Data Loss Prevention for Cross-Domain Information Exchange

Kyrre Wahl Kongsgård

Thesis submitted for the degree of Philosophiae Doctor
Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo
September 11, 2017

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 1934

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.
Print production: Reprosentralen, University of Oslo.

This thesis consists of an introduction and the following seven papers:

Paper I
Policy-Based Labelling: A Flexible Framework for Trusted Data Labelling
In Proceedings International Conference on Military Communications and Information Systems (ICMCIS), 2015, IEEE conference proceedings, IEEE.
Kyrre W. Kongsgård, Nils A. Nordbotten, and Stian Fauskanger

Paper II
Automatic Security Classification by Machine Learning for Cross-Domain Information Exchange
Military Communications Conference (MILCOM), 2015, pages 1590-1595, IEEE.
Hugo Hammer, Kyrre W. Kongsgård, Aleksander Bai, Anis Yazidi, Nils A. Nordbotten and Paal E. Engelstad

Paper III
Automatic Security Classification with Lasso
International Workshop on Information Security Applications (WISA), 2016, pages 399-410, Lecture Notes in Computer Science 9503, Springer.
Paal E. Engelstad, Hugo Hammer, Kyrre W. Kongsgård, Anis Yazidi, Nils A. Nordbotten and Aleksander Bai

Paper IV
Data Loss Prevention Based on Text Classification in Controlled Environments
Information Systems Security: 12th International Conference (ICISS) 2016, pages 131-150, Lecture Notes in Computer Science 10063, Springer.
Kyrre W. Kongsgård, Nils A. Nordbotten, Federico Mancini and Paal E. Engelstad. (Springer Best Paper Award)

Paper V
An Internal/Insider Threat Score for Data Loss Prevention and Detection
In Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics (IWSPA), 2017, pages 11-16, ACM.
Kyrre W. Kongsgård, Nils A. Nordbotten, Federico Mancini and Paal E. Engelstad.

Paper VI

Data Leakage Prevention for Secure Cross-Domain Information Exchange

Communications Magazine, Military Communications, 2017, IEEE. (Accepted for publication)

Kyrre W. Kongsgård, Nils A. Nordbotten, Federico Mancini, Raymond Haakseth and Paal E. Engelstad.

Paper VII

Data Loss Prevention for Cross-Domain Instant Messaging

Symposium Series on Computational Intelligence (SSCI, CISDA), 2017, IEEE. (Accepted for publication)

Kyrre W. Kongsgård, Nils A. Nordbotten, Federico Mancini and Paal E. Engelstad.

Acknowledgments

I wish to thank my supervisors Nils Agne Nordbotten and Federico Manicini for their support and guidance. I also wish to thank my collaborators Paal E. Engelstad, Raymond Haakseth, Stian Fauskanger, Hugo Hammer, Aleksander Bai and Anis Yazid, as well as the rest of the ISIC project group (Bodil Hvesser Farsund, Anne Marie Hegland, Frode Lillevold and Anders Fongen), and the Norwegian Defence Research Establishment and the Institute for Technology Systems at the University of Oslo (formerly UNIK) for jointly financing the work. I also wish to thank my trial defense opponents Edgar Lopez and Slobodan Petrovic for their feedback and Hallo Langweg at the COINS initiative for organizing it. Lastly, I wish to thank my family for their support.

Kyrre Wahl Kongsgård, Kjeller, September 11, 2017

Introduction

Contents

1 Introduction

The topic of this thesis revolves around developing and enhancing methods for data loss prevention and detection (DLP) in the context of cross-domain information exchange. Data loss prevention, as a subfield of information security, is tasked with the goal of preventing any confidential or sensitive data from being leaked to unauthorized third parties. Examples of an organization's confidential data may include intellectual property (IP), patient records or other health-related data, financial reports, credit card numbers or for the military or government: classified information. In other words, the need for DLP systems is motivated not only by financial reasons but also for supporting national security.

The proposed security mechanisms involved in preventing data loss ranges from encrypting file systems, to the physical removal of USB ports and restricting a software user's ability to copy and paste text, to inspecting all outgoing network traffic for the presence of known sensitive or confidential data. It is especially the latter content inspection based paradigm, that we adopt and adapt to the secure cross-domain information exchange setting. As a simple example of a content-based DLP system one could use regular expressions to detect the presence of any credit card- or social security numbers that have been embedded in outgoing email messages. These regular expressions makes for a rudimentary, but effective, rule-based DLP system that yields a high performance, as measured in terms of precision and recall, for detecting credit card numbers. Their effectiveness is attributed to the structured nature of the confidential data, i.e., the format of credit card numbers is well-defined and is not typically confused for other contents of benign email traffic. This situation stands in stark contrast to the one we are studying, in the sense that we are concerned with a DLP scenario in which, for the most part, the confidential data encompasses generic non-structured, natural language text data. Thus, instead of relying on hand-crafted rules, we are forced to operate within the realm of natural language processing (NLP) and a primarily data-driven machine learning focused paradigm.

Summarily expressed, the research goal motivating much of our work can be stated as the task of assessing the likelihood of the presence of *classified material* in textual data objects that are being transferred between two computer systems, for example, an email- or chat message-based communication protocol, with the end objective of ensuring that the governing security policy is not violated such that sensitive data is leaked.

Before we outline how each component in the thesis individually contribute

to achieving this as well as how they interact and relate to each other, we first introduce and motivate the general cross-domain information exchange setting, and then discuss some of the more nuanced, domain specific aspects that arise in this specific DLP setting.

