Estimating Asset Sensitivity by Profiling Users

Youngja Park[1], Christopher Gates[2], and Stephen C. Gates[1]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2] Purdue University, Indiana, USA
{young_park,scgates}@us.ibm.com, gates2@purdue.edu

Abstract. We introduce algorithms to automatically score and rank information technology (IT) assets in an enterprise, such as computer systems or data files, by their business value and criticality to the organization. Typically, information assets are manually assigned classification labels with respect to the confidentiality, integrity and availability. In this paper, we propose semi-automatic machine learning algorithms to automatically estimate the sensitivity of assets by profiling the users. Our methods do not require direct access to the target assets or privileged knowledge about the assets, resulting in a more efficient, scalable and privacy-preserving approach compared with existing data security solutions relying on data content classification. Instead, we rely on external information such as the attributes of the users, their access patterns and other published data content by the users. Validation with a set of 8,500 computers collected from a large company show that all our algorithms perform significantly better than two baseline methods.

Keywords: Asset Sensitivity, Criticality, Data Security, Information Security.

1 Introduction

Recently, a growing number of advanced persistent threats (APTs) [7] and insider threats [18] have demonstrated the capability of attacking specific highly sensitive entities in a government or company. The computer security community has recognized that not all IT assets have the same value or importance to the company, and, therefore, they require different levels of protection corresponding to their sensitivity and value. By prioritizing the security efforts and budget to better protect highly sensitive assets, organizations can reduce the security risk. Further, quantitative measurement of the sensitivity of IT assets enables other important applications such as intelligent file backup and business continuity planning.

To achieve this vision, all assets in an organization need to be assigned a sensitivity value that properly indicates the business value and criticality to the organisation. Currently, the asset classification is primarily done manually by the system administrators with respect to the confidentiality, integrity and availability of the assets. However, there are critical limitations in the manual approach. First, it is very hard for a large organization to assign appropriate labels to all the assets in the organization. The number of assets in a large organization can grow huge, and, often, the assets are created and managed independently in different departments, so it is extremely hard to catalog and centrally manage all the assets. Second, most of the guidelines are descriptive and can

J. Crampton, S. Jajodia, and K. Mayes (Eds.): ESORICS 2013, LNCS 8134, pp. 94–110, 2013.

be interpreted subjectively. Therefore, the classification of assets can differ significantly by different human judges. Third, they typically measure the sensitivity using a coarse-grained (3 to 5-scale) rating as in the Bell-LaPadula model [5] ranging from the most sensitive (e.g., *Top Secret*) to the least sensitive (e.g, *Un-classified*).

In this paper, we explore methods for semi-automatically scoring various assets within an enterprise using information about the users. To our knowledge, there has been little effort to automatically quantify the sensitivity of IT assets. Previous studies mostly focus on a specific type of assets, e.g., data files [4,12,13] or network assets [3], or propose a ranking method using a small number of manually generated features [10]. We propose a new method for determining asset values using automatically extracted features that are generic to various asset types including data and network assets. We use only information about the users of the target asset including attributes of the users, their access patterns and externally published data by the users such as personal and project webpages and files shared by the users. Note that this information can be easily extracted and does not require direct access to the target asset or detailed knowledge about the asset, such as the owner of the asset and the sensitivity of the data in the asset.

Further, we note that there are many different aspects for an asset being considered sensitive, and the criterion can change over time. For instance, a computer is considered very sensitive because it stores sensitive data (i.e., confidentiality), or it hosts important applications for the business (i.e., availability). Based on these observations, we apply *instance-based learning* approaches, making the system domain independent and easy to adapt to new sensitive asset types. Given a small set of *known* sensitive assets, we learn their characteristics and score other sensitive assets using the models. In this work, we explore a *k*NN (Nearest Neighbor)-based method, a clustering-based method and the *k*NN-based method with distance metric learning techniques. We validate the algorithms using a real-world data set comprising about 8,500 computers. Our experiments show that all our algorithms perform significantly better than the baseline cases, and the *k*NN-based method with distance metric learning techniques outperform the other algorithms. The main contributions of this paper are as follows.

- Previous studies presented solutions for a specific IT asset type such as data, servers or computer networks, forcing companies to manage multiple heterogenous approaches. Our methods rely on meta-level information that can be extracted from most IT assets in the same way. This domain-independent set of features makes our methods applicable to many different IT asset types.
- Further, extraction of the meta-level features does not require direct access to the target assets or privileged knowledge about the assets, and, thus, our method is very efficient and can be easily scalable to a large set of heterogeneous assets.
- Our system assigns a quantitative value to each asset rather than a coarse-grained set of labels, allowing companies to adopt more fine-grained security measures.
- A major obstacle in applying machine learning methods to computer security problems is the lack of labeled data. In this work, we propose new semi-supervised machine learning methods that learn the characteristics of sensitive assets from a small number of examples.
- We validate our approaches with a large set of real data. Experimental results confirm that the proposed algorithms can retrieve sensitive assets with high ranks producing higher precision and recall than baseline methods.

2 Meta-level Features of Assets

As discussed in the introduction, a main goal of this study is to identify a set of features that can be uniformly used for different asset types and be extracted without having to access the target asset or privileged knowledge about the asset. This set of features may not be as accurate as a small set of features carefully produced by domain experts, but it makes the system very efficient and scalable and can provide a good estimate for *potentially* sensitive assets.

In this study, we investigate 72 features from three kinds of knowledge — who accesses the asset (user features), how they access the asset (usage features) and what kinds of tasks or projects the users work on (external content features). Table 1 describes the high-level feature categories used in this study.

