DARD: Deceptive Approaches for Robust Defense Against IP Theft

Alberto Maria Mongardini, Massimo La Morgia, Sushil Jajodia, Luigi Vincenzo Mancini and Alessandro Mei

Abstract—With the rise of smart working and recent global events, the risk of cyberattacks is increasing steadily. Sometimes adversaries focus on stealing valuable data, such as intellectual property (IP): they exfiltrate a large volume of IP documents from a target company. They then identify those of their interest by leveraging automated methods. In this work, we propose the DARD (Deceptive Approaches for Robust Defense against IP Theft) system, a framework designed to deceive adversaries who rely on automatic approaches to classify exfiltrated documents. Starting from an original repository of documents, DARD automatically generates a new deceptive repository that misleads popular automatic approaches, resulting in clusters of documents that are significantly different from the actual ones. By utilizing this approach, DARD aims to hinder the accurate clustering and the identification of the topic of documents by adversaries relying on automated techniques. The paper presents four deceptive operations (Basic Shuffle, Shuffle increment, Shuffle reduction, and Change topic) that DARD leverages to create a deceptive repository. We evaluate the efficacy of our approach by considering three different types of adversaries, each possessing varying levels of knowledge and expertise. We show experimentally that the DARD system can deceive both topic modeling and document clustering techniques, including commercial tools such as Amazon Comprehend. As a result, our solution provides a robust defense mechanism against Intellectual Property (IP) theft.

Index Terms—Deceptive repository, clustering, topic modeling, adversarial setting.

1 Introduction

According to recent cybersecurity reports from sources such as Deloitte [1] and Interpol [2], there is a noticeable rise of businesses suffering cyber attacks. This upward trend can be attributed to several factors, including the growing number of employees working remotely, which has become increasingly prevalent since the onset of the COVID-19 pandemic. Additionally, the outbreak of war in Ukraine has led to increased threats of cyberattacks against Western businesses, with reported attacks against European companies in particular [3]. In 2022, there were many instances of exfiltration attacks, which involved unauthorized data extraction from targeted systems. The Cybersecurity and Infrastructure Security Agency (CISA) registered an exfiltration attack within the Defense Industrial Base organization [4]. The adversaries infiltrated the organization's information system, compromised its network, and illicitly accessed and stole the organization's sensitive data. The press also reports significant thefts of Intellectual Property (IP) almost daily. The U.S. based cloud solution provider Blackbaud suffered a data breach that lasted from February to mid-May 2020, during which cyber criminals allegedly were able to exfiltrate a huge amount of data. In October 2020, cyber-criminals stole about 1TB of employee information and company documents from the German tech firm Software AG [5]. The Australian Toll Group in 2020 was hit by cyber criminals twice in three months, with an alleged data loss of over 200GB of corporate data [6]. In some cases, months might pass by before a successful compromise of an enterprise network is discovered. According to the 2021 Verizon's report [7], 20% of data breaches that occurred in 2020 were discovered several months after the attack, such as the SolarWinds cyber attack [8] that remained undetected for 9 months. Adversaries interested in a company's information could exploit the interval after intrusion and before detection to exfiltrate large amounts of IP documents from the company.

Given the vast amount of exfiltrated data, adversaries often employ a strategy of analyzing their contents to identify specific documents related to their interests. Using human domain experts is one option but it is a time-consuming activity for adversaries. Consequently, as a first step, they typically select documents related to certain topics of interest through an automated approach to focus their in-depth analysis only on a few documents. In the final phase, human domain experts come into play to assess the value of the few selected documents in terms of IP and the presence of innovative content. Our adversary model encompasses a broad range of adversaries, including Wikileaks users who possess the capability to employ topic modeling and clustering techniques. These techniques allow them to identify specific documents of interest within the vast collection published by Wikileaks, such as those containing highly sensitive information.

This paper aims to hinder the first phase of the attack. To achieve this, it proposes the DARD (Deceptive Approaches for Robust Defense against IP Theft) system. This system is designed to ensure that adversaries fail in their attempts to analyze a large repository of exfiltrated documents using

- *S. Jajodia is with the Center for Secure Information Systems at George Mason University, Fairfax, VA 22030 USA. E-mail: jajodia@gmu.edu*
- *A.M. Mongardini, M. La Morgia, L.V. Mancini and A. Mei are with the Computer Science Department at "Sapienza" University of Rome, 00185 Rome, Italy. E-mail: mongardini@di.uniroma1.it, lamorgia@di.uniroma1.it, mancini@di.uniroma1.it, mei@di.uniroma1.it*

automated tools. As a result, adversaries will be left with the only expensive option of using human domain experts to examine the entire repository thoroughly. Starting from an original repository $\mathcal{R}$ of documents, DARD automatically generates a deceptive repository $\mathcal{R}'$. When an automatic clustering technique analyzes $\mathcal{R}'$, it produces a set of documents clusters that is far away, in terms of the number of clusters and of individual documents grouped in each cluster, from what is actually present in $\mathcal{R}$. Specifically, to generate such a deceptive repository, this work presents four deceptive operations. A defender can use these deceptive operations to implement a defense strategy against IP thefts, hindering the use of both topic modeling and document clustering techniques. Regarding topic modeling, defenders can build a deceptive repository $\mathcal{R}'$ that, when automatically parsed, presents topics of no interest to the adversaries instead of the original potentially sensitive topics contained in the documents of $\mathcal{R}$. In this case, the adversaries have the following options: (1) trust the result found by the automated process; (2) attempt to reverse the deceptive operations, obtaining poor results, as shown in this paper; or (3) use human experts to identify documents of interest within $\mathcal{R}'$. In the case of document clustering, deceptive operations can build a new deceptive repository $\mathcal{R}'$ in such a way that clustering techniques return clusters in which sensitive documents are distributed. Thus, adversaries interested in retrieving these sensitive documents cannot exclude any cluster from their analysis; if they disregard certain clusters during subsequent analysis, they risk losing identification of sensitive documents contained in the excluded clusters and still require human effort to retrieve the sensitive documents in the remaining clusters. A defender can combine the two strategies to deceive topic modeling and document clustering approaches, achieving higher levels of defense against IP thefts and effectively slowing down the adversaries and requiring increased effort on their part.

From the point of view of authorized users, DARD is completely transparent. Indeed, it is possible to store the mapping of the replaced keywords in the repository's documents to reconstruct the original version. However, the keywords mapping, the most sensitive data of the DARD system, can not be stored on the conventional file system, risking being exfiltrated together with the other repository documents. To address this concern, a secure application must be developed for file restoration, utilizing a Secure Enclave Solution [9]. By adopting this approach, the mapping can be securely stored within the Secure Memory of the Secure Enclave. Performing the restore operation on the Secure CPU ensures that the keywords mapping remains isolated and safeguarded. The details of the solution are not addressed as they are out of the scope of this paper.

The contributions of this work include:

- **Deceptive operations:** We designed and implemented four deceptive operations (Basic Shuffle, Shuffle increment, Shuffle reduction, and Change topic) that select and replace some keywords present in the documents of the repository $\mathcal{R}$ with deceptive keywords. These operations can be used to create a deceptive repository $\mathcal{R}'$ that, when automatically parsed, results in a different number of clusters than those in the repository $\mathcal{R}$ and produces new clusters containing documents initially belonging to different topics of $\mathcal{R}$.
- **Extensive experimentation:** The deceptive operations have been applied to a repository made of real papers collected through the Arxiv APIs. We evaluate the performance of three kinds of adversaries on an experimental repository and show that the adversaries cluster the documents as planned by the defender.
- **Topic modeling and commercial tool evaluation:** We evaluate the possibility of deceiving the adversaries on the actual topics covered within a deceptive repository. We find that the first 10 keywords by relevance retrieved by topic modeling algorithms in the deceptive repository are all deceptive keywords. This finding indicates that defenders can manipulate the topics retrieved by adversaries, presenting them with believable yet fake topics. Furthermore, we test the effectiveness of DARD against adversaries using commercial tools like Amazon Comprehend [10] and find that the adversaries were only able to retrieve deceptive keywords. This result underlines the effectiveness of the DARD system, even in the face of adversaries using commercial tools.

2 Analysis of an Exfiltrated Repository

Assuming that adversaries have managed to exfiltrate a company's original repository, the purpose of this section is to show an example of how such adversaries could automatically infer the topics covered by each document in the exfiltrated repository and then select only the documents they are interested in. For simplicity, here we assume that the victim company has not adopted deceptive techniques in document production, and the adversaries are not aware of any of the topics covered by the documents of the repository. More powerful attack models will be defined in Sec. 4 and evaluated in the experiments in Sec. 5. This section assumes that the adversaries will follow the methodology defined in Sec. 2.2 since it represents the classical approach for document clustering and topic modeling tasks. Indeed, this pipeline is also used as a benchmark by other important proposals for new document clustering and topic modeling techniques described in the literature [11], [12].