Cross-domain information exchange

The physical compartmentalization of a computer system into isolated security domains is a time-tested approach for ensuring the confidentiality of sensitive or classified data and for protecting the integrity and availability of the systems. A concrete example, often encountered in military environments, is the notion of air gapping computer systems in order to handle information of varying sensitivity levels. There could be three physically separated systems, for example, one for "NATO Secret", another for "NATO Restricted" and a third for the processing of "Unclassified" data, with the latter system potentially being connected to the Internet. In order to transfer information between a "High"and "Low" (e.g. "NATO Secret" and "NATO Restricted" respectively) domain one could employ the use of a physical storage device, such as a USB memory stick. Figures 1.1 provides an illustration of how this process works in practice.

Modern military operations are increasingly coalition-based in nature, with multiple NATO members participating in joint operations or assisting and communicating with NATO members and potentially non-members. The ability to exchange information securely between these entities is an important capability for safe and efficient operations. A recent case study of such a system is the Afghanistan Mission Network (AMN). As a NATO funded network implemented in 2010, the intended purpose of the AMN was to enable the sharing of critical information among the International Security Assistance Force (ISAF). It was designed to provide support for chat, VOIP, email, HTTP and video chat applications [73]. While both the AMN and its successor Federated Mission Networking (FMN) uses a single security domain to simplify information sharing internally, it is still necessary to have Information Exchange Gateways (IEG) that make use of cross-domain sharing solutions such as data guards to enable the exchange of information with external security domains [60].

In order to facilitate information exchange across two previously isolated security domains we need a cross-domain sharing solution, whose role is to perform a set of security controls (e.g. information flow control, antivirus and access control). These factors ensure that the interconnection does not compromise security.

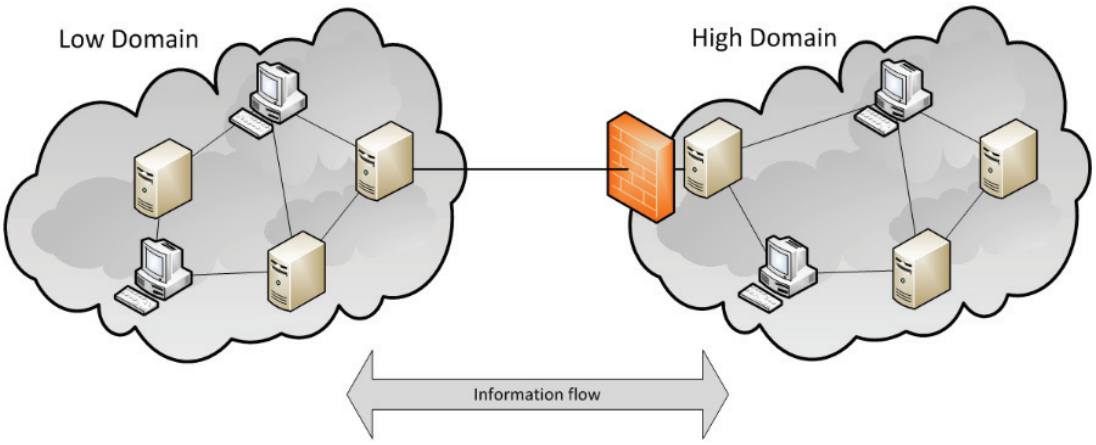

Figure 1.1: The data guard enables two-way information flow between a High and Low domain. Each object passing through the guard has its security label validated, its content checked according to policy (e.g. content may be scanned for malware and the presence of dirty words), and its sender and destination fields may be verified and subject to access control. Having passed these checks, the object is then released on the condition that its security label is such that it is considered, as specified by the governing security policy, to be releasable.

Cross-domain data guards

A data guard is one example of a cross-domain sharing solution that operates as a gateway deployed between domains. It typically requires that each data object, the user or service in the High domain requests to export, is first bound (using a suitable data-binding format) together with the security label. The bound object is then cryptographically signed using a digital signature scheme. On receiving the object, the guard validates the signature and assesses whether the value of the assigned security label is such that the data object is considered releasable, which is determined by the security policy (e.g. only data marked as NATO restricted and below is allow to pass from the high to the low domain). High assurance guards can be implemented utilizing a separation kernel to partition the guard into multiple components that interact through tightly controlled interfaces [35].

Content checking

These solutions do not, however, address the central question of whether or not the data object has been labelled correctly by the user or service that initiated the transfer request. To mitigate the risk of incorrect security labellings, another layer of protection in the form of a *content-checker* may be applied. For text-based data objects it is common to use a "dirty word list" mechanism, which scans the object for the presence of keywords considered to indicate classified content (e.g. security classifications, certain technical terms, locations, and project acronyms). The dirty word content-checker is a rudimentary security mechanism whose performance is wholly dependent on the quality of the list of offending keywords. This list must be regularly updated and maintained by domain experts – a laborious manual process. Effective content checking is essential for maintaining the security as military operations are becoming increasingly coalition-based and moving away from the old "need to know" and towards a "need to share" paradigm, where the communication infrastructure is more ad-hoc in the sense that it consists of a myriad of different sub-systems, which are potentially operated by different agencies and countries. For non-text data, such as image or video files, techniques that are similar to those we have proposed may be applied (e.g. utilizing computer vision based methods), but this thesis focuses on text based data formats as well as certain structured data formats.