Table 1. Features for estimating the sensitivity of IT assets

Feature Categories	Feature Definition
User Features	
Manager vs. NonManager	Is the user a manager or a non-manager employee
Job Roles	Job roles in the organization such as *S/W Developer* and *Finance*
Rank in the organizational hierarchy	The distance from the highest-ranked employee to the user in the organization hierarchy
Usage Features	
Access Frequency	the total number of accesses by a user (heavy or light)
Access Pattern	the patterns of the accesses (e.g., regular, semi-regular, irregular)
External Content Features	
External Data Content	Topics discovered from the externally published data content such as papers, patent and webpages of the users

2.1 User Features

User attributes such as job roles and the rank in the organization may affect the sensitivity of the asset. For instance, an asset used primarily by executives would elevate the sensitivity of the asset. In this work, we leverage these types of user attributes for sensitivity estimation.

To extract the attributes of the users, we first need to identify the users of the asset in the access logs. Some access logs, such as logs for a file repository or a system logon, typically contain the user accounts, thus, identifying the users is straightforward for these assets. For computer network assets, user accounts are generally not available in the logs (e.g., DNS logs). Instead, the logs contain the IP address from which the lookup was requested. The process of determining which user is performing a DNS lookup is not a trivial task. In most situations, we first need to find the most likely candidate user who is assigned to a specific IP address during a specific time period. The resolution

of an IP address to a user, while easy in a simple system, becomes more challenging in a dynamic system with many different ways to access the network and with a large set of users. Users can log into the network over WiFi or using an ethernet cable, or from remote locations via VPN (virtual private network).

For computers in a network, we perform the IP to user mapping using various sources including media access control (MAC) addresses, application (e.g., internal web portals) logs, and WiFi logs. If the MAC address is present, then, during a DHCP session setup, we can correlate the MAC address used for that session to the IP address that is assigned, which, in turn, can give us an IP to user mapping. However, the MAC addresses are not reliable for users using OS X and are often unavailable when new devices are introduced. To alleviate the limitations, we also use application and WiFi logs for the user mapping. The application level logs can correlate the act of a user logging into an application (such as an internal web portal) to an IP address. The WiFi logs can correlate a user establishing a connection to the WiFi with the authentication credentials that are used to log in to the system. Since the user to IP mapping is not perfect, we discard all DNS lookups for which we are unable to identify the user and all logs that are resolved to more than one user (i.e., ambiguous logs) for our study.

After obtaining the set of users of an asset, we extract various user attributes that can indicate the users' job roles and the sensitivity of the data they generate. The high-level categories of the user attributes used in this work are shown in Table 1. We extract 26 user attributes in total including *Manager*, *NonManager*, *Rank-High*, *Rank-Middle*, *Rank-Low*, and 21 different job roles defined in the company such as *IT Specialist*, *Human Resources* and *Finance*. Note that these attributes can be extracted from most companies' employee directory. The feature value of each feature is the number of users who possess the attribute. For instance, if 100 managers, 500 non-manager employees and 1 high-rank employee accessed the asset, the asset is represented *Manager*=100, *NonManager*=500 and *Rank-High*=1.

2.2 Usage Features

The access patterns of the users add additional insights on the sensitivity of an asset. For instance, a user who occasionally uses the asset will have less impact than a user who uses the asset frequently. On the other hand, if a user's access pattern is very regular (e.g., every day at 5am), that may indicate that the user is running an automated job (e.g., file backup), so the accesses should not affect much on the asset's sensitivity. Figure 1 shows typical daily DNS lookup activities.

In this work, we analyze access logs with the timestamps to discover the frequency of a user's access and the patterns of the accesses. We first group the logs by each pair of a user and an asset, and record the number of log entries as the access frequency of the user to the asset. We categorize the access frequency into *Heavy* or *Light* using a predefined threshold. Further, we determine if a connection to the asset is done through an automated access or a manual access (i.e., access pattern). We observe that automated accesses tend to be regular, for instance, once a day at 4am or once every hour, while human accesses are more sporadic. In other words, automated accesses are more predictable while human accesses are more uncertain. Based on this observation, we apply

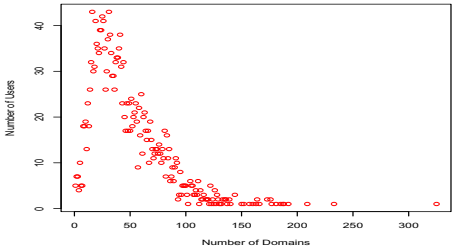
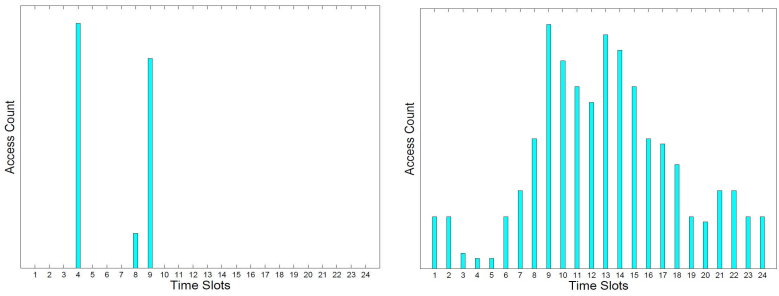


Fig. 1. Number of unique domains accessed per user in a single day. The data show that most users access 20 to 30 different domains in a day, while a few users connect to over 200 different domains.

the Shannon entropy, $H(X)$, which measures the uncertainty in a random variable [16] to determine the access patterns.

$$H(X) = -\sum_{i} p(x_i) \log(p(x_i))$$

Now, we explain in detail how we measure the entropy of user accesses. First, for each user and asset pair, we split all the accesses over each hour of the day (i.e., grouping accesses into 24 time slots). For instance, we count how many accesses a user initiated at the 9am–9:59am period in the logs collected over a long period time. Figure 2 shows two sets of access patterns over the 24 time slots. Figure 2(a) illustrates cases where the accesses were made at the same time periods repeatedly, while Figure 2(b) shows cases where the accesses spread across many different time slots. After obtaining a



(a) Regular Access Pattern (b) Irregular Access Pattern

Fig. 2. Access Patterns

24-dimensional count vector for a user-asset pair, we then normalize the counts into probability distributions and compute the entropy. If an access distribution produces a low entropy, then the accesses are regarded as automated accesses. We divide access

patterns into three categories–*Regular*, *SemiRegular* and *Irregular*–based on the entropy values (i.e., high, medium and low respectively).