2.1 The Repository

The exfiltrated repository presented in this subsection is also used in the experiments in Sec. 3, Sec. 4, and Sec. 5. This repository is a collection of 450 scientific papers, evenly divided into three different topics of computer science: Artificial Intelligence (AI), Database (DB), and Cryptography and Security (CR). The repository contains papers retrieved from ArXiv [13], an open-access archive for scholarly articles. ArXiv provides APIs[1] that allow users to retrieve documents specifying the domain (Computer Science), and a domain-related field (namely: Artificial Intelligence, Database, and Cryptography and Security). The documents in the repository are in Portable Document Format (PDF) and contain an average of 7,826 words each. The smallest document has 1,227 words, while the largest one 57,169. In the following, we refer to this repository as $\mathcal{R}_d$.

1. http://export.arxiv.org/api/query?search_query=query

2.2 Clustering the Documents and Retrieving Topics

Since adversaries know neither the exact number of clusters nor the topics covered by the repository, they want to discover both automatically and then focus their in-depth analysis only on documents related to topics of their interest. In the first automatic phase, the adversaries can use document clustering and topic modeling techniques. Document clustering and topic modeling are two data mining techniques used to automatically organize and retrieve information from unorganized collections of text documents. The goal of document clustering [14] is to organize a repository of documents into groups of similar documents. Instead, topic modeling techniques [15], [16] aim to build a latent semantic representation of the documents, detecting keywords that describe the subject dealt with by the documents. In particular, the latent semantic representation of a set of documents is called Topic. In the following sections, we describe the steps the adversaries should perform on the exfiltrated repository to retrieve the documents of their interest.

2.2.1 Text pre-processing and feature extraction

Before starting the analysis, the text has to be normalized and cleaned of all the elements that do not provide information about the topic (*e.g.,* numbers). To this end, the pre-processing phase is a key component of every text classification tool [17]. Hence, the adversaries perform on the documents standard pre-processing operations such as tokenization, stemming, normalization of the upper and lowercase, and deletion of number and symbol characters. Once the documents in $\mathcal{R}_d$ have been normalized, the adversaries proceed with the feature extraction. In text analysis, a document and its content are usually represented as a vector, where each position of the vector represents a term (*i.e., one or more consecutive words in the document*) with an associated weight.

In the feature extraction step, the adversaries extract the terms that occurred within the documents and assign them a weight through TF-IDF. The TF-IDF (Term Frequency-Inverse Document Frequency) [18] is a function that assigns a weight to a term in relation to a document. The greater the weight, the greater the importance of the term for the document. The idea behind the TF-IDF is to give more importance to terms that occurred within a document but are generally not frequent within the document repository. Therefore, terms that are characteristic only of a group of documents are considered significant.

By calculating the TF-IDF for each term, the adversaries obtain a TF-IDF matrix as the one in Fig. 1. Each column represents a document with a Document Vector containing the weights of the terms for that document. Instead, each row indicates a term with a Word Vector containing the weights of that term for each document in the repository.

2.2.2 Document clustering

At this point, the adversaries are ready to group the documents according to the features extracted in the previous step. First, they need to estimate the correct number of clusters in the repository, which is one of the major challenges in cluster analysis [19]. The most popular approach proposed in the literature is internal clustering [20]. This method typically involves three steps: (1) apply to the dataset several clustering algorithms using different combinations of parameters, (2) compute the corresponding internal validation score for each obtained partition, and (3) detect the optimal number of clusters by choosing the partition with the best internal validation score. There are over thirty typologies of internal clustering evaluation [20] that can be used in steps 2 and 3 described above. Among the most widely adopted, we consider the Silhouette Coefficient [21], the Calinski-Harabasz Index [22], and the Davies-Bouldin Index [23].

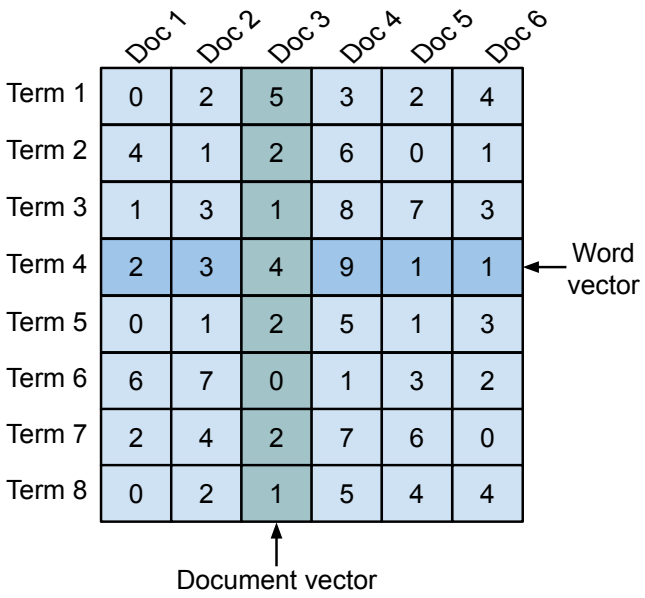


Figure 1. Matrix representation of documents: each column identifies a document, while each row represents a term. A Document Vector is the column associated with a document and contains the weights of the terms for that document. A Word Vector is a row related to a term and contains its weights for each document in the repository.

Thus, we assume adversaries use these internal validation scores to infer the number of clusters K in the repository. Once detected the number of K clusters (three in this case), the adversaries rely on the TF-IDF weighting scheme and K-means to cluster the documents. K-means [24] is a popular clustering algorithm that takes as input a number k of expected clusters and finds a k-partition such that the squared error between the empirical mean of a cluster (centroid) and the points in the cluster is minimized. Applying K-means on the document vectors of the TF-IDF weights, the adversaries obtain clusters that correctly group the exfiltrated documents according to their topics, as shown by their projection in Fig. 2. To visualize the clusters obtained by the adversaries in this section and the next ones, we performed a dimensionality reduction on the TF-IDF weights by applying the t-distributed stochastic neighbor embedding (t-SNE) [25], an algorithm that allows visualizing high dimensional data in a low dimensional space. The color of each item in the figure represents the original topic of the document, whereas the shape of the item represents the cluster the document belongs to. As we can see, the clustering result is remarkably similar to the real one. Therefore, the adversaries are able to correctly distinguish documents belonging to the three different topics.

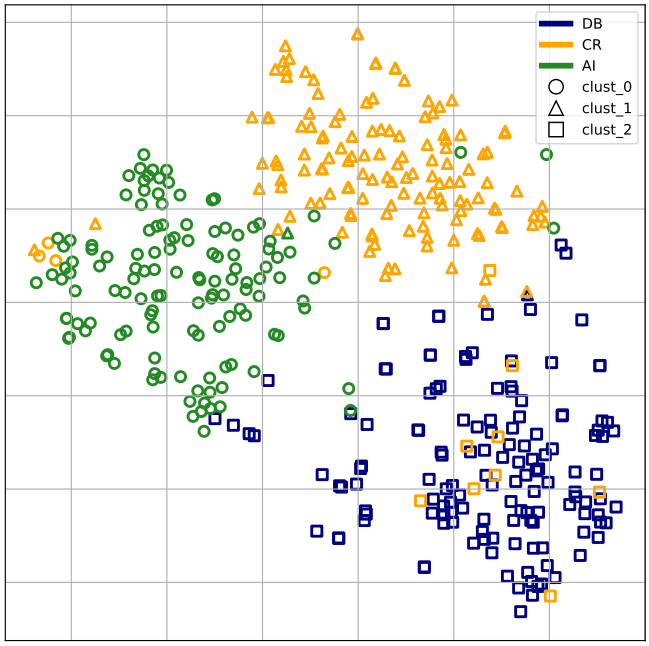


Figure 2. Projection of the clusters obtained through K-means from the original repository composed of documents related to Artificial Intelligence(AI), Database (DB), and Cryptography and Security (CR).

2.2.3 Topic modeling

In the previous section, adversaries cluster the documents according to the TF-IDF weights. Here, they want to infer the topic covered by each of the K clusters by retrieving the terms that describe each specific cluster. To this end, adversaries leverage Latent Dirichlet Allocation (LDA) [16], one of the most used topic modeling algorithms, that provides as output for each cluster a list of terms ordered by their relevance to the topic. We define as M $keywords$, the first M terms of the output list representing the topic of a given cluster. Tab. 1 shows the top 10 keywords extracted from each cluster in the repository $\mathcal{R}_d$ by the adversaries. The latter might infer from these keywords the three topics covered in $\mathcal{R}_d$, namely: Artificial Intelligence, Database systems, and Cryptography and Security. After this step, adversaries focus their analysis only on documents addressing specific topics in which they are interested. The paper does not cover the second phase since our techniques aim to deceive the first phase, of which results also influence the second one. Due to the proposed deceptive operations, the documents covering the topic of interest for the adversaries will be scattered throughout all clusters. As a result, adversaries can not focus on just one cluster but on all of them.

3 Operations

3.1 Replacement operations of terms

This subsection describes the idea behind the deceptive operations, or the term-replacement operations, illustrating the relationship between the term-replacement operations and the resulting changes in the TF-IDF matrix calculated on the sets of documents in $\mathcal{R}$. This paper refers to *keyword* k as the term to be replaced and *deceptive term* dk as a new term, not contained in $\mathcal{R}$, that replaces one or more keywords k.

Table 1
Top 10 keywords extracted from each cluster using LDA.