Outline and contributions

This thesis considers how one can increase the security of cross-domain information exchange. Our primary approach is to introduce sophisticated content checkers that scan the outgoing data objects for 1) the presence of classified information by both comparing against repositories of existing known classified content and by constructing models that infer and predict the sensitivity from a training set, and 2) data whose format does not conform with valid or known, specifically allowed data types. Both rule-based and data-driven content-checking methods are explored and evaluated. Meta-models built on top of these methods are utilized to further increase trust in the correctness of the assigned security label and to detect persistent mislabeling by malicious insiders or misconfigured services, thus effectively preventing or detecting potential data-leak incidents. Further, we have investigated the feasibility of augmenting and improving existing DLP methods by incorporating external author profiling signals such as authorship verification. By building authorship verifi-

cation models we capture the unique stylometric signature of users which can then be used to further reduce the number of incorrect classifications, e.g., by identifying content in certain communication settings such as email or instant messaging, for which the asserted authorship claim is incorrect. Below we provide a brief description of each paper and how they relate to each other (refer to Section 7 for more elaborate summaries) while Figure 1.2 shows the logical dependencies between the papers.

- Policy-based labelling (Paper I): We investigate how attribute based access control (ABAC) principles can be used to build systems for performing policy-based labelling. We envision that the resulting system can be utilized in conjunction with the automated security level estimation algorithms (see below), e.g., by implementing and invoking these methods as custom attribute modules whose predictions are part of the policy rule sets. More basic example rules include the use of dirty word list content checkers or format and content verification for structured data (e.g. XML schema validation).

- Automated sensitivity level estimation (Paper II, III and IV): For unstructured textual data we are not able to build simple rule-based detection algorithms. Thus, Paper II marks our foray into using NLP and machine learning based algorithms to construct data-driven classifiers that automatically predicts a document's security label based on its contents. This work is further expanded upon in Paper III, where we modify the inference algorithm such that the end model becomes become easier for humans to interpret. By keeping track of the documents a user accesses, and their associated sensitivity, we in Paper IV, construct classifiers that exploit this information, thus effectively providing a better context for the predictions, with the end result being a significant increase in the predictive performance of the models.

- Internal/insider threat scores (Paper V): We use the predicted confidence scores to construct probabilistic models of how users conduct security labelling. By aggregating any long-term discrepancies between the predicted and user- or service-assigned scores we demonstrate the feasibility of computing a metric that allows us to uncover anomalous labelling behaviour (indicative of malicious insider activity) and to better handle false alarms.

- Authorship verification/profiling (Paper VII): In this paper we combine the output from existing DLP systems with an authorship profiling model.

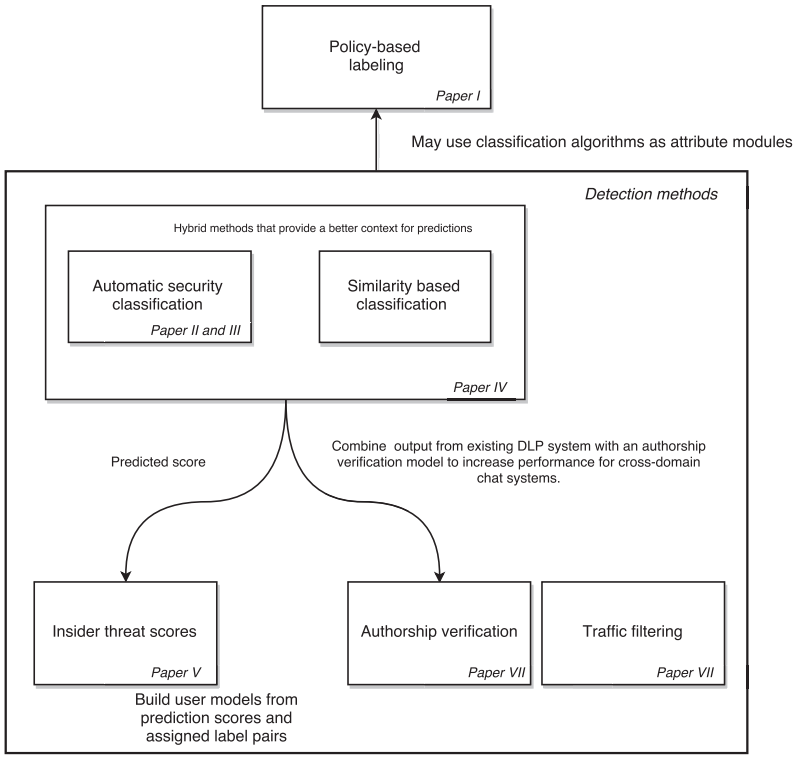

Figure 1.2: Graphical representation of how the papers included in this thesis relates to each other.

The rationale behind this is that by levering stylometric information we can, in a cross-domain communication setting, further reduce false alarms.

- Traffic filtering (Paper VII): Methods that distinguish between normal and abnormal traffic in a cross-domain instant messaging system. Examples of abnormal traffic here include encrypted data, image files, executable files and non-chat like textual data.

While we rely heavily on machine learning techniques for constructing our detection algorithms, the machine learning methods we use are established ones and the emphasis as such has not been on the development of completely novel methods (with the exception of the work done for detecting insiders). This is intentional as it has been showed that most of the predictive performance for a machine learning task is often captured by relatively simple methods (e.g. generalized linear models) and that the boost in performance attributed to a

complex method often is illusory in the sense that it is rightly attributed to implicit overfitting and unrealistic assumptions [36]. Instead, we focus our attention on the task of properly adapting time-tested methods (e.g. logistic regression) to problems that have received only cursory attention previously.

The remainder of the thesis is organized as described below: Section 2 discusses Data-Loss Prevention (DLP) solutions and the relationship between content checking and plagiarism detection. We provide background material on information retrieval concepts and similarity measures and present the related work for security label classification, external plagiarism detection and both rule-based and data-driven content-based DLP. Section 3 addresses the insider threat and overviews the visualization, anomaly-based methods and data sets that have been the topic of insider detection and prevention research. Section 4 describes how we increase trust in the validity of the data being sent, introduces the concept of author profiling, and present an overview of the field of authorship verification.