By combining the access frequency and the access pattern features, we generate 6 usage features: *RegularHeavy*, *RegularLight*, *Semi-regularHeavy*, *Semi-regularLight*, *IrregularHeavy* and *IrregularLight*. If the accesses by a user to an asset exhibit a regular pattern (i.e., low entropy), and the user has a large number of accesses, it is considered as *RegularHeavy*. On the other hand, if the access pattern is irregular (i.e., high entropy) and the access count is low, then it is considered as *IrregularLight*. Similarly to the user features, the number of users that exhibit a certain access pattern is the feature value for the asset, i.e., how many users access the asset using *RegularHeavy* or *RegularLight* pattern.

2.3 External Content Features

The sensitivity of an asset is dependent largely on how sensitive the data in the asset are, and, thus, the topics of data in the assets can be good indicators of the asset sensitivity. When content inspection can be performed, the sensitivity can be measured by the techniques presented in [12,13]. When direct content inspection is not feasible, we propose to use external data contents generated by the users as a substitute. External contents of a user can include any documents or data sources the user produced outside the target asset, such as papers, patents, and project webpages. These external contents are used to conjure the user's job responsibilities and the tasks the user is working on. Note that we only extract the contents that can be accessed without an access permission to the host system. Some examples of external data content include:

- Published documents such as patents and papers
- Titles of files the user has shared in a file-sharing site
- Wiki or project websites where the user is a member of
- Personal webpages
- Blogs created by the user
- Tags the users added on webpages

Document of a User: We combine all the external data published by a user and generate a document for the user using the *bag-of-word* representation. We then remove *stop words* [1] and count the occurrences of each word in the user document. The basic assumption is that more frequently used words indicate the topics of the user more strongly than less frequently used words.

Document of an Asset: We then generate a hypothetical document for an asset by combining the documents of its users. Furthermore, we assume that the users who access the asset more frequently influence the content of the asset more than the users who uses it occasionally. We scale the frequency of words in the user documents based on the frequency of the user's access, which is defined as the number of days the user accessed the asset. Figure 3 depicts the high level process of generating documents for assets, and Definition 1 provides a formal description.

[1] Stop words are very commonly used words in most documents such as prepositions (e.g., "to", "in") and pronouns (e.g., "I", "this").

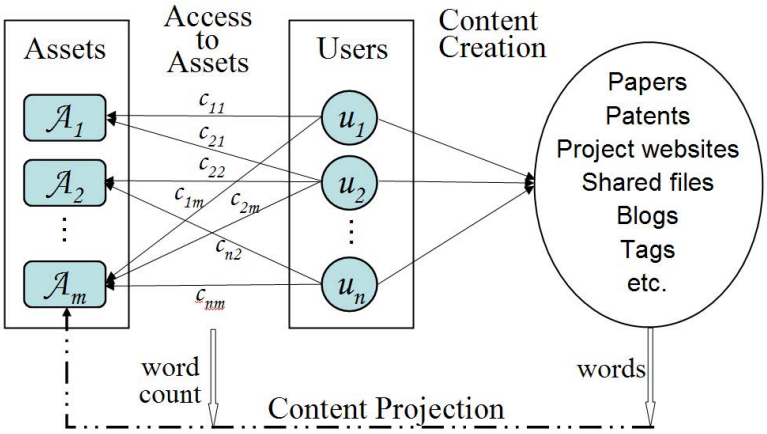


Fig. 3. High level description of content generation for assets using external user contents and the users's access counts to the assets. The words in an asset document come from external contents generated by the asset users, and the counts of the word occurrences in the document are determined based on both the word counts in the user documents and the users' access counts for the asset.

Definition 1. *Let asset $\mathcal{A}$ have n users, $\mathcal{U} = \{u_1, \cdots, u_n\}$, and the document of a user u_i be $\mathcal{D}(u_i)$. Then, the document of asset $\mathcal{A}$, $\mathcal{D}(\mathcal{A})$, is defined as $\mathcal{D}(\mathcal{A}) = \cup_{u_i \in \mathcal{U}} \cup_{w_j \in \mathcal{D}(u_i)} w_j$. Further, the count of a word in $\mathcal{D}(\mathcal{A})$, $c(w_j)$, is computed as*

$$c(w_j) = \sum_{i=1}^{n} \delta_i \cdot c(w_{ji})$$

$c(w_{ji})$ is the count of word w_j in $\mathcal{D}(u_i)$, and δ_i is the weight of user u_i for the asset $\mathcal{A}$ and defined as $\log(\#days(u_i, \mathcal{A}))$.

Topic Discovery: Once we generate a document representation of an asset, a set of assets can be considered as a collection of documents. The document collection for all assets in an organization typically contain a large number of words. Treating individual words as features will result in a very high dimensional feature space and data sparseness issues. Instead, we can group the words into topics and use the topics as the content features. Each asset can then be represented as the probability distributions over the discovered topics.