AI	DB	CR
learn	query	scheme
plan	node	protocol
agent	object	security
decision	logic	message
policy	xml	signature
action	attribute	public
network	tree	attack
constraint	semantic	service
strategy	update	random
intelligence	predicate	bit

To explain the effect of a term-replacement operation, we rely on the concepts of centroid and distance between centroids. Let T be the set of terms contained in the documents of $\mathcal{R}$, S be a set of documents in $\mathcal{R}$, and consider a sub-matrix of the TF-IDF weights of $\mathcal{R}$ that contains only the columns representing the documents in S. We define as the centroid of S the vector that contains the element-by-element average of the rows in this submatrix. Let S_1 and S_2 be two sets of documents, the distance between S_1 and S_2, denoted $d(S_1, S_2)$, is the Euclidean distance between their centroids.

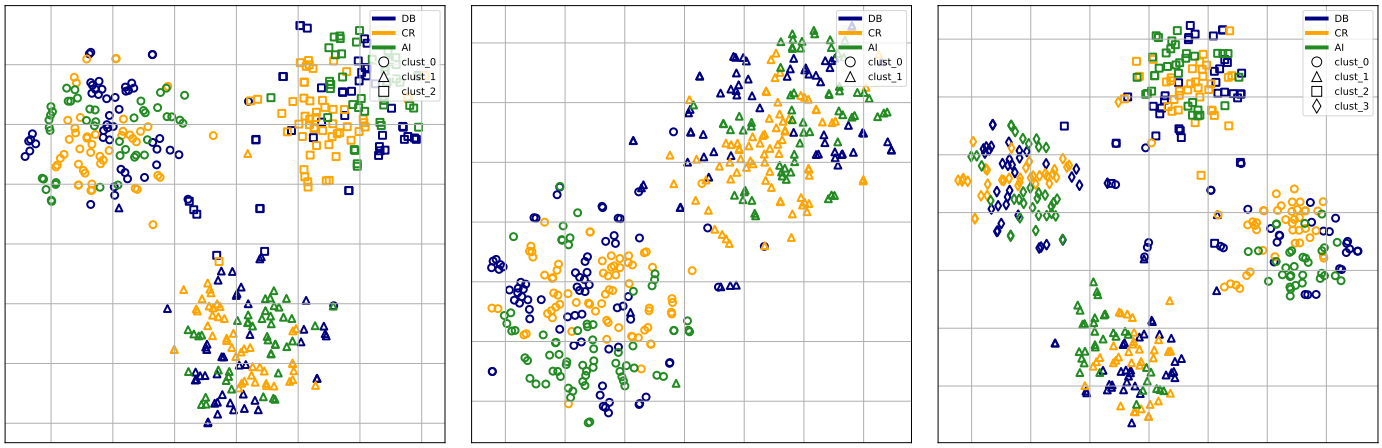
Given a repository of documents $\mathcal{R}$, this paper considers the following four strategies to replace keywords at the level of the documents set, where all occurrences of a certain keyword are replaced inside all the documents contained in a specific set.

(i) **1-to-1 replacement:** Consider a repository $\mathcal{R}$ partitioned in $n > 1$ sets of documents, such that $\mathcal{R} = \{S_1 \cup \cdots \cup S_n\}$. Let k be a keyword for the repository $\mathcal{R}$, and dk a deceptive term. The 1-to-1 replacement operation changes all the occurrences of k in all the documents of the repository $\mathcal{R}$, with the deceptive term dk.

After the 1-to-1 replacement, the deceptive term dk appears in the same documents and with the same frequencies of k. Thus, there is a new row in the TF-IDF matrix of $\mathcal{R}$ for dk, which has precisely the same weights as k. Moreover, since the keyword k no longer appears in the documents of $\mathcal{R}$, the row in the TF-IDF matrix associated with k disappears as well. Hence, the 1-to-1 replacement does not alter the relative position among the centroids of all the sets of documents S_i. In addition, since k was a keyword for $\mathcal{R}$, also dk will be a keyword for the deceptive repository $\mathcal{R}'$.

Of course, it is possible to perform several times the 1-to-1 replacement operation that, for brevity, in the following, we call **m-multiple 1-to-1 replacement**. In particular, let M be the maximum number of keywords computed on the repository $\mathcal{R}$, and $m < M$. A m-multiple 1-to-1 replacement performs m times a 1-to-1 replacement on $\mathcal{R}$ using different keywords, says $\{k_1, \ldots, k_m\}$, and different deceptive terms $\{dk_1, \ldots, dk_m\}$. In particular, each j^{th} execution of the m-multiple 1-to-1 operation replaces a keyword k_j with a deceptive term dk_j in all the documents of $\mathcal{R}$. Therefore, dk_j represents the deceptive term with which the m-multiple 1-to-1 operation replaces k_j in the documents of the repository $\mathcal{R}$ at the j^{th} execution of the 1-to-1 replacement.

(ii) **1-to-N replacement:** Consider a repository $\mathcal{R}$ parti-



(a) Basic Shuffle. (b) Shuffle reduction. (c) Shuffle increment.

Figure 3. Fig. 3(a) shows the projection of the repository (composed by AI, DB, and CR) modified by the Basic Shuffle operation. Fig. 3(b) and Fig. 3(c) show, respectively, the projections of the repository obtained by applying Shuffle reduction and Shuffle increment.

tioned in $n > 1$ sets of documents, such that $\mathcal{R} = \{S_1 \cup \cdots \cup S_n\}$. Let k be a keyword for $\mathcal{R}$, and $\{dk_1, \ldots, dk_n\}$ be a set of deceptive terms. The 1-to-N operation replaces all the occurrences of k with a different deceptive term dk_i in each document of S_i. Thus, for every i, after the 1-to-N replacement, the term dk_i appears in the documents of S_i instead of the keyword k and dk_i does not appear in the documents of $\mathcal{R} \setminus S_i$.

Note that after the 1-to-N replacement, the keyword k no longer appears in the documents of $\mathcal{R}$ and, consequently, in its TF-IDF matrix. At the same time, after the replacement, in the TF-IDF matrix n new rows appear, one for each deceptive keyword dk_i. Finally, since the deceptive term dk_i appears only in the documents of the set S_i, its weight will be greater than zero in the documents that belong to S_i and zero for the others. Hence, the centroid of each set S_i tends to move away from the centroids of the sets S_l for each i, l with $i, l \in \{1, \ldots, n\}$ and $i \neq l$. In particular, the higher the rank of keyword k is, the more the centroids tend to move away from each other.

It is possible to perform several times the 1-to-N replacement. We call in the following this operation **m-multiple 1-to-N replacement**. This operation replaces, in all the repository $\mathcal{R}$, m keywords with $m \times n$ deceptive keywords. Let M be the maximum number of keywords computed on the repository $\mathcal{R}$, and let $m \leq M$. A m-multiple 1-to-N replacement performs m times a 1-to-N replacement on all the $S_i \in \{S_1, \ldots, S_n\}$ replacing m different keywords, $\{k_1, \ldots, k_m\}$. In particular, each j^{th} execution of the 1-to-N operation replaces a keyword k_j with n deceptive keywords $dk_{1,j}, \ldots, dk_{n,j}$ in each of the subsets S_i. Thus, $dk_{i,j}$ represents the deceptive term with which the m-multiple 1-to-N operation replaces k_j in the set S_i during the j^{th} execution of the 1-to-N replacement operation.

(iii) **N-to-1 replacement:** Consider a repository $\mathcal{R}$ partitioned in $n > 1$ sets of documents, such that $\mathcal{R} = \{S_1 \cup \cdots \cup S_n\}$. Let the keywords $\{k_1, \ldots, k_n\}$ be a set of terms, such that k_i is a keyword for the set of documents S_i, while k_i is not a keyword for S_l, with $i, l \in \{1, \ldots, n\}$ and $i \neq l$. The N-to-1 operation replaces in every set of documents S_i all the occurrences of the keyword k_i with the deceptive keyword dk. Thus, after the N-to-1 replacement, the deceptive term dk appears in the documents of S_i instead of k_i, for every i. In the N-to-1 replacement, the goal is to bring closer the centroids of the set of documents S_i, replacing N different keywords with the same deceptive keyword dk. Differently, the 1-to-N replacement aims to move away the centroids of the set of documents S_i, replacing a unique keyword with N different deceptive keywords. Following the N-to-1 replacement, the TF-IDF weights of all the keywords k_i drop to zero for the documents in S_i, whereas a new row associated to the deceptive keyword dk appears in the TF-IDF matrix of $\mathcal{R}$. The TF-IDF weight of dk is greater than zero for all the documents in S_i that previously contained the keyword k_i. Hence, the centroid of each set S_i tends to get closer to the centroid of S_l, for every i, l. In particular, the higher the rank of the keywords k_i is, the more the centroids tend to get closer to each other.

It is possible to perform several times the N-to-1 replacement. We call this operation in the following **m-multiple N-to-1 replacement**. This operation replaces in all the repository $\mathcal{R}$, $m \times n$ keywords with m deceptive keywords. Let M the biggest number of keywords such that all the $S_i \in \{S_1, \ldots, S_n\}$ have at least M keywords, and let $m \leq M$. A m-multiple N-to-1 replacement performs m times a N-to-1 replacement on all the S_i, replacing n different keywords, say $\{k_{1,j}, \ldots, k_{n,j}\}$, for each execution j, with $j \in \{1, \ldots, m\}$. In particular, each j^{th} execution of the N-to-1 operation replaces the keyword $k_{i,j}$ in the set of documents S_i, for every i, with the deceptive term dk_j.

