows the security team to investigate and analyze the reasoning of an opaque principal component analysis based insider detection algorithm.

For each user in a large private corporation Gavai et al. [4] constructs a set of features encoding information such as the weekly number of messages sent, total time spent on career, entertainment, tech etc. web sites and other social and online activity data. These user profiles are then fed to an anomaly detection algorithm that compares the user with respect to both peers and past behavior.

5. CONCLUSION

In this paper we have introduced the concept of computing internal/insider threat scores using the output of existing data loss prevention systems. We have discussed how an insider threat score should be defined in terms of mathematical properties that reflect the characteristics of known insider behavior. In particular we have focused on incorporating the user properties associated with the act of mislabeling data objects.

We have proposed a generative model in which a we infer the mislabeling propensities using a Bayesian network model (BN). Evaluation experiments have been performed using a modified previously published machine learning classifier and accompanying data set. We also studied an anomaly based method aiming to detect potential manipulation attacks. It relies on computing feature vectors of the score summary statistics in order to allow each user to be compared with past behavior as well as with the behavior of his peers, thus reducing the problem to one of finding outliers. In addition to having been analyzed, the utility of the internal/insider threat score has been demonstrated on the task of identifying likely internal threats.

6. REFERENCES

[1] A. Harel et al. M-score: A misuseability weight measure. *IEEE transactions on dependable and secure computing*, 2012.

[2] A. Vartanian et al. TM-score: A misuseability weight measure for textual content. *IEEE Transactions on Information Forensics and Security*, 2014.

[3] B. Carpenter et al. Stan: A probabilistic programming language. *J Stat Softw*, 2016.

[4] G. Gavai et al. Detecting insider threat from enterprise social and online activity data. In *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats*. ACM, 2015.

[5] H. Hammer et al. Automatic security classification by machine learning for cross-domain information exchange. In *Military Communications Conference, MILCOM 2015-2015 IEEE*. IEEE, 2015.

[6] J. Glasser et al. Bridging the gap: A pragmatic approach to generating insider threat data. In *Security and Privacy Workshops (SPW), 2013 IEEE*, 2013.

[7] J. Salvatier et al. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2016.

[8] K. Kongsgård et al. Data loss prevention based on text classification in controlled environments. In *Information Systems Security*. Springer, 2016.

[9] M. Barreno et al. The security of machine learning. *Machine Learning*, 81(2), 2010.

[10] R. Sommer et al. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, 2010.

[11] T. Minka et al. Infer .NET 2.4, 2010. Microsoft Research Cambridge, 2010.

[12] X. Shu et al. Fast detection of transformed data leaks. *IEEE Transactions on Information Forensics and Security*, 11(3), 2016.

[13] J. Heiser. Understanding data leakage. *Gartner Research*, 2007.

[14] Inspector General, DoD. DoD evaluation of over-classification of national security information, 2013.

[15] P. Legg. Visualizing the insider threat: Challenges and tools for identifying malicious user activity. In *Visualization for Cyber Security*. IEEE, 2015.

[16] Verizon. Verizon's 2016 data breach investigations report, 2016.

[17] B. Vielmetti. Chinese engineer accused of stealing trade secrets from GE, 2014.