In this work, we apply Latent Dirichlet Allocation (LDA) [6], a generative topic modeling technique, to discover the topics from a collection of documents. LDA is a probabilistic generative model for collections of discrete data such as text collections. Each document in a corpus is modeled as a finite mixture over underlying set of topics, and each topic is, in turn, modeled as a distribution over words. LDA allows for multiple topic assignments to a document (i.e., probabilistic clustering) and, thus, better explains the underlying topic distributions in the given corpus.

LDA assumes the following generative process for creating a document d in a collection of document $\mathcal{D}$:

1. For each document $d \in \mathcal{D}$, a distribution over topics is sampled from a Dirichlet distribution, $\theta \sim Dir(\alpha)$.
2. For each word w in a document, select a topic, z, according to the distribution, *Multinomial*(θ).
3. Finally, a word is chosen from a multinomial probability conditioned on the topic, $p(w|z, \beta)$. β is a matrix of word probabilities over topics which is to be estimated from the training data.

LDA requires the number of topics to be discovered as an input parameter. In this work, we run LDA with 40 topics, and, therefore, each asset is represented as a probability distribution over the 40 topics. Table 2 shows three sample topics discovered from our data set.

Table 2. Sample topics discovered from document representations of computer servers. Topic5 indicates *Speech Recognition*, Topic28 is related to related to *Analytics and Business Intelligence*. *BAMS* stands for business analytics and management. Topic37 is related to *Computer Security*.

Topics	Most Relevant Words
Topic5	speech, recognition, system, using, models, language, translation, based, detection, arabic, transcription, model, speaker
Topic28	business, community, management, analytics, method, system, supply, project, *BAMS*, data, performance, applications, research
Topic37	system, computing, virtual, security, community, secure, method, research, data, trusted, applications, operating

2.4 Feature Normalization

The selection of features is critical for machine learning methods, as the data are represented as points in a multi-dimensional feature space, where a feature corresponds to an axes. Another important consideration is the range of feature values. Most data mining and machine learning algorithms rely on a metric or a distance function to evaluate how similar two data points are in the feature space. When there is a large difference in the range of the feature values along different axes, these metrics implicitly assign higher weights to features with larger ranges. To mitigate the effect, a feature normalization technique is often applied and converts all features into an equal range.

In this study, the values of the user and usage features are the counts of the features in the target asset, while the content topic features are the probabilities in range of [0, 1]. The raw count values, especially for the usage features, can grow very large when the data set is collected over a long time period. We normalize the user and usage features using the cumulative distribution function (CDF) following the findings by Aksoy and Haralick [1] [2]. CDF-based feature normalization is performed as follows. Given a

[2] We experimented with other feature normalization techniques such as linear scaling, unit range normalization and rank normalization, and the CDF normalization performed best for our data.

random variable $x \in \mathbb{R}$ with cumulative distribution function $F_x(x)$, the normalized feature value, $\tilde{x}$, of x is defined as $\tilde{x} = F_x(x)$ which is uniformly distributed in [0, 1].

3 Sensitivity Estimation Algorithms

In this section, we present our algorithms for estimating the sensitivity of assets. As noted earlier, there are many different aspects that make an asset sensitive to the organization. For instance, an asset is considered sensitive because it contains sensitive business data, or it hosts important applications. Based on these observations, we apply *instance-based learning* approaches, in which we learn the characteristics of sensitive assets from a small number of *known* sensitive assets. Therefore, our methods do not require any prior knowledge about the domain or the target assets, making the algorithms very flexible and easy to adapt to new domains. In this work, we explore three semi-supervised machine learning approaches: a *k*NN-based method, a clustering-based method, and the *k*NN method with distance metric learning techniques.

3.1 *k*NN-Based Method

The *k*-nearest neighbor classification is a type of *instance-based learning* which assigns a new data point to the majority class among its k nearest neighbors from the training data set [8]. The kNN approach is extremely flexible and non-parametric, and no assumption is made about the probability distribution of the features. The similarity is computed based on the distances between feature vectors in the feature space.

More formally, let $X = \{x_1, \ldots, x_n\}$ be the training data set, and $Y = \{y_1, \ldots, y_C\}$ be the set of classes. In the basic kNN classification, the class for a new data point x is defined as $\arg\max_{1 \le i \le C} \sum_{j=1}^{k} 1(y_i, y_j)$, where y_j is the class of the j-th neighbor, and $1(y_i, y_j)$ is an indicator function that returns 1 if $y_i = y_j$ and 0 otherwise. In many applications, the vote is weighted by the distance between the new point and a neighbor, and the decision is influenced more by closer neighbors.

$$\arg \max_{1 \le i \le C} \sum_{j=1}^{k} \omega(d(x, x_j)) \cdot 1(y_i, y_j)$$

where $\omega(d(x, x_j))$ is a weight function that is inversely related to the distance $d(x, x_j)$.

In this work, we extend the weighted kNN approach and compute the sensitivity of a new asset based on the distance to its *k*NN assets in the training data and the sensitivity scores of the *k*NN assets. When the sensitivity scores are not provided for the training data, we can assign the same value to all the training data. The sensitivity of a new asset $\mathcal{A}$, $\mathcal{V}(\mathcal{A})$, is then defined as a weighted average score of its *k*-nearest neighbors among the *known* sensitive assets, $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$.

$$\mathcal{V}(\mathcal{A}) \;=\; \sum_{i=1}^{k} e^{-d(\mathcal{A}, \mathcal{S}_i)} \cdot \mathcal{V}(\mathcal{S}_i) \tag{1}$$

$\mathcal{V}(\mathcal{S}_i)$ is the sensitivity value of $\mathcal{S}_i$, and $e^{-d(\mathcal{A}, \mathcal{S}_i)}$ is the weight function where $d(\mathcal{A}, \mathcal{S}_i)$ is the Euclidean distance of the two assets. The *k*NN-based sensitivity estimation is described in Algorithm 1.