Section 5 provides an introduction to machine learning, including the challenges faced when developing and deploying the above-mentioned methods as security mechanisms. Section 6 introduces the background material for generative models and probabilistic programming. These concepts and modeling methodologies are necessary background knowledge for the discussion on how to construct models for detecting insider behavior (and to better handle the often large number of spurious alarms) by using the long-term/consistent discrepancies between the user-assigned and predicted security labels. We conclude the thesis in Sections 7 and 8 by summarizing each paper and highlighting our contributions before finishing up with some final thoughts on promising extensions and future research directions.

2 Data loss prevention and detection

In this section we discuss how cross-domain content checking relates to the more general DLP task. We survey related work for content-based DLP, plagiarism detection and sensitivity estimation (security level classification) while introducing the necessary background material to make the discussion as self-contained as possible.

Readers unfamiliar with machine learning (and the security aspects of the methods and their usages) are advised to refer to Section 5 for a brief overview.

DLP solutions

A content-checker is a sub-component of a data loss prevention (DLP) solution as applied to the cross-domain information exchange setting. A DLP solution usually refers to a more extensive product that includes support for centralized policy management, enforcement, data discovery and monitoring as well as a content analysis engine, whose role is to detect sensitive information by inspecting the contents of files directly [57, 74]. We will refer to DLP methods that revolve around content inspection as "content-based DLP", while the term "content-checker" will be reserved for methods adapted to the more constrained cross-domain information exchange setting.

There are several commercial products in the DLP space, but as it would require a significant amount of reverse engineering to analyze such a product [34], we therefore omit them from further discussion. Network-based DLP solutions operate by analyzing the payload data, such as the contents of HTTP headers or e-mail messages, in order to detect the presence of any sensitive content [34]. In terms of performing the detection itself, we can partition the current proposed approaches into the following categories [57]:

- Exact matching: By maintaining lists of hash values of known sensitive documents, instances of outgoing data objects violating DLP policies can be detected, with a high false negative rate, by simply computing the hash value and cross-referencing the said value with those values of the pre-computed list [57].
- Partial matches: Rolling hash methods such as the Rabin-Karp and Aho-Corasick (AC) algorithms can be used to detect partial matches or snippets in a scalable way (e.g. $\mathcal{O}(n)$ for AC) [76]. This category also includes

sequence-alignment algorithms, in which minor modifications do not derail or distort any attempts to recover potential matches [15].

- Rule-based: Hand-crafted rules for detecting the exfiltration of specific sub-types of data, such as regular expressions for credit card and social security numbers.

- Data-driven: The use of data-driven (including statistical analysis) techniques for detecting outgoing documents that share similarities (in terms of semantic content or meta-data) with known sensitive documents. Similarity based methods can be made scalable either through the construction of inverted indices [9], or by approximate methods and their efficient implementations [42, 49].

Plagiarism detection

Content-based DLP shares some similarities with the field of plagiarism detection, in which there has been more research conducted, especially in terms of enhancing the content analysis and detection routines. In plagiarism detection the task is to determine which (if any) part(s) of a given (suspicious) document have been plagiarized by the claimed author. There are two common scenarios which have branched into the sub-fields of *Extrinsic* (or External) and *Intrinsic* (or Internal) plagiarism detection:

- Extrinsic plagiarism detection: In this setting there is an external corpus to which the suspicious document is compared; that is, an explicit assumption exists that any potential plagiarized content can be traced back to one of the documents in the reference collection [2].

- Intrinsic plagiarism detection: For the Intrinsic task there is no external corpus and as such there is more emphasis on detecting any paragraphs or passages that deviate from the baseline characteristics as captured by the stylometric traits of the author. This task is much more challenging than the extrinsic plagiarism detection task.

As the extrinsic plagiarism detection task resemblances the data guard content-checking setting, we will therefore provide a overview of the related work in this area as it has influenced our own research to a greater extent than the more basic work done in content-based DLP.

Extrinsic plagiarism detection is often viewed as a two-step process that consists of candidate selection and alignment phases [32, 61]. In the candidate

selection phase the suspicious document is used to query the external repository for documents that have potentially been plagiarized (the candidates). This filtering step is necessary due to the high time complexity of the algorithms used in the alignment phase. The alignment phase takes as input the suspicious and candidate document pair and attempts to align their sentences [58]. For the candidate selection phase, it is a standard approach employed by many studies to recast the problem as an information retrieval one [39]. The external corpus is then used to create an inverted index. Indexing can be done on document or sentence levels [86]. When it comes to the alignment phase, one popular approach involves using a variation of the Rabin-Karp algorithm followed by the auxiliary post-processing step in which the set of detected plagiarized passages is merged. The Rabin-Karp algorithm produces adequate results only for verbatim copies and the plagiarizer can, through minimal obfuscation efforts, completely ruin the detection rate [58].

Document representations and similarity measures

As an intermediate representation for the documents most utilize a variation of the n-gram and Bag of Words (BoW) model. An n-gram is a contiguous sequence of n words; for example, the 2-grams (more commonly referred to as bigrams) we have for the short sentence "my name is kyrre":

$$\textbf{my name is kyrre} \rightarrow^{n\text{-}gram} \{(\text{my, name}), (\text{name, is}), (\text{is, kyrre})\} \tag{2.1}$$