(iv) **N-to-N replacement:** Consider a repository $\mathcal{R}$ partitioned in $n > 1$ sets of documents, such that $\mathcal{R} = \{S_1 \cup \cdots \cup S_n\}$. Let $\{k_1, \ldots, k_n\}$ be a set of keywords, such that each k_i is a keyword for one set of documents S_i, while k_i is not a keyword for S_l, with $i, l \in \{1, \ldots, n\}$ and $i \neq l$, and $dk_1, \ldots, dk_n$ be a set of deceptive terms. The N-to-N operation replaces in every set of documents S_i all the occurrences of the keyword k_i with the deceptive keyword dk_i. Thus, after the N-to-N replacement, the deceptive term dk_i appears in the documents of S_i instead of k_i, for every

i.

The N-to-N replacement is similar to a 1-to-1 replacement applied to a single set of documents S_i, instead of to all the repository $\mathcal{R}$. After an N-to-N replacement, the deceptive term dk_i will have in the TF-IDF matrix of $\mathcal{R}$, the same TF-IDF weights of the keyword k_i. Hence, since k_i was a keyword for the set S_i, also dk_i will be a keyword after the N-to-N replacement.

3.2 Shuffle Clusters

Starting from a repository $\mathcal{R}$ where each cluster is made of documents that belong to a single topic, the Shuffle operation builds a deceptive repository $\mathcal{R}'$ in which some or all the clusters contain documents of different topics. Thus, the adversaries cannot precisely cluster the documents of $\mathcal{R}'$ according to the original topics of $\mathcal{R}$. The Shuffle operation, given a set L made of l different clusters, partitions each cluster in L into p subsets of documents and mixes these subsets among themselves, building a new set of clusters L'. In particular, each new cluster in L' is made of l subsets of documents each of which belongs to a different original cluster of the set L. Since each new cluster contains exactly one subset of each original cluster, the relationship between p and l determines the number of resulting new clusters in L'.

Definition 3.1. Shuffle ($\mathcal{R}$, C_1,...,C_l)$\Rightarrow \mathcal{R}'$.
*Given a Repository R composed of $n > 1$ clusters, let CR be the set of clusters in R. Consider the clusters $L = \{C_1, \ldots, C_l\}$ in CR, with $1 < l \leq n$. The Shuffle operation partitions each cluster C_i of L into p subsets of documents, with $l - 1 \leq p \leq l + 1$ and $l - 1 \geq 2$, such that $C_i = s_{i,1} \cup \cdots \cup s_{i,p}$, where $s_{i,j}$ represents the subset j of the cluster C_i. The Shuffle operation replaces some keywords in L in such a way as to form p new clusters C'_i. Each new cluster C'_j is composed of l subsets of documents $s_{i,j}$ in such a way that $C'_j = s_{1,j} \cup \cdots \cup s_{l,j}$ with $j \in \{1, \ldots, p\}$. Depending on the relationship between p and l, the effect of the Shuffle operation on the repository $\mathcal{R}'$ is different. In particular, there are three possible variants: The **Basic Shuffle** in which the number of partitions p is equal to l and thus the number of clusters in the repository $\mathcal{R}'$ is n, the **Shuffle Increment** where the number of partition p is equal to $l + 1$; in this case $\mathcal{R}'$ contains $n + 1$ clusters, and the **Shuffle Reduction** in which the number of partitions p is equal to $l - 1$, and the repository $\mathcal{R}'$ contains $n - 1$ clusters. After one of the three Shuffle operations, the adversaries that search for $n - l + p$ clusters in the repository $\mathcal{R}'$ will find the following set of clusters: $\mathcal{CR}' = (\mathcal{CR} \setminus L) \cup L'$, where $L' = \{C'_1, \ldots, C'_p\}$.*

In our implementation, the Shuffle operations first compute the keywords of all the clusters C_i in L. Then, it partitions the documents of each C_i into p subsets of documents such that $C_i = s_{i,1} \cup \cdots \cup s_{i,p}$, with $s_{i,j}$ that represents the subset j^{th} of the cluster C_i. The Shuffle operations select subsets of documents to compose the new clusters $\{C'_1, \ldots, C'_p\}$, such that each new cluster C'_h, with $h \in \{1, \ldots, p\}$, is made of l subsets of documents, one subset from each C_i. For the sake of simplicity, we assume that the Shuffle operations select the subsets of documents that compose the new cluster C'_h picking from each cluster C_i the h^{th} subset, thus $C'_h = s_{1,h} \cup \cdots \cup s_{l,h}$. Finally, for each cluster C'_h, the Shuffle operations perform an l-to-1 replacement, which overall sums up to p times a l-to-1 replacement operations. An l-to-1 operation replaces l different keywords, each one computed on each C_i, $i \in \{1, \ldots, l\}$ with the deceptive term dk_h (see Sec. 3.4 for details about the keywords selection). In particular, let k_i be a keyword of C_i. The l-to-1 operation replaces all the occurrences of k_i in the subset $s_{i,h}$ with the deceptive term dk_h, with $h \in \{1, \ldots, p\}$ for every $i \in \{1, \ldots, l\}$. Note that each of the h execution of the l-to-1 operation uses a different deceptive term dk_h.

A single l-to-1 replacement could not be enough to bring the centroids of all the subset $\{s_{1,h}, \ldots, s_{l,h}\}$ sufficiently closer to form the new cluster C'_h, for every $h \in 1, \ldots, p$. Therefore, the Shuffle operation has to perform m-multiple l-to-1 replacements such that the following equation is satisfied:

$$d(s_{i,h}, (C'_h \setminus s_{i,h})) < \underset{\forall i \in \{1,\ldots,l\} \wedge \forall h \in \{1,\ldots,p\}}{d(s_{i,h}, (C_i \setminus s_{i,h}))} \quad (1)$$

The formula verifies that after each iteration of l-to-1 replacement, the centroid of $s_{i,h}$ is closer to the centroid of $C'_h \setminus s_{i,h}$ (left term of the formula) than to the one of $C_i \setminus s_{i,h}$ (right term). In this way, each pair ($C'_h \setminus s_{i,h}$, $s_{i,h}$) is close enough to build the new cluster C'_h, and the subsets of C_i will not cluster together.

Fig. 3(a) 3(b) 3(c) show the deceptive repository $\mathcal{R}'_d$ after we applied on the repository $\mathcal{R}_d$ the Basic Shuffle (Fig. 3(a)), the Shuffle Increment (Fig. 3(b)) and the Shuffle Reduction (Fig. 3(c)). To perform the three operations, we insert into the set L all the clusters of the repository $\mathcal{R}_d$ and set the value of p as 2, 3, 4 respectively for the Shuffle Reduction, the Basic Shuffle, and the Shuffle Increment. After the operations, each cluster (denoted in the figures by the circle, square, diamond, and triangle markers) is made of a mixture of topics (green, blue, and yellow markers).

3.3 Change Topic of the cluster

The Change Topic operation aims to change the original topic of a cluster of documents C_t in $\mathcal{R}'$. The Change Topic operation builds a repository $\mathcal{R}'$ replacing several keywords of C_t with a set of deceptive terms $\{dk_1, \ldots, dk_l\}$, in such a way that topic modeling performed on $\mathcal{R}'$ returns a topic that depends on the deceptive terms, and such a topic can be different from the original one.

Definition 3.2. Change Topic($\mathcal{R}$,C_t,$\{dk_1, \ldots, dk_l\}$)$\Rightarrow \mathcal{R}'$.
Given a repository R that contains $n > 1$ clusters, a target cluster C_t in $\mathcal{R}$ and a set of l deceptive terms $\{dk_1, \ldots, dk_l\}$, the Change Topic operation replaces l keywords with l deceptive terms, using one different deceptive term for each different keyword. At the end of the operation, the adversaries that perform topic modeling on the repository $\mathcal{R}'$ will find for the cluster C_t the following keywords $\{dk_1, \ldots, dk_l\}$.

In our implementation, the Change Topic operation computes the keywords of C_t and ranks them by their TF-IDF weight. Let $\{k_1, \ldots, k_l\}$ be the first l keywords of C_t in the rank. The Change Topic operation performs a 1-to-1 replacement on the documents of C_t, replacing all the occurrences of the keyword k_i with the deceptive term dk_i, for every $i \in \{1, \ldots, l\}$. Overall, the Change Topic operation

performs l times a 1-to-1 replacement on each document of C_t.

After the operation, the deceptive term dk_i is a keyword for the cluster C_t in $\mathcal{R}'$. Indeed, dk_i has the same TF-IDF weight in $\mathcal{R}'$ as k_i has in $\mathcal{R}$. Thus, since k_i is a keyword for C_t in $\mathcal{R}$, dk_i is a keyword for C_t in $\mathcal{R}'$ as well. Moreover, since k_i and dk_i have the same weight in the TF-IDF of $\mathcal{R}$ and $\mathcal{R}'$, respectively, the centroid of C_t is the same both in $\mathcal{R}$ and in $\mathcal{R}'$.