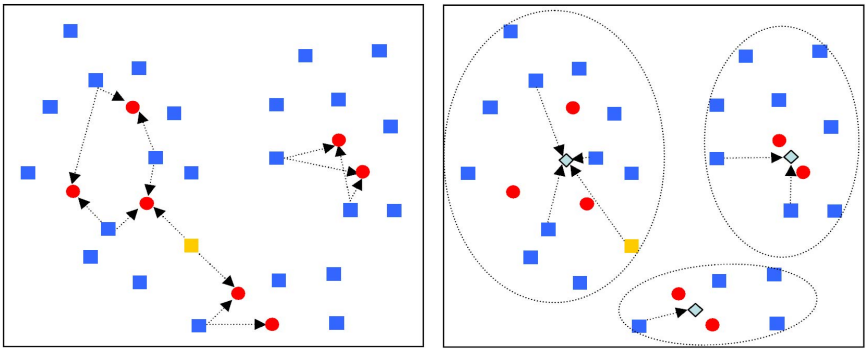
Algorithm 1. Sensitivity Estimation based on k-Nearest Neighbors

1: **Input:** Unlabeled assets $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, a set of *known* sensitive assets $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, and, *optionally*, the sensitivity scores of $\mathcal{S}$, $\mathcal{V} = \{\mathcal{V}(\mathcal{S}_1), \ldots, \mathcal{V}(\mathcal{S}_m)\}$
2: **Output:** Ordered list of assets $\mathcal{A}' = \{\mathcal{A}'_1, \ldots, \mathcal{A}'_n\}$, where $\mathcal{V}(\mathcal{A}'_i) \geq \mathcal{V}(\mathcal{A}'_{i+1})$
3: **for** $\mathcal{A}_i \in \mathcal{A}$ **do**
4: $\quad kNN(\mathcal{A}_i) \leftarrow \{\mathcal{S}_i, \ldots, \mathcal{S}_k\}$, k assets from $\mathcal{S}$ that are closest to $\mathcal{A}_i$
5: $\quad$ Compute the sensitivity of $\mathcal{A}_i$, $\mathcal{V}(\mathcal{A}_i)$ using Equation (1)
6: Sort $\mathcal{A}$ in descending order of $\mathcal{V}(\mathcal{A}_i)$

3.2 Clustering-Based Method

The clustering-based method considers that the assets are from many different business units such as product development groups, HR or Finance department, and, therefore, they will naturally form distinct groups. Suppose only one sensitive asset from the HR department is included in the training data. With the kNN method with $k > 1$, the sensitivity of assets from the HR department will be measured with assets from other departments. By taking into account the subgroups in the dataset, we can determine the sensitivity level of an asset using the sensitive assets from the same subgroup.

First, a clustering technique is used to discover these underlying subgroups in the data set. We then generate the centroid of the sensitive assets in each cluster, which is the the mean of the sensitive assets in the cluster. Similarly to the *k*NN-based method, we measure the sensitivity of an asset $\mathcal{A}$ as the weighted average score of the k-nearest centroids as described in Algorithm 2. The difference of the *k*NN-based approach and the clustering-based approach is illustrated in Figure 4.



(a) *k*NN-based method where *k*=2 (b) Nearest centroid-based method

Fig. 4. Illustrations of the kNN and Clustering-based methods for sensitivity estimation. The circle symbols denote known sensitive assets and the square symbols denote unlabeled assets. The diamond symbols in 4(b) represent the centroid of the sensitive assets in each cluster. Note that the sensitivity of the light-colored (yellow) square is measured with a sensitive asset from a different cluster in Figure 4(a).

Algorithm 2. Sensitivity Estimation based on k-Nearest Centroids

Input: Unlabeled assets $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, a set of *known* sensitive assets $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, and, *optionally*, the sensitivity scores of $\mathcal{S}$, $\mathcal{V} = \{\mathcal{V}(\mathcal{S}_1), \ldots, \mathcal{V}(\mathcal{S}_m)\}$
Output: Ordered list of assets $\mathcal{A}' = \{\mathcal{A}'_1, \ldots, \mathcal{A}'_n\}$, where $V(\mathcal{A}'_i) \geq V(\mathcal{A}'_{i+1})$
Cluster all assets, $A \cup S$, into K subgroups, $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$.
for $\mathcal{C}_i \in C$ **do**
    $\mathcal{S}^i \leftarrow \mathcal{C}_i \cap \mathcal{S}$ // the set of sensitive assets in $\mathcal{C}_i$
    $\bar{C}_i \leftarrow$ the centroid of $\mathcal{S}^i$
    $\mathcal{V}(\bar{C}_i) \leftarrow$ the mean sensitivity value of $\mathcal{S}^i$
for $\mathcal{A}_i \in \mathcal{A}$ **do**
    Let $\bar{C} = \{\bar{C}_1, \ldots, \bar{C}_k\}$ be the k nearest centroids from $\mathcal{A}_i$
    $\mathcal{V}(\mathcal{A}_i) \leftarrow \sum_{i=1}^{k} e^{-d(\mathcal{A}, \bar{C}_i)} \cdot \mathcal{V}(\bar{C}_i)$
Sort $\mathcal{A}$ in descending order of $\mathcal{V}(\mathcal{A}_i)$

3.3 kNN Method with Distance Metric Learning

The accuracy of many machine learning algorithms including both *k*NN classification and clustering is heavily dependant on the distance (or similarity) metric used for the input data. However, when the data are in a high-dimensional space, the selection of an optimal distance metric is not intuitive. Distance metric learning is a machine learning technique that aims to automatically learn a distance metric for the input data from a given set of labeled data points. The basic idea is to learn a distance metric that puts instances from a same class closer to each other and instances from different classes far apart. Recently, many studies have demonstrated that an automatically learned distance metric significantly improves the accuracy of classification, clustering and retrieval tasks [17,14,20].