The BoW model simply discards any word ordering not captured by the computed n-grams. While the combination of n-gram features and bag of words (BoW) model make for simple document representations, they remain robust with respect to word re-orderings, and have formed the basis for some of the most successful plagiarism detection methods [43, 62]. Machine learning using natural language data first involves computing some variation of n-gram frequency statistics. An especially popular variant of these frequency counts are the so-called term-frequency inverse document frequency (TF-IDF) weights. These are used to represent each document by a high-dimensional sparse vector $\mathbf{x}$, in which the x_i entry is the TF-IDF weight for words associated with dimension i. For example, a document denoted by d containing the words 'man', 'missile' and 'aide' would be transformed into the vector:

$$\mathbf{x}_d = \left[\overbrace{x_{d,1}}^{\text{man}} \overbrace{x_{d,2}}^{\text{missile}} \ldots \overbrace{x_{d,N}}^{\text{aide}} \ldots \right] \tag{2.2}$$

where N is the vocabulary size. The entries $\mathbf{x}_{d,t}$ are the TF-IDF weights defined as:

$$\mathbf{x}_{d,t} = \underbrace{\sqrt{f_{t,d}}}_{\text{tf}} \times \underbrace{\left(1 + \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}\right)}_{\text{idf}} \tag{2.3}$$

where $f_{d,t}$ denotes the frequency of term t in document d and D the set of all documents [52]. Informally, TF-IDF re-weights the frequencies to better capture the importance of the word or term t to the document d. These vector representations are used for computing similarities between the set of external documents and the suspicious document, as they constitute a vital component both in information retrieval and in sequence alignment methods. A similarity measure is a function $\texttt{sim}(x, x')$ that quantifies the similarity between two objects. For real-valued vectors, a common choice is the cosine similarity measure:

$$\texttt{sim}(\mathbf{x}_1, \mathbf{x}_2) = \cos\theta = \frac{\mathbf{x}_1^T \mathbf{x}_1}{\|\mathbf{x}_1\|_2 \|\mathbf{x}_2\|_2} \tag{2.4}$$

For textual objects another way of measuring the similarity between documents (more typically a document and a query string) is found in the world of statistical language modelling. A language model is a statistical representation of a document, in which we place a probability distribution over the words of the document. If we represent the language model of document d as a time-homogeneous Markov chain of order n, it follows that the probability of observing the particular sequence S is mathematically expressed by

$$P(S) = \prod_{i=1}^{r} P(k_i | k_{i-1}, \cdots, k_{i-(n-1)}) \tag{2.5}$$

where we take the distribution $P(k_i)$ to be multinomial. A language model is estimated for each document d in the collection D. Then, given a query q, the relevance of a document d can be assessed by the conditional probability $P(d|q)$. Through Bayes' theorem a new retrieval model is derived, in which the documents are ranked proportionally to the posterior $P(q|d)P(d)$. The query likelihood $P(q|d)$ represents the probability of the language model of document d generating the terms in the query q

$$P(q|\theta_d) = \prod_{t \in q} p(t|\theta_d)^{n(t,q)} \tag{2.6}$$

where the exponent $n(t, q)$ denotes the number of occurrences of term t in the query. For the term distribution $P(t|\theta_d)$ we use the maximum-likelihood estimate $P(t|\theta_d) = \frac{n(t,d)}{n(d)}$, while the document prior $P(\theta_d)$ is usually assumed to be either uniformly distributed or based on one of several popularity metrics.

Other methods

A popular competing class of methods uses clustering techniques as an attempt to reduce the time complexity, but these methods have not been shown to be competitive in terms of the predictive performance they yield. There are a myriad of additional methods such as outlier detection [62] and the re-casting of the task as a binary (plagiarized and not-plagiarized) classification problem that can be addressed using standard supervised-learning methods like SVM [10].

Data and evaluation metrics

A single best overall model does not exist, and most of the proposed techniques are variations of the ideas described in the extrinsic plagiarism detection section. A more interesting point is how the methods are evaluated, or in in other words, which data sets are used for benchmarking them. Of particular interest is the question of how the document transformations are simulated in the absence of any large repository of known plagiarized documents, and therefore such documents must be synthetically generated. A description of some of the proposed evaluation data sets are [27]:

- Paraphrasing. A corpus of manually paraphrased documents that were manually created by PhD and MSc students.
- Fairy-tales. As fairy-tales belongs to an oral tradition, having been disseminated through word of mouth instead of being written down, multiple versions of the same fairy tale exist. For this corpus there are slight to strong intra-tale variations.
- Insertions. The insertion of plagiarized phrases into otherwise innocuous content. The plagiarized fraction of each document was kept relatively small.
- Synonyms. The use of synonyms and synonym phrases to obfuscate the plagiarized passages.

- Translations. The use of automated translation and summarization tools to introduce external noise into the process.

Three of these data sets correspond to the same or similar transformations that we have independently proposed and utilized as part of our controlled environment simulations, i.e., translation, insertion (mixing) and spinner (synonyms).

2.1 Rule-based systems

When the data format of the transferable data objects is highly structured such as XML documents with a specific schema, it is not unreasonable to assume that hand-crafted rule-based detection mechanisms and policies can be effective in detecting and deterring inadvertent data leaks. This is similar to how application level security filters operates. However, as this assumption relies on a tightly controlled environment scenario, most DLP research (and commercial products) have focused their attention on more lenient scenarios.

Related research (for generic DLP) addresses e-mail systems in which rules are used to catch erroneous or unnecessary recipients [78] and frameworks for creating pattern-matching rules for the detection of credit card or social security numbers [54]. Other rule-based systems include application-layer proxies, firewalls or security filters such as the Thales Trusted Security Filter TSF201, which filters out the UDP packets, sent from one security domain to another, whose entries not satisfy a specific set of rules, e.g., byte value ranges.