3.4 Observations on the keywords and the selection of the documents

This section describes some of the possible approaches to partition a cluster C of documents of $\mathcal{R}$ into subsets suitable to be used by the deceptive operations described in the previous sections. In addition, we present the criteria we used to select the keywords to be replaced with the deceptive terms. When partitioning a cluster C to apply one of the deceptive operations, there are two main aspects to face: the number of documents each partition should contain and which documents of C should be grouped in the same partition.

The number of documents each partition is made is a crucial parameter to decrease the purity [26] of the resulting repository $\mathcal{R}'$. Purity is an external evaluation metric that assesses the quality of given clusters by indicating the percentage of the total number of correctly classified objects (documents). For instance, in the Shuffle operation, creating partitions with roughly equal numbers of documents leads to creating new clusters in $\mathcal{R}'$ with a purity roughly equal to zero, which guarantees the greatest possible deception. In the following experiments, all the clusters in our test repository $\mathcal{R}_t$ have the same number of documents. Hence, given the observations above, the best strategy in our case is to create the partitions used in each deceptive operation with the same number of documents. The second aspect to face is which documents of C should be placed in the same partitions. A trivial approach is to partition the documents of a cluster C in subsets of documents $\{s_1, \ldots, s_l\}$ through a random selection of the documents in C. This approach likely leads to group into the same subset documents uniformly spread among the cluster C, with the centroid of each subset s_i near the centroid of C and thus close to each other. However, the more the centroids of the subsets are close to each other, the more keywords the cluster operations need to replace in order to push the centroids away among them (See Tab. 2). A better choice is to partition the documents so that the centroids of the subsets s_i result far away among them. An approach to generate such subsets s_i is to leverage a clustering algorithm, such as K-means. Since standard K-means may generate partitions with an unbalanced number of documents (*e.g.*, a partition with most of the documents and others with very few documents), we used the constrained version of k-means [27], which ensures that each partition has a roughly equal number of documents.

For what concern the selection of the keywords to be replaced with the deceptive terms, we select the keywords by their TF-IDF weights in descending order. This approach minimizes the number of deceptive keywords to be replaced to accomplish any of the cluster operations. Indeed, the effectiveness of the 1-to-N replacement and the N-to-1 replacement in pushing away or bringing close among them the centroids of partitions $\{s_1, \ldots, s_l\}$ is proportional to the TF-IDF weight of the replaced keywords (as discussed in Sec. 3.1).

Table 2
Number of terms to involve for deceptive operations selecting subsets of documents randomly or by grouping similar ones and replacing random terms.

	B. Shuffle	Shuffled In.	Shuffle Red.
Random selection	8,6	10,2	7,4
Constrained K-means	8	10	6
K-means + random terms	71,6	86,6	41,9

To better understand how the documents and the keywords selection affect the number of keywords needed to perform a deceptive operation, we evaluated the deceptive operations in the following three settings: partitions created with a random selection of the documents and keywords selected by TF-IDF weight; partitions created leveraging the constrained K-means and keywords selected by TF-IDF weight; and partitions created leveraging the constrained K-means and keywords selected randomly. For each of the above settings (except the constrained k-means version), we repeat the experiment 10 times and compute the average number of keywords needed to perform the cluster operations. Tab. 2 shows the results of this experiment. The best combination to minimize the number of keywords replaced is the one based on the constrained k-means and the keyword selected by TF-IDF weight (constrained k-means in the table). This is in line with our previous observations in this section. Randomly selecting the keywords increases the number of keywords drastically to be replaced. For example, in the case of the Shuffle Increment operation, the number of keyword replacements increases from about 10 to more than 80. Building the partitions by randomly selecting the documents requires a few more replacements than partitioning the documents via k-means.

3.5 Deceiving the number of topics

An accurate clustering result requires the right estimation of the number of clusters in a repository of documents. By our assumption, the adversaries that exfiltrated the repository $\mathcal{R}'$ do not know the number of clusters that the repository contains. Thus, they have to estimate the number of clusters in the repository $\mathcal{R}'$ through internal cluster indices (see Section 2.2.2). This section aims to illustrate our proposed technique to deceive adversaries in such estimation, making them believe that the repository $\mathcal{R}'$ contains a given numerical value K_d for the number of clusters which is deceptive.

In the literature, there are several internal cluster indices that the adversaries can leverage to estimate the number of clusters of $\mathcal{R}'$. The main idea behind these indices is to evaluate the compactness (how close are the items of the same cluster), the separation (how distant are the clusters from each other), or a combination of them. Every index evaluates these criteria accordingly with the different

evaluation methodologies they use (*e.g.*, average distance, minimum distance, the sum of square error).

However, it is not possible to know in advance the validation indices the adversaries will use. Therefore to deceive adversaries, the clusters in $\mathcal{R}'$ have to be enough compact and separated so that for all the indices, or at least most of them, the resulting number of clusters is K_d. In terms of our cluster operations, we propose to redefine the stopping criteria for the number of term-replacement to be performed (recall that both the 1-to-N and the N-to-1 operation contribute to pushing away or bringing close clusters among them) so that their number is greater than or equal to those defined in Eq. 1.

To evaluate the minimum number of term-replacement to deceive adversaries on the estimation of the number of clusters in the repository $\mathcal{R}'$, we introduce the following function:

$$f(\mathcal{R}, Op, K_d, K_{max}, T_{max}, \mathcal{S}_{ivi}) \tag{2}$$

The function f, given a repository $\mathcal{R}$, a cluster operation Op, and a Set of internal validation indices $\mathcal{S}_{ivi}$, computes the minimum number of term-replacement operations such that all the indices in $\mathcal{S}_{ivi}$ evaluate K_d as the estimated number of clusters. Since it is impractical to evaluate all the possible numbers of clusters, we reduce the search space of the number of clusters from 2 up to K_{max}. Finally, T_{max} represents the maximum number of term-replacement operations we are willing to perform. It is important to set T_{max} because internal validation indices, depending on how they evaluate the compactness and the separation, could cause an unlimited number of term-replacement operations when evaluating particular data distribution (*e.g.*, presence of outliers, skewed distribution) [28].

Computing f on the repository $\mathcal{R}_d$ for the Basic Shuffle, the Shuffle Increment, and the Shuffle Reduction operations, we find that to deceive adversaries about the number of clusters contained in $\mathcal{R}'_d$, for the Basic Shuffle operation, we have to perform 41 term-replacement instead of 8, 38 replacements for the Shuffle Increment instead of 10, while 18 for the Shuffle Reduction. For the above-mentioned results, we compute the function f evaluating the following indices: the Silhouette Coefficient (SIL), the Calinsky-Harabasz index (CH), and the Davies-Bouldin Index (DB), and we set as 100 the maximum number of replacement operation T_{max}. For the Shuffle operation, we set K_{max} as 9 since we divided the repository into 9 partitions, whereas K_d as 3 because we aim to make the adversaries believe that $\mathcal{R}'_d$ contains 3 clusters. For the Shuffle Increment operation, we set K_{max} as 12, and K_d as 4. Finally, for the Shuffle Reduction operation, we set for K_d and K_{max} respectively 2 and 6.

Tab. 3 shows the scores of the internal validation indices computed on $\mathcal{R}'_d$ varying the number of clusters that the adversaries are looking for. As we can see, the number of clusters estimated by the adversaries after each operation coincides with the predetermined deceptive number of clusters K_d.

Table 3
Scores of the Davies-Bouldin Index (DB), Calinski-Harabasz Index (CH), and Silhouette Coefficient (SIL) based on the number of clusters searched for.

		Estimated Number of clusters				
		2	3	4	5	6
Shuffle	DB	4.13	**3.76**	3.86	4.25	4.50
	CH	19.54	**20.45**	16.26	13.90	12.52
	SIL	0.044	**0.063**	0.063	0.046	0.038
Shuffle-Incr	DB	4.36	4.07	**3.55**	3.62	3.57
	CH	17.16	17.90	**19.50**	15.26	12.77
	SIL	0.040	0.056	**0.070**	0.062	0.059
Shuffle-Red	DB	**3.90**	4.68	4.59	4.43	4.76
	CH	**23.17**	16.01	12.97	11.31	10.15
	SIL	**0.052**	0.037	0.030	0.032	0.032

4 POSSIBLE ADVERSARIES

4.1 The Attack Model

This subsection defines three models, each representing an adversary with different knowledge of both the content of the repository $\mathcal{R}'$ and the deception techniques adopted.

- **Black Box adversaries:** They are the weakest kind of adversaries we consider in this work. They are not aware of the proposed deceptive techniques and believe that the exfiltrated repository $\mathcal{R}'$ is the original one.
- **Gray Box adversaries:** These adversaries suspect that some deceptive operations may have been executed on the repository $\mathcal{R}'$. Nonetheless, even though they know the presented deceptive operations, Gray Box adversaries neither know how many and which deception operations were performed, nor how many and which deceptive keywords have been used to perform each operation.
- **Enhanced Gray Box adversaries:** They share the same knowledge as the Gray Box adversaries. However, these adversaries also leverage the **Oracle Function** to obtain an ordered list of terms in $\mathcal{R}'$ that may have been replaced by the deceptive operations (details in Sec. 4.1.1). The ability to invoke the Oracle Function makes Enhanced Gray Box adversaries the strongest. They represent the worst-case scenario in which we evaluate the performances of the deceptive operations proposed.