Distance metric learning algorithms are further divided into global distance metric learning and local distance metric learning. Global distance metric learning algorithms learn a distance metric that satisfy all the pairwise constraints, i.e., keep all the data points within the same classes close, while separating all the data points from different classes. Local distance metric learning algorithms, on the other hand, learn a distance metric satisfying local constraints, and has been shown to be more effective than global distance learning for multi-modal data.

In this study, we apply a global distance learning algorithm and a local distance metric learning algorithm to transform the feature space. For global learning, we apply Relevant Component Analysis (RCA) [17] to learn a distance metric as proposed in [2]. The RCA-based distant metric learning algorithm learns a Mahalanobis distance metric using only equivalence constraints (i.e., instances in the same class) and finds a new feature space with the most relevant features from the constraints. It maximizes the similarity between the original data set X and the new representation Y constrained by the mutual information $I(X, Y)$. By projecting X into the new space through feature transformation, two data objects from the same class have a smaller distance in Y than in X. For local distance metric learning, we apply the Large Margin Nearest Neighbor (LMNN) distance learning algorithm [14]. The LMNN algorithm also learns a Mahalanobis distance metric, but it identifies k-nearest neighbors, determined by Euclidean

distance, that share the same label and enforces the k-nearest neighbors belong to the same class while instances from different classes are separated by a large margin.

After the feature space projection using the distance metric learning algorithms, we apply the kNN-based sensitivity estimation method described in section 3.1.

4 Experimental Results and Evaluation

To validate the algorithms, we conducted experiments with a real life data set comprising about 8,500 computers. In this section, we describe in detail the experimental settings and evaluation results. Henceforth, we denote the kNN-based method using the original feature space as kNN, the centroid-based method as $Centroid$, the kNN method with the LMNN distance metric learning as $LMNN$, and the kNN method with the RCA distance metric learning as RCA.

4.1 Data

The computers used in the experiments were extracted from DNS logs collected in the authors' organization over 3.5 months from April, 1, 2012 to July, 15, 2012. We extracted 12,521 unique computers for which we were able to identify the user but discarded the computers with only one user or fewer than three look-up requests, resulting in 8,472 computers. We use the 8,472 computers for training and evaluation of our models–80% of the computers for training and 20% for evaluation respectively. Using the mapping of IP address to user described in section 2.1, we identified 2,804 unique users for the 8,472 computers.

In a separate effort, the company had attempted to manually compile a list of servers, for the purpose of disaster recovery and business continuity, that host important applications of the company. The list provides the server names and their business criticality value (BCV) assigned manually by domain experts. Each computer is assigned with a BCV from five BCV categories–BCV1 to BCV5–and each BCV category is associated with a numeric value from 10 (BCV1) to 50 (BCV5). We found 253 servers from this list in our collected data set, and, thus, use the 253 servers as the labeled (i.e., ground truth) data for this study. The ground truth data account for about 3% of the experimental data, and we use the data set for both training and evaluation of the algorithms. Table 5 and Figure 6 show the size of the experimental data, the size of the ground truth set, and the distribution of the ground truth data over the five BCV categories.

4.2 Evaluation Metrics

We observe that the problem of identifying sensitive assets can be cast as an information retrieval (IR) problem — finding relevant (sensitive) assets in a large collection of assets and ranking them according to their relevance. This allows us to apply the evaluation metrics developed for IR such as recall, precision and discounted cumulative gain (DCG) [19,11,9] to validate the performance of our algorithms.

No. of computers	8,472
No. of unique users	2,804
No. of known sensitive computers	253

Fig. 5. Experimental data set

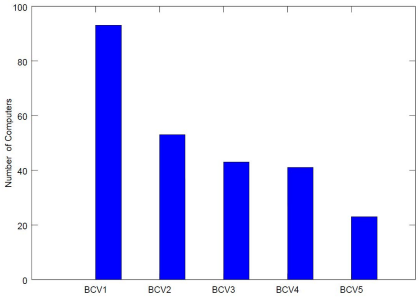


Fig. 6. Distribution of the ground truth data across the business criticality values

Precision and Recall: Precision and recall are widely used metrics for binary decision problems including information retrieval and pattern recognition. In a binary decision problem, a system labels data samples either positive or negative. Precision measures the fraction of samples classified as positive that are truly positive, and recall measures the fraction of positive samples that are correctly classified.

$$Precision = \frac{|\{\text{true positives in the result}\}|}{|\{\text{all samples in the result}\}|} \quad Recall = \frac{|\{\text{true positives in the result}\}|}{|\{\text{all positive samples}\}|}$$

In a ranked retrieval context as in our study and in most web search engines, precision and recall are typically measured at the top n results. Further, when the class distribution is skewed, Precision-Recall (PR) curves are often used. A PR curve is generated by plotting the precision at different levels of recall rates, and provides a more comprehensive view on the system's performance.

Discounted Cumulative Gain (DCG): In addition to ranking the results, when the relevance of an instance is measured using a multi-scale rating (e.g., from *completely relevant* to *completely irrelevant*), the quality of the results can be more precisely measured using a graded relevance scale of the results. For instance, two search engines can produce the same precision and recall, but the search engine that retrieves documents with a higher relevance scale at the top of the results is more useful.

DCG measures the usefulness (or gain) of a search result based on its position in a search result list. The gain of each result is discounted logarithmically proportional to its position in the ranked list, and the DCG of a system is defined as the accumulated gain from the top of the result list to the bottom [9].

$$DCG = REL_1 + \sum_{r=2}^{n} \frac{REL_r}{\log_2(r)}$$

where REL_r is the relevance score of the result at rank r, and n is the number of instances in the result.