The NATO Communications and Information Agency (NCIA) has proposed a new information sharing infrastructure for NATO called "Content-based Protection and Release" (CPR) [7], which is closely related to the cross-domain information exchange task. While not content checking per se it is an example of a rule-based system, which can be used together with our policy-based labeling framework discussed in Paper I. Here the release or protection decision is determined by policies expressed using attributes (metadata). As an example of such such a policy they include a scenario related to the (simulation of) intercepting missile attacks, in which the rules are as quoted:

- *"A soldier may see sub-munition locations and descriptions while a civilian cannot"*
- *"A soldier taking part in the operations in the area of the Passive Missile Defence simulation can access the map, but a soldier not involved in the mission - even one with a high rank - cannot"*

- *"To access the description of the consequences of intercepting the missile, the terminal should have an enhanced configuration guaranteeing a secure connection and local encryption of data"*

2.2 Data-driven content checking

In order to address the shortcomings of existing content-checkers one can leverage both similarity and machine learning-based techniques in the data object release flow. Although this process may appear be to identical to the extrinsic plagiarism detection setup as explained, it is worth emphasizing the major difference: in this case we are not interested in detecting copied segments for the sake of exposing instances of plagiarism. Instead, our primary concern lies in constructing methods that allow us to detect the presence of any classified information embedded in textual content that is incorrectly marked as unclassified. This allows us to create hybrid methods that operate in a middle ground between external plagiarism detection and sensitivity estimation. More specifically, we simultaneously try to: 1) detect any embedded known sensitive information by comparing the contents of new data objects with that of known classified documents, e.g., a collection of internal classified reports, and 2) use text classification algorithms to more generically determine whether or not a document contains sensitive data at all without attempting to trace it back to an originating source.

While the idea of using machine learning to estimate the security level can be traced back to a master's thesis written in 2008 [20], the pioneering proof-of-concept work that demonstrated the feasibility of building machine learning-based methods for this task was carried out by Brown et al. in a series of military technical reports in 2010 [17, 18]. In their studies, parts of the Digital National Security Archive (DNSA) corpus [1] were used to determine how a classifier trained on documents regarding a particular topic or time-period would generalize to other topics or time-periods. The data set they used contained mostly the plain-text abstract for each document, which did not necessitate the use of a noisy OCR preprocessing phase. Their goal was to develop a tool to aid in the manual classification and declassification of documents (especially in the context of the Canadian Access to Information Act [ATI]), rather than to detect (inadvertent) data leakages.

After our initial work in the area (Paper II, III) there has been additional research published in the field. Researchers at the NATO Communications and Information Agency have performed experiments using both semantic analysis and machine learning-based methods for predicting the security classification

of text documents. Furthermore, they developed a proof-of-concept Microsoft Word plugin to assist users with determining the appropriate label [83]. As a data source for training and evaluation they used declassified NATO documents from the 1950s, for which an OCR process first had to be applied. The work was later expanded upon by the same researchers [71]. In this paper the confidence scores of the machine learning classifier were used to determine which documents that should be manually reviewed by human operators. They also investigate how robust the classifiers are to the noise introduced by the OCR process.

Alzhrani et al. [4] uses the stolen diplomatic cables released by WikiLeaks (diplomatic cables) to build automated security level classification algorithms. Their motivation is the detection of malicious insider behaviour of the form conducted by Edward Snowden and Chelsea Manning. The novelty and main contribution lies in that they perform classification on the paragraph level for multiple security classification levels. The authors expand upon this work in a follow-up paper [5] where additional methods are proposed and evaluated on the same dataset. One of the methods they adapt is to first cluster paragraphs and then perform classifications on a per-cluster basis. While we were aware of the availability of the WikiLeaks data set from the start, we decided not to use it due to ethical and legal concerns given that these are stolen documents whose content has not (despite being leaked) been formally declassified. However, our views on this issue are not universally shared by the scientific community [53].

When it comes to the topic of detecting transformed data leaks, Susilo et al. evaluates an n-gram based statistical method on a set of documents that had been "spun"[1] [3]. In lieu of a data set annotated with real security labels, they instead perform a form of topic categorization, which is an easier classification task, and use the resulting performance metric as a proxy for what can be achieved on the actual task. A paper by Shu et al. [75] introduces a scalable GPU-based sequence alignment algorithm for detecting data leaks in transformed documents. The authors' focus was restricted to a special class of transformations that included replacing white space characters with the "+" character which happens as part of URL encoding. For benchmarking they use a source code repository meant to simulate a scenario in which the code represents strategically important intellectual property that are potentially leaked to third parties through HTTP.

In terms of recent commercial products Amazon Macie was announced and released in mid-August 2017. It is a proprietary system that uses machine learn-

[1] Spinning refers to a search engine optimization (SEO) tactic in which lists of synonyms and synonym phrases are used to alter documents.

ing algorithms (SVM) and regular expression rules to "automatically discover, classify, and protect sensitive data in AWS" [6]. It is able to discover and classify personally identifiable information (PII) and sensitive data such as such as credit card numbers, email addresses and drivers license IDs as well identify and determine the content type of S3 objects such as E-books, `C++` source code and various log files. Each object is assigned a risk level which is determined by the max level associated with the predicted content type, PII status and file extension. Anomaly detection algorithms with respect to S3 object-level API activity are continuously being run and the users are provided with dashboards that display information regarding any anomalous incidents.

3 Insider threat detection

In this section we shift the focus to one of the topics of Paper V: the task of detecting malicious insider activity in the cross-domain setting. Our research entails modelling insiders by looking at consistent long-term trends in the misclassification(s) by the users. This procedure is complicated by the fact that we do not know the true label and we therefore need to rely on the imperfect predictions of the content-checker. It should be noted that a second aspect of the paper is concerned with ways of handling the number incorrect predictions by computing a meta-score based on the output from the existing content checker. The implication being that the proposed method can be used both to detect insiders as well as other entities, such as misconfigured services, that are leaking data.