We assume that Gray Box and Enhanced Gray Box adversaries can remove deceptive keywords from the repository $\mathcal{R}'$, as discussed in Sec. 4.2. In addition, we assume that all the adversaries described in this section: cannot access the mapping of the keywords replacements, use K-means as the clustering algorithm and the Silhouette Coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index to estimate the number of clusters contained in the exfiltrated repository. It is important to note that none of the three adversaries can self-estimate metrics such as the Purity or the ARI on their clustering since they do not know the ground truth of the repository $\mathcal{R}$.

4.1.1 The oracle function

Knowing which are the deceptive keywords in the repository $\mathcal{R}'$ may be a great advantage for the adversaries. In-

deed, leveraging this information, they can eliminate those terms to subvert the operations.

In this section, we define the Oracle Function to emulate adversaries that somehow gained access to the list of terms and thus are able to select the deceptive terms we used to build the deceptive repository.

Definition 4.1. Oracle ($\mathcal{R}'$)$\Rightarrow L_k$.
Consider the repository $\mathcal{R}'$ made of M keywords, such that D keywords are all and only the deceptive keywords used to generate the repository $\mathcal{R}'$, and the remaining $M-D$ terms are the original keywords that are both in $\mathcal{R}$ and $\mathcal{R}'$. The Oracle Function takes as input a deceptive repository $\mathcal{R}'$ and returns as output an ordered list of M keywords $L_k = \{dk_1, \ldots, dk_D, k_{D+1}, \ldots, k_M\}$, where the first D items of the list are deceptive keywords, while the remaining $M-D$ items are unchanged keywords. In particular, the deceptive keyword dk_i, with $i \in \{1, \ldots, D\}$, represents the i^{th} deceptive keyword used to generate the repository $\mathcal{R}'$. The remaining $M-D$ keywords are ordered as they were the next terms to be replaced by the operation used to generate the repository $\mathcal{R}'$.

Although Enhanced Gray Box adversaries, through the Oracle Function, can access in an ordered way all the deceptive keywords, they still do not know the number D of deceptive keywords contained in the repository. Thus, Enhanced Gray Box adversaries cannot be sure if the j^{th} keyword, with $j \in \{1, \ldots, M\}$, provided by the Oracle Function, is a deceptive keyword or not.

4.2 Countering the operations

The Gray Box adversaries are aware of the deceptive operations described in this work. In this section, we explore a possible approach that this kind of adversary could carry on to counter the deceptive operations and build a new repository $\mathcal{R}^r$ that is more significant than $\mathcal{R}'$.

The adversaries, to smooth the effect of the deceptive operations, have to solve the following two problems: (1) estimate the right number of clusters contained in the repository, and (2) estimate how many and which deceptive keywords are in the repository $\mathcal{R}'$. Recall that if the adversaries evaluate the number of clusters in $\mathcal{R}'$ leveraging standard techniques such as the Silhouette score or other internal validation measures as explained in Sec. 2.2.2, they find out the deceptive clusters accordingly with Sec. 3.5. Therefore, adversaries have to counter the deceptive operations to obtain meaningful information from the exfiltrated repository.

The adversaries know that, whatever the adopted policy to replace keywords with deceptive keywords, the subsets of documents that form the clusters in $\mathcal{R}'$ are held together, or separated, among them by the keywords with higher TF-IDF weight. Thus, a possible approach to reduce the effect of the deceptive operations and restore the original clustering of documents is to remove those keywords that are likely deceptive keywords from the repository $\mathcal{R}'$.

The Gray Box adversaries do not know how many keywords they have to remove from $\mathcal{R}'$. To estimate the number of keywords to remove and the real number of clusters in $\mathcal{R}'$, they may perform the following iterative approach: using an internal validation index (*e.g.,* Silhouette Coefficient), they infer the optimal number of clusters K_{init} for the repository $\mathcal{R}'$. For each cluster C'_i in $\mathcal{R}'$, with $i \in \{1, \ldots, K_{init}\}$, they rank the keywords by TF-IDF weight and build the ordered list of keywords LK_i for C'_i. Let T be the maximum number of keywords the adversaries are willing to remove from each document. The choice of T is a trade-off for the adversaries: the more keywords the adversaries delete from the documents, the higher the probability of discarding both deceptive and original keywords. The adversaries perform T times the following procedure. For each cluster C'_i in $\mathcal{R}'$, they select the keyword k from LK_i with the highest TF-IDF weight. Then, the adversaries delete all occurrences of k from the documents in R', and remove k from the list LK_i. Let K_{estim} be the maximum number of clusters the adversaries suppose to be in $\mathcal{R}$. At the end of each step, the adversaries assess on the repository the internal validation score for K different number of clusters, with $K \in \{2, \ldots, K_{estim}\}$. At the end of the T*$K_{deceptive}$ steps, the adversaries evaluate all the internal validation scores they computed and select the configuration that achieved the best score accordingly with the internal validation index they used. If R^r is the repository that achieves the best internal validation score, the adversaries assume as R^r the restored repository of R. Therefore, the adversaries consider the keywords deleted from R' to achieve R^r as the deceptive keywords of R' and the number of clusters of R^r as the number of topics covered by R.

Enhanced Gray Box adversaries follow the same approach as Gray Box, but they have a significant advantage in determining the deceptive keywords since they can rely on the Oracle Function. Indeed, Enhanced Gray Box adversaries leverage the Oracle Function to build lists of potential deceptive keywords to remove.

For instance, assume that Enhanced Gray Box adversaries exfiltrate the repository $\mathcal{R}'_d$, that has been generated starting from $\mathcal{R}_d$ using the Basic Shuffle operation and 60 deceptive terms for each document (see Fig 3(a)). The adversaries apply the procedure outlined in this section to obtain a repository $\mathcal{R}^r$. Enhanced Gray Box adversaries set T and K_{max} respectively to 300 terms and 6 clusters. Fig. 4 shows the Silhouette scores obtained by Enhanced Gray Box adversaries removing the terms from the repository $\mathcal{R}'_d$. Each colored line represents the Silhouette score for a different number of clusters. At the end of the assessment, the adversaries find out that the Silhouette score is maximized when removing 130 terms from $\mathcal{R}'_d$ and searching for 5 clusters (red dot in the figure). Thus, the adversaries build the repository $\mathcal{R}^r$ accordingly with the configuration found.

5 Results

5.1 The deceptive repositories

To evaluate the adversaries' performances, we built 6 different deceptive repositories starting from the repository $\mathcal{R}_d$ described in Sec. 2.1. To build the 6 repositories, the following deceptive operation has been applied on $\mathcal{R}_d$: Basic Shuffle, Shuffle Increment, and Shuffle Reduction. Each deceptive operation has been executed twice, once partitioning the documents through random selection and once through the constrained version of K-means. With both approaches, all the subsets have been made approximately of the same number of documents. The keywords have been replaced in descending order by their TF-IDF weight. Finally, for the

Table 4
Adjusted Rand Index (ARI) values achieved by the adversaries on the repository $\mathcal{R}'_d$ against Basic Shuffle, Shuffle Incrementation (Shuffle Incr.), and Shuffle Reduction (Shuffle Red.) applied to subsets of documents selected randomly (random), by grouping similar ones (similar), or involving random terms. Note that the ARI score obtained by the adversaries on the original repository R is 0.94.

		Black Box		Gray Box			Enhanced Gray Box		
	m-replacement	#clust	ARI	#del	#clust	ARI	#del	#clust	ARI
Basic-Shuffle$_{random}$	60	3	-0.004	50	5	0.33	130	5	0.56
Shuffle-increment$_{random}$	60	4	-0.005	60	6	0.28	130	5	0.56
Shuffle-reduction$_{random}$	60	2	-0.003	50	5	0.23	130	5	0.56
Basic-Shuffle$_{similar}$	60	3	-0.004	60	6	0.27	110	6	0.51
Shuffle-increment$_{similar}$	60	4	-0.005	80	6	0.21	110	6	0.51
Shuffle-reduction$_{similar}$	60	2	-0.003	40	6	0.20	110	6	0.51

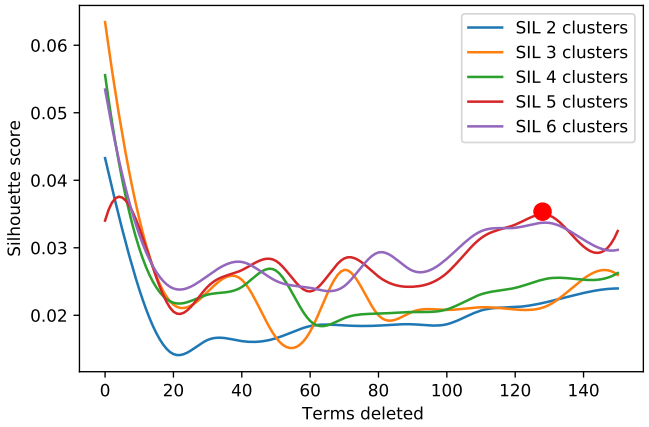


Figure 4. Values of the Silhouette Coefficient obtained on the modified repository removing up to 300 terms from each identified cluster. The red dot indicates the number of terms to remove in order to achieve the best Silhouette score.

sake of comparison, all the operations performed the same number of m-terms-replacements. We set the number of m-terms-replacements to 60, since, according to our experiments, it is the minimum number of replacements such that the Silhouette Coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index return the deceptive number of clusters for the repositories. For example, Fig. 3(a), 3(b), 3(c), respectively, show the deceptive repositories built with the Basic Shuffle, the Shuffle Reduction, and Shuffle Increment operations, using the constrained version of K-means to build the subset of documents.