For IR systems, the relevance of a search result is typically judged using a 5-scale rating from 0 (completely irrelevant) to 4 (completely relevant). For our study, we use

the five BCVs as the relevance scores of computer assets by mapping the BCVs of [10, 50] into [1, 5], and by assigning 0 to all other computer assets.

4.3 Baseline Methods

We designed two hypothetical baseline methods to compare our algorithms with. The first baseline produces a random ordering of the assets (hereafter denoted as $Random$). The second baseline is based on the assumption that assets used by high-rank employees are more sensitive than those used by low-rank employees. This method (denoted as *OrgRank*) produces a ranking of the assets by sorting the assets in descending order by *Rank-High*, *Rank-Middle*, and *Rank-Low* (the *Rank* features described in Table 1).

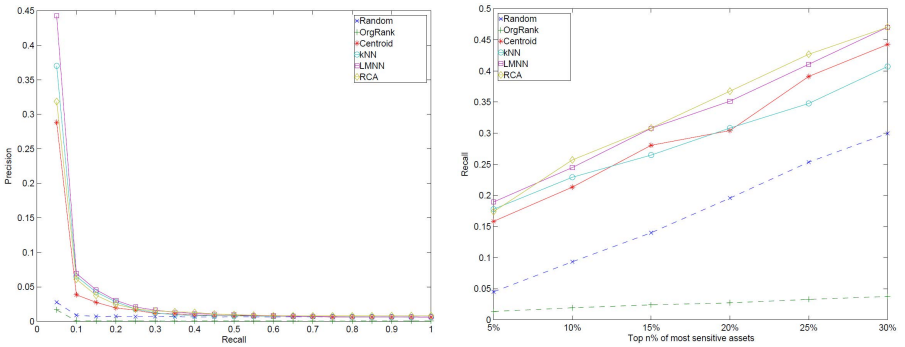
4.4 Experimental Results

In the experiments, each algorithm produces a ranked list of the computer assets, and we compare the six algorithms based on precision, recall and DCG. We set k to 3 for all kNN-based methods, and, for the clustering-based method, we generated 150 clusters for the data and k = 1 for similarity estimation. The evaluation is conducted using 5-fold cross validation methods. In a 5-fold cross validation, the ground truth data is randomly divided into 5 equally sized subgroups, and each of the subgroups is used for evaluation. At *i*-th validation ($1 \leq i \leq 5$), the *i*-th subgroup (i.e., 20% of the data) is withheld to evaluate the model's performance, and the remaining four subgroups (i.e., 80% of the data) are used to train the model. Since cross validation does random splitting of the ground truth data, we conducted 5-fold cross validation 10 times, and all the results reported here are the average performance of the 10 runs. The results of the *Random* baseline system is also the average performance from 10 random orderings.

Precision and Recall: First, we show the precision-recall curves of the algorithms. The precisions are measured at 20 different recall rates ranging from 0.05 to 1 as shown in Figure 7(a). All four algorithms yield significantly higher precision up to $recall$=0.2 than the baseline systems, with $LMNN$ outperforming the others. We notice that the precision drops rapidly as the recall increases. This is mainly due to the high skew in the class distribution in our data set (only 0.6% of samples are positive).

Next, we examine recall in more detail, as high recall is more desirable for the applications with highly imbalanced data. Figure 7(b) shows the recall levels measured at the top $n\%$ ($5\% \leq n \leq 30\%$) of the most sensitive assets in the ranked lists. As we can see, our algorithms produce much higher recall than the baseline systems, and the distance metric learning methods outperform the other algorithms across all levels of n. For instance, *RCA* achieves about 300% and 57% higher recall than *Random* at top 5% and top 30% respectively. Interestingly, *OrgRank* performs very poorly and produces much lower precision and recall than *Random*.

Discounted Cumulative Gain: Figure 8 shows the DCG values at each rank in the ranked list of the data. As noted, DCG is a better metric for applications where the relevance is judged in multi-scales. The comparison of DCG clearly show that our



(a) Precision-Recall Curve (b) Recall

Fig. 7. Comparison of the precision and recall. Figure 7(a) shows the precision at different recall levels. Figure 7(b) shows the recall measured at top $n\%$ of the ranked data.

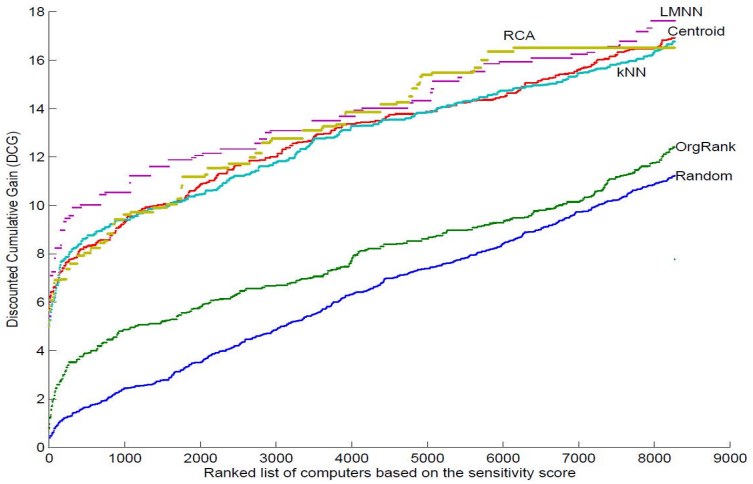


Fig. 8. The discounted cumulative gains of different algorithms. The x-axis represents the ranks of the data in descending order, i.e., x=1 represents the most sensitive computer ranked by each algorithm.

algorithms perform significantly better than the baseline methods, and $LMNN$ performs slightly better than the other algorithms. Further, our algorithms converge much more quickly achieving high DCGs early in the ranked list. This shows that our algorithms are able to assign high ranks to highly sensitive assets. We also notice that the *OrgRank* method performs better than *Random* when measured by DCG.