Insiders

The recent and highly publicized accounts of insider activity, both in terms of leaking classified data (Edward Snowden, Chelsea Manning) as well as industrial espionage (e.g. Chinese engineers [79, 80]), have brought an increased focus on developing measures for detecting and preventing malicious behavior by trusted insiders.

When building models for detecting insider behavior the most common procedure is to utilize an anomaly detection approach. A baseline model is fitted using what is considered *normal* user behaviour, and then any activity that significantly deviates from that standard profile is considered anomalous, and potentially indicative of insider activity. As part of an initiative to better prepare for and handle future insider events DARPA launched the Anomaly Detection at Multiple Scales (ADAMS) program [55]. The program involved two parallel research tracks; one focusing on model development, and the other on the creation of realistic data sets simulating various insider scenarios. These data sets (scenarios) were then used by the participants in the first track to evaluate and benchmark their algorithms. The Carnegie Mellon University CERT (CMU-CERT) insider threat data sets were created as part of the data set track. It consists of the simulated logs for 1000 users and include data such as logon, logoffs, USB events, HTTP traffic logs and e-mail data etc. as well as communications, psychometric and topic models. The topic models were generated from Wikipedia articles [25]. At the end of the ADAMS programme Lindauer et. al [50] reviewed the quality of the dataset as measured by its realism and the degree of usefulness it had provided to the other programme participants. They

concluded that while it had been useful for integration testing and development it proved to be less useful in other regards: 1) the generated text was considered too simplistic and naive to be used for the class of algorithms proposed by participants of the model track, 2) the data was significantly "cleaner" in the sense that there were much fewer inconsistencies in it than what is the case for real-life data and 3) because of the ad-hoc way parameters were chosen it was not possible to verify hypotheses regarding the behaviour of malicious (or non-malicious) users. As our work had an emphasis on text classification we decided not to base our experiments on this data set.

Legg [48] uses the CMU-CERT Insider Threat Dataset [25] to create an interactive model visualization (i.e. visual analytics) tool that allows the security team to investigate and analyze the reasoning of an opaque principal component analysis based insider detection algorithm. It adapts a "Zoom, Filter and Analysis" mindset, in which analysts, when noticing suspicious indicators, can drill down and analyze the past activities of the user in question.

A much more realistic insider data set is studied by Gavai et al. [24]. They collaborate with a large anonymous corporation in order to use real data from the organization' internal operational network. For each user in this network environment a set of features is produced, encoding information such as the weekly number of messages sent, the total time spent on websites for career, entertainment and tech etc. in addition to other social and on-line activity data. These user profiles are then fed to an anomaly detection algorithm (Isolation Forest) that compares current user behavior to past behavior and to that of peers. The Isolation Forest is an iterative algorithm that, in contrast with the methods that first compute a baseline profile or distribution and then identifies outliers, explicitly isolates anomalies. At each step, it randomly samples a feature and a split value between the minimum and maximum values of that feature [51]. The fewer splits needed to isolate a data point, the more likely it is that the point is anomalous.

When it comes to the more specialized task of detecting insiders by analyzing how they interact with sensitive information, little research has been published. However, our work was influenced by the related concept of misusability weights. A misusability weight attempts to quantify, on a per-user basis, the extent of damage that the organization would incur if the data that user has accessed leaked to third parties. The M-Score [22] and the TM-Score [23] algorithms are two such methods tailored for relational and textual data respectively. While the authors of the TM-Score paper do not explicitly mention machine-learning these methods fits perfectly within this category. The TM-

Score algorithm works by first having domain experts annotate a subset of data with numerical sensitivity from a range such as $[1, 5]$ where 5 signifies the most sensitive content. In the second phase these scores and a similarity measure are used to assign scores to the remainder of the data set by using a weighted average of the scores of the nearby (as determined by the similarity measure) documents. This process is reminiscent of applying a k-NN classifier in which there is an additional re-weighting mechanism that takes other aspects such as document decay, exposure time and authorship into account. It is not clear how more traditional machine learning algorithms would fare in this task as while the Enron data set is publicly available, the associated sensitivity scores have not been published by the authors. The TM-Score is then taken as the weighted sum of the scores for the documents that the user has accessed. The weighting factor ensures that the score of duplicate documents are not added twice.

4 Data validation and authorship profiling

As explained in Section 2.1, in some scenarios the system enforces restrictions on the type or format of the data objects that are allowed to be exported across domains. For example, a system might operate with a policy that prohibits any transfer of non-XML documents of a specific XML schema. This idea of validating the data format can be further extended for non-structured data. By constructing data-driven machine learning-based algorithms with the purpose of determining the data type we can increase the accuracy in the correctness of the data object; thereby helping to prevent non-conforming data from being exfiltrated by tunneling through the cross-domain sharing solution by first embedding it within a seemingly valid object. For example, we can envision performing validity checks on the body or field contents of valid XML documents, which effectively constitutes a type of deep data format inspection and validation[2]. Related research for data validation is primarily to be found in the form of traffic identification with respect to the protocol or application [82, 85]. Similar ideas have also been pursued and explored in the context of more generic anomaly detection settings [81]. A reoccurring theme in these papers is to operate directly on the TCP payload by analyzing variations in the byte frequency distributions, which differs from our setting and approach of analyzing the contents from a term-frequency angle.