5.2 Attacking the deceptive repositories

To evaluate the adversaries' performance, we use the Adjusted Rand Index (ARI) [29]. Given a predicted clustering (the one obtained by the adversaries, in our case) and the clustering given by the true labels of the documents, the ARI measures the similarity between these two clusterings. The value of the ARI varies between -1 and 1, where a value of 1 indicates a perfect match between the two clusterings, a value close to 0 a random labeling of the predicted clustering, and a negative value a labeling worst than a random one.

The Black Box adversaries believe they have exfiltrated the unmodified repository. Therefore, Black Box adversaries analyze the exfiltrated repository as described in Sec. 2. At the end of the analysis, Black Box adversaries will discover just the deceptive clusters. Since our operations build each deceptive cluster by grouping together with a uniform distribution of documents of different topics, such clustering of the Black Box adversaries achieves an ARI approximately equal to 0, which corresponds to the same result that the adversary would have by grouping the documents randomly.

Enhanced Gray Box and Gray Box adversaries are aware that the exfiltrated repositories could be deceitful. However, they can not be sure of that. They have two options. They can consider the repositories not deceitful and analyze the repositories as the Black Box adversaries obtaining the same results. Alternatively, they can try to get rid of the deceptive keywords, for example, applying the algorithm described in Sec. 4.2, and building for each exfiltrated repository a recovered version $\mathcal{R}^r$. We assume that the adversaries analyze the deceptive repositories by searching for 2 up to 6 possible clusters, attempting to remove up to 150 keywords, and evaluating the repositories using the Silhouette Coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index. At the end of the analysis, the adversaries find out that the best configurations for each repository $\mathcal{R}^r$ are the ones reported in Table 4. The table reports only the results with higher ARI obtained by the Enhanced Gray Box and Black Box adversaries. All the reported results have been achieved by leveraging the Silhouette Coefficient. Indeed, it was the internal evaluation index that provided better performances in all the experiments. Enhanced Gray Box adversaries are those that achieve the highest ARI, between 0.51 and 0.56, while the Gray Box obtains an ARI between 0.20 and 0.33. The highest performances of the Enhanced Gray Box adversaries are due to the Oracle Function. Although the Enhanced Gray Box adversaries remove several original keywords from the documents, on average 60 original keywords from each document (over a total of 2,433 terms), they can completely clean up the document from the deceptive keywords. Conversely, the Gray Box adversaries remove a few original keywords from the documents (about 10 original keywords). However, in the Gray Box scenario, each document still contains more than 10 deceptive keywords on average that are sufficient to keep the document in the deceptive clusters created with the deceptive operations described in this work. It is worth noting that the repositories built using the constrained version of K-means (*similar* in Tab. 4 and Tab. 6) are those that lead both White-box and Gray-box adversaries to obtain the worst results (*i.e.,* incorrectly clustering the documents).

Hence, the *similar* approach appears to be the most robust against the counter operation. Enhanced Gray Box adversaries are able to achieve notable results. Indeed, an ARI of 56% can intuitively be interpreted as the 56% of documents are correctly clustered. However, some considerations have to be taken into account. Even assuming the Enhanced Gray Box adversaries can self-estimate the ARI achieved by the repositories, they still can not infer which documents are correctly labeled and which are not (*i.e.,* the adversaries do not know the original topics of the documents). An ARI of 56% means that if the adversaries pick one document in a cluster C', there is roughly 50% probability that C' contains the majority of documents with the same original topic.

5.3 Increasing the number of true topics

In this subsection, we explore how deceptive operations perform by scaling up the number of true topics in the original repository. For this analysis, we built 6 different deceptive repositories, each of them containing a different number of true topics, from 2 up to 7. All the repositories have been built leveraging the Basic Shuffle operation (see Sec. 3.2) and partitioning the documents randomly. In particular, the 7 true topics we used to build the repositories are the three described in Sec. 2 (Artificial Intelligence (AI), Database (DB), Cryptography and Security (CR)), and four new topics (Robotics (RO), Computers and Society (CS), Logic in Computer Science (LO), Computational Complexity (CC)) that we collected in the same fashion always from ArXiv. To assess the performance of the Shuffle Operation, we compute for each repository the ARI before applying the deceptive operation (*i.e.,* the ARI the adversaries would have gotten exfiltrating the plain repository) and the ARI obtained by the three kinds of adversaries after the operation has been applied.

Table 5 sums up the results of these experiments. As for the previous experiments, the Black Box adversaries achieve an ARI of about 0 for all the configurations. This result is straightforward since the Black Box adversaries attempt to cluster the deceptive repositories without applying any countermeasures. Instead, in the other cases (Original repository, Enhanced Gray Box, and Gray Box adversaries), the ARI drops as we increase the number of topics into the repository. However, while analyzing the original repositories, the ARI falls of few points (-0.11), from 0.96 on the repository containing 2 topics to 0.85 on the repository containing 7 topics, for the Gray Box and the Enhanced Gray Box the ARI fall down drastically. The Gray Box adversaries achieve an ARI of 0.51 in the case of a deceptive repository made of 2 topics, while an ARI of 0.12 for the deceptive repository made of 7 topics. The Enhanced Gray Box adversaries go from 0.68 to 0.28. These results show that the deceptive operations become much more effective as the number of clusters in the deceptive repository increases.

5.4 Topic Modeling

The previous subsections analyzed the effect of deceptive operations on document clustering. In particular, this subsection shows how deceptive operations affect topic modeling. We consider adversaries that try to infer the underlying topics of the exfiltrated repository leveraging LDA, the one of most popular topic modeling algorithms. Given a repository of documents and a number of topics T, LDA computes for each term in the documents the probability that the term belongs to one of the T topics. The set of n terms that have the highest probability according to the LDA algorithm is defined as the *topic keywords* of each topic. Note that the most used number of topic keywords is 10 [16].

Table 5
Scores of ARI achieved by the adversaries in the Original repository and after the Basic Shuffle operation when increasing the number of topics.

	Number of topics involved					
	2	3	4	5	6	7
Original	0.96	0.94	0.92	0.92	0.89	0.85
Black Box	-0.0002	-0.004	-0.005	-0.005	0.006	0.0003
Gray Box	0.51	0.33	0.29	0.31	0.14	0.12
Enh. Gr. Box	0.68	0.56	0.48	0.47	0.43	0.28

Table 6
Percentages of deceptive keywords returned by LDA according to the number of keywords retrieved from each topic and the deceptive operation applied on the repository.

	Number of topic keywords				
	10	20	30	40	50
Basic-Shuffle$_{random}$	100%	98%	98%	98%	95%
Shuffle-increment$_{random}$	100%	93%	92%	86%	80%
Shuffle-reduction$_{random}$	100%	95%	88%	85%	75%
Basic-Shuffle$_{similar}$	100%	98%	96%	96%	89%
Shuffle-increment$_{similar}$	100%	90%	86%	83%	80%
Shuffle-reduction$_{similar}$	100%	100%	95%	85%	74%

Assume that a Black Box adversary wants to infer the topic of each cluster of documents into the exfiltrated repository R_d. Thus, the number of topics T that such adversaries seek is equal to the number of deceptive clusters.

To assess if our deceptive operations are able to deceive the Black Box adversaries in inferring the topics, we use the same deceptive repositories described in Sec. 5.1, and the LDA version of Scikit-learn library [30]. In particular, we measure the presence of deceptive keywords that the adversaries retrieve with LDA for each topic with n varying from 10 to 50. Analyzing the topic keywords computed by LDA on the repositories, we make the following observations:

(i) With $n = 10$, for each topic, all the keywords retrieved by LDA are deceptive keywords. Instead, with $n = 50$, the percentage of deceptive keywords varies between 74% and 95%, depending on the deceptive operations used to build the repository. Thus, even considering a significant amount of topic keywords ($n = 50$), very few real topic keywords are retrieved using LDA. Moreover, as we can see from Tab. 6, the real topic keywords are mainly at the lower position of the rank, which means they have a low probability of being significant for the specific topic. Indeed, looking at the real topic keywords returned by LDA, we find that they are generic keywords of computer science, such as: *lemma, root, induct, resource, program, rate, label*.

(ii) Each topic retrieved by LDA describes with its top ten keywords a different deceptive cluster within the decep-

tive repository. Consequently, adversaries relying on LDA will infer the deceptive topics designed by the defenders.

(iii) Each deceptive cluster of the deceptive repository has one topic associated with it.