5 Related Work

There have been little work on automatically measuring the sensitivity (or criticality) of IT assets. A related body of work has been studied by [3,10,12,13,15]. Park *et al.* [12,13]

and Beaver *et al.* [4] proposed methods for scoring the value of the information stored in host computers using text processing and classification. While these methods are very useful for data security, they can not be applied to other types of IT assets. Further, these methods require direct access to the assets to crawl the data, thus, they are harder to apply to a large scale heterogenous environment. Beaudoin and Eng presented a method for computing the values of network assets based on the network topology, systemic dependencies among the network assets and the interfaces between the network [3]. They manually assign the initial values to some of the sources called "user services", and percolate the values from the user services back to the supporting assets using a graph mining algorithm. Sawilla and Ou presented *AssetRank*, a generalization of the PageRank algorithm, which calculates the importance of an asset to an attacker [15]. Their approach uses the dependency relationships in the attack graph and the vulnerability attributes to compute the relative importance of attacker assets rather than the importance of the asset itself. Kim and Kang [10] described a method for scoring and ranking cyber assets using a small number of hand-crafted features. They utilize three types of features – static factors (e.g., the criticality of application on the asset and value of data on the asset), static value-sensitive factors (e.g., who owns the machine) and dynamic value-sensitive factors (e.g., who is currently logged onto the machine). Crucially, their features are hard to extract automatically, and, thus, they extract the feature values in five-point scale from domain experts using a user survey.

6 Discussion and Conclusions

In this paper, we proposed algorithms for automatically scoring IT assets with a minimum of human intervention. Our algorithms provide several technical advantages that make our system more efficient, scalable, and privacy preserving than other existing methods. First, our methods do not require access to the assets or any detailed knowledge about the targets. Second, the features are very domain-independent and can be mostly extracted from access logs. Third, we apply semi-supervised machine learning approaches to minimize human efforts.

We confirmed through experiments that our algorithms perform much better than a random ordering or a simple hypothesis-based approach. Further, the performance improvement was larger when the multi-scale sensitivity values were taken into account. This indicates that our algorithms were able to retrieve assets with higher scores at higher ranks. The experiments also demonstrated that distance metric learning techniques improves the accuracy of the algorithms.

The system envisions to provide fine-grained security on high-value enterprise assets and help large enterprises manage the security risks associated with these assets. Firstly, the fine grained estimation of sensitivity values can be used to define access control policies based on the sensitivity levels. For instance, we can define access control policies granting access to assets with sensitivity levels up to a defined threshold. Another application of the dynamic computation of sensitivity values is in risk-based security methods. These methods typically rely on bounding the worst case damage caused by incorrect access control decisions. The ability to dynamically estimate the sensitivity values would make risk based methods effective and applicable in practice.

References

1. Aksoy, S., Haralick, R.M.: Feature normalization and likelihood-based similarity measures for image retrieval. Pattern Recognition Letters 22(5), 563–582 (2001)
2. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: Proceedings of International Conference on Machine Learning, ICML, pp. 11–18 (2003)
3. Beaudoin, L., Eng, P.: Asset valuation technique for network management and security. In: Proceedings of the Sixth IEEE International Conference on Data Mining Workshops, ICDMW 2006, pp. 718–721. IEEE Computer Society (2006)
4. Beaver, J.M., Patton, R.M., Potok, T.E.: An approach to the automated determination of host information value. In: IEEE Symposium on Computational Intelligence in Cyber Security, CICS, pp. 92–99. IEEE (2011)
5. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. MITRE Corporation, 1 (1973)
6. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
7. Cole, E.: Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization. Syngress (2012)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
9. Jarvelin, K., Kekalainen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (4), 422–446 (2002)
10. Kim, A., Kang, M.H.: Determining asset criticality for cyber defense. Technical Report NRL/MR/5540–11-9350, NAVAL RESEARCH LAB WASHINGTON (2011)
11. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
12. Park, Y., Gates, S.C., Teiken, W., Chari, S.N.: System for automatic estimation of data sensitivity with applications to access control and other applications. In: Proceedings of The ACM Symposium on Access Control Models and Technologies, SACMAT (2011)
13. Park, Y., Gates, S.C., Teiken, W., Cheng, P.-C.: An experimental study on the measurement of data sensitivitys. In: Proceedings of Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS, pp. 68–75 (2011)
14. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metriclearning for large margin nearest neighbor classification. In: Proceedings of the Neural Information Processing Systems Conference, NIPS (2005)
15. Sawilla, R.E., Ou, X.: Identifying critical attack assets in dependency attack graphs. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 18–34. Springer, Heidelberg (2008)
16. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal (1948)
17. Shental, N., Hertz, T., Weinshall, D., Pavel, M.: Adjustment learning and relevant component analysis. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 776–790. Springer, Heidelberg (2002)
18. Stamati-Koromina, V., Ilioudis, C., Overill, R., Georgiadis, C.K., Stamatis, D.: Insider threats in corporate environments: a case study for data leakage prevention. In: Proceedings of the Fifth Balkan Conference in Informatics, BCI 2012, pp. 271–274 (2012)
19. Voorhees, E.M.: Variations in relevance judgments and the measurement of retrieval effectiveness. In: Proceedings of the 21 st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, vol. 24, pp. 315–323 (1998)
20. Yang, L.: Distance metric learning: A comprehensive survey (2006)