Combining these observations, we have that LDA finds as topic keywords for a certain deceptive cluster, the same deceptive keywords used to build the deceptive cluster itself. Thus, the defender has the ability to manipulate the topic that the adversaries will infer, choosing wisely the deceptive keywords to use with the deceptive operations. The defender has multiple strategies to pick the deceptive keywords in order to show a specific deceptive topic to the adversaries. Suppose the defender selects deceptive keywords that fit both the deceptive context of the sentence and the part of speech of the terms to be replaced. In that case, it will be more challenging also for a domain expert to recognize the documents modified by our operations. To automatically perform this task, the defender can leverage language modeling techniques such as [31], [32]. This paper does not discuss the possible strategies of this extension, as it affects the second phase of the attack, while we focus on the first (see Sec. 1).

Note also the following interesting consequence of the observations depicted above. Adversaries that first leverage LDA to feed a clustering algorithm at the end of the computation will group the documents of R_d in a way very close to the deceptive clusters constructed by the defender. Indeed, on our 6 deceptive repositories, the Black Box adversaries group on average the 97% of the documents in R_d accordingly to the deceptive clusters built by the deceptive operations.

As a further experiment, to assess the robustness of the deceptive operations against LDA, we also leverage a commercial topic modeling tool, the **Amazon Comprehend-Topic Modeling** [10]. It is a topic modeling tool developed by Amazon Inc. that leverages the Amazon SageMaker Latent Dirichlet Allocation (LDA) algorithm, a custom version of LDA developed by Amazon Inc.. Given a repository of documents, Amazon's tool provides as output two CSV files. The first file reports for each document of the repository the topic number that the document is assigned to, and the proportion of the document concerned with that specific topic. Instead, the second file contains the top 10 topic keywords for each topic.

As for the results obtained with SKlearn's implantation of LDA, at the end of the experiments on the deceptive repositories R_d, all the topic keywords returned by Amazon's tool are deceptive keywords. All the topic keywords retrieved belong to a single deceptive cluster, and each deceptive cluster has a topic associated with it. The result of the experimentation with Amazon's tool shows that our proposed deceptive operations are robust also if adversaries employ an implementation of LDA not known to the defender, and with optimized parameter and data processing pipeline.

Note that leveraging the Amazon Comprehend tool makes sense only for the Black Box adversaries. Indeed, the Enhanced Gray Box and Gray Box adversaries are aware of the presence of deceptive keywords in the exfiltrated repository. Thus, they do not perform topic modeling but attempt to get rid of the deceptive keywords and cluster the documents achieving the poor results shown in the previous sections.

6 Related work

Adversarial setting: Several studies propose attacks to clustering algorithms through the generation of adversarial settings [33], [34]. Generally, an adversary injects malicious examples into the training data to impact the clustering results. There are two main typologies of these attacks: poisoning and obfuscation. The poisoning attack aims to worsen the clustering results as much as possible by corrupting the data. The strategy is to create new clusters or bridges between clusters adding samples within the dataset. In the first case, the purpose is causing the misclassification of a single cluster into more clusters. As for the bridge case, the goal is to pretend that two different original clusters are one. Instead, an obfuscation attack aims to hide a specific data set. Typically it consists in adding a set of samples to join the target cluster to hide with another one. As a result, clustering methods return a unique cluster that conceals the target cluster with another one.

Differently from previous studies that focused on fooling image classifiers [33], [34] and malware classifier [34], we target text document classifiers. Specifically, DARD introduces a novel approach by utilizing adversarial settings not for attacking models but as a defense mechanism against adversaries relying on automated techniques to classify exfiltrated repositories. This paradigm shift also alters the threat model. Previous studies assumed adversaries would carry out stealthy attacks and target a specific system, typically knowing its parameters. However, in our work as defenders, we cannot assume in advance the number of clusters the attackers will seek or whether they will attempt to circumvent the DARD system.

Deceptive repositories: Chakraborty et al. propose Forge [35]. This system leverages ontologies to generate new fake documents, credible to unauthorized readers, from a set of original documents. The resulting deceptive repository will contain fake and original documents that are indistinguishable to unauthorized readers. Identifying the fake documents within the repository requires extensive reading and thus a significant investment of time to differentiate them from the original ones. WE-Forge [36], an extension of Forge, goes beyond ontologies and utilizes word embeddings to automatically generate fake documents that closely resemble authentic ones, enhancing the credibility of the forged documents. While Forge and similar systems focus on building a deceptive repository to mislead the human reader, we aim to generate a deceptive repository to deceive automatic systems.

7 Conclusion

In this work, we proposed DARD, a framework of 4 deceptive operations able to manipulate the resulting clusters of a repository of documents, to deceive the adversaries that use automatic approaches to classify exfiltrated documents. To this end, the deceptive operations replace some of the original keywords in the documents with deceptive keywords through term-replacement operations. We outlined

how to apply the term-replacement operations for each deceptive operation and analyzed the minimum number of terms needed to be replaced. Then, we investigate different criteria for selecting the terms to be replaced, highlighting the pros and cons of the different approaches. We show experimentally that our operations can achieve a high level of deception. We conduct our experiments with three different types of adversaries: the Black Box, an adversary who does not know anything about deceptive operations; the Gray Box, that knows how deceptive operations work; and the Enhanced Gray Box, an adversary that can leverage the Oracle Function to discover the potential deceptive keywords in the repository. Our results show that deceptive operations completely deceive adversaries without knowledge of this work (0% ARI). They are very effective (average ARI of 25%) against those adversaries who know how the deceptive operations work (Gray Box), achieving, in the worst-case scenario with the Enhanced Gray Box adversaries, an average ARI of 53.5%. In addition, we analyzed the impact of deceptive operations in the topic modeling task. We found that when the adversaries perform topic modeling with LDA on the deceptive repositories, LDA describes the topics using only deceptive keywords when only 10 keywords for topics are requested. Moreover, we show that our approach against topic modeling successfully deceives also commercial tools such as Amazon Comprehend.

REFERENCES

[1] Deloitte. (2021) Impact of covid-19 on cybersecurity. [Online]. Available: https://www2.deloitte.com/ch/en/pages/risk/articles/impact-covid-cybersecurity.html

[2] Interpol. (2021) Covid-19 cyberthreats. [Online]. Available: https://www.interpol.int/Crimes/Cybercrime/COVID-19-cyberthreats

[3] J. Tidy, "The three russian cyber-attacks the west most fears," https://www.bbc.com/news/technology-60841924.

[4] CISA, "Impacket and exfiltration tool used to steal sensitive information from defense industrial base organization," https://www.cisa.gov/uscert/ncas/alerts/aa22-277a.

[5] S. AG. (Oct. 2020) Software ag adhoc: Disruption of services due to malware attack. [Online]. Available: https://www.softwareag.com/en_corporate/company/news/2020/1005_malware_attack.html

[6] T. Group. (May 2020) Toll it systems update. [Online]. Available: https://www.tollgroup.com/toll-it-systems-updates

[7] Verizon. (2021) Dbir - 2021 data breach investigations report. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/

[8] Wikipedia. (2020) 2020 united states federal government data breach. [Online]. Available: https://en.wikipedia.org/wiki/+2020_United_States_federal_government_data_breach

[9] Intel, "Intel software guard extensions," https://www.intel.com/content/www/us/en/developer/tools/softwareguardextensions/overview.html.

[10] Amazon, "Topic modeling," 2021. [Online]. Available: https://docs.aws.amazon.com/comprehend/latest/dg/topic-modeling.html

[11] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000.

[12] M. Marcińczuk *et al.*, "Text document clustering: Wordnet vs. tf-idf vs. word embeddings," in *Proceedings of the 11th Global Wordnet Conference*, 2021, pp. 207–214.

[13] "Open-access archive. arxiv." [Online]. Available: https://arxiv.org/

[14] M. S. G. Karypis, V. Kumar, and M. Steinbach, "A comparison of document clustering techniques," in *TextMining Workshop at KDD2000 (May 2000)*, 2000.

[15] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine learning*, vol. 42, no. 1-2, pp. 177–196, 2001.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[17] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing & Management*, vol. 50, no. 1, pp. 104–112, 2014.

[18] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization." Carnegie-mellon univ pittsburgh pa dept of computer science, Tech. Rep., 1996.

[19] R. Tibshirani *et al.*, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[20] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, 1985.

[21] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[22] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[24] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[25] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[26] U. Cambridge, *Online edition (c) 2009 Cambridge UP An Introduction to Information Retrieval Christopher D*. Manning Prabhakar Raghavan Hinrich Schütze Cambridge University Press . . . , 2009.

[27] K. Wagstaff *et al.*, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, 2001, pp. 577–584.

[28] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 911–916.

[29] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[32] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[33] J. Dutrisac and D. B. Skillicorn, "Hiding clusters in adversarial settings," in *2008 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2008, pp. 185–187.

[34] B. Biggio *et al.*, "Is data clustering in adversarial settings secure?" in *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 2013, pp. 87–98.

[35] T. Chakraborty, S. Jajodia, J. Katz, A. Picariello, G. Sperli, and V. Subrahmanian, "Forge: A fake online repository generation engine for cyber deception," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[36] A. Abdibayev, D. Chen, H. Chen, D. Poluru, and V. Subrahmanian, "Using word embeddings to deter intellectual property theft through automated generation of fake documents," *ACM Transactions on Management Information Systems (TMIS)*, vol. 12, no. 2, pp. 1–22, 2021.