Authorship profiling

An idea introduced as part of this thesis is concerned with the possibility of exploiting latent author traits to further enhance the performance of existing, more primitive, DLP systems. For example, if we have systems where users are generating text, e.g., cross-domain e-mail or chat, we posit that we can first infer latent user traits such as gender, native language or age from the textual contents of existing email and chat messages. For any new email or messages one could then compute the predicted gender, age and ethnicities of the sender. These predictions could then be taken into account by the DLP system that determines whether or not the outgoing data object contains classified or sensitive information. As a concrete example consider a situation in which a civilian, 21 year old male user is writing emails in a cross-domain system. If he produces a new email which the machine learning system determines (with some confidence score) is written by a female, 60 year old navy officer, then this is clearly

[2] As opposed what can be achieved using regular XML validation.

suspicious and should influence (raise) the score of the DLP system. The task of inferring latent author traits is called author profiling and has mainly been studied as part of digital forensics (e.g. who wrote the threatening blog comment) and marketing (e.g. inferring demographics) [72]. While author profiling is a new field, there has been a sharp increase in research activity with academic contests for inferring the gender and language variety (e.g. distinguishing between Australian, Canadian and United States variety of the English written language) for multiple languages (English, Spanish, Portuguese and Arabic) using social media data (Twitter) [70, 72, 77]. The results have been promising with the overall accuracies for the gender and language task hovering around 0.82 and 0.91 respectively.

There is a connection between our proposed use of author profiling and methods proposed in the field of active authentication[3]. Here, the goal is to create non-password (biometric) based authentication methods, and then continuously re-authenticate the user throughout a session by using these primarily biometric-based credentials (i.e. derived from the unique intrinsic physical or behavioral traits). While the first phase of the DARPA challenge concerns itself with building biometric models that do not require the user to install custom hardware; the second phase shifts the focus to combining the biometric mechanisms in a multimodal fashion. As the features used in different biometric algorithms are statistically independent, combining their predictions will improve recognition and reliability, and will also make spoofing attacks more difficult. Just as we can have a multimodal system that combines subsystems for face, iris and fingerprint recognition [37], we posit that the same principle can be applied to the content checking task by operating with multiple models that each focus on a different classification task such as data type validation, language detection, sensitivity estimation etc., and in Paper VII we have demonstrated this by incorporating the authorship verification signal in a similarity based DLP model. Figure 4.1 shows an illustration of how we may view the architecture of a future multimodal system for DLP.

Authorship verification

As part of the thesis we conducted experiments where these ideas were explored by building authorship verification models for instant messaging data. Authorship verification (AV), a subtask of authorship profiling, may be seen as a somewhat special variant of data type validation in that it uses an author's

[3]This is another DARPA initiated and funded program [44].

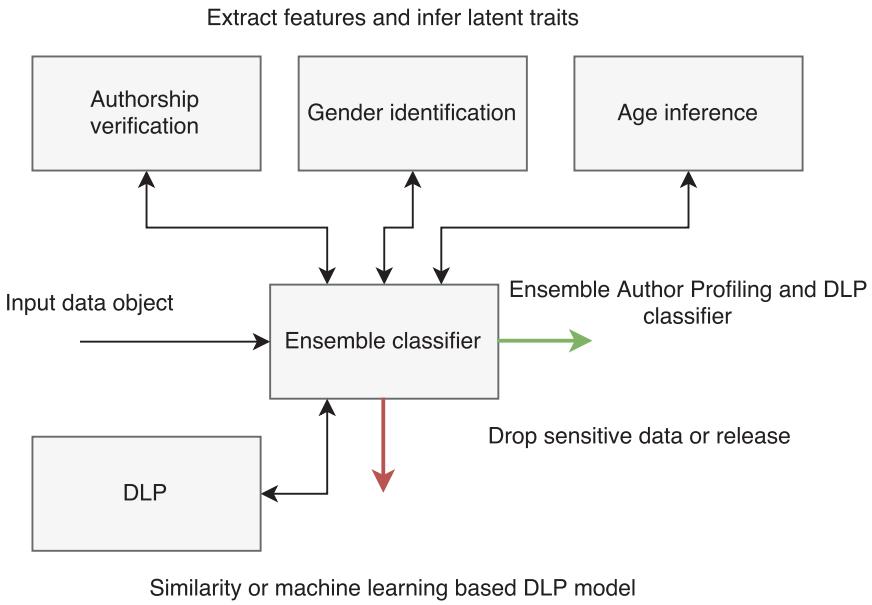

Figure 4.1: An example of a potential multimodal author profiling, data format validation and DLP system.

stylometric profile (as inferred from past documents) to validate authorship for new documents. AV differs from authorship attribution (AA) in that it is not a closed-world problem, i.e., for AA the task is to determine who in the given candidate set is the true author.

The AV task is similar to the intrinsic plagiarism detection task discussed in Section 2, and has an identical setup. It differs only in the fact that past research for AA and AV has mostly been done in the context of literature studies and digital forensics science. The corpora used for these studies are long documents (500+ words [45]); while the previous work for short messages has been conducted primarily using emails (Enron) [16]. Related work has been done in AA for short messages (tweets) but it differs (compared with our work) as follows: 1) each user is associated with a large number of tweets, 2) it assumes a closed-world scenario with a set of candidate twitter users, 3) an English corpus was used, and 4) in the usage of meta-data (profile text, picture, location information, description etc.) [12].

5 Machine learning

In this section we provide a brief introduction to machine learning with a particular emphasize on the challenges faced when developing and deploying machine learning algorithms as security mechanisms.

5.1 Statistical machine learning

While statistical machine learning is a field made up of the four sub-tasks: supervised, unsupervised, reinforcement and rule learning [30], only the supervised approach is used in this thesis as this is the most established branch [30]. Presented with two sets of random variables; the inputs $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ and the corresponding outputs $(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n)$, we seek to determine a function $f(\mathbf{x_i})$ that will accurately predict the outputs $\mathbf{y}_k$ for future input values $\mathbf{x}_k$. The function space in which we seek a predictive function $f(\mathbf{x}_i)$ is commonly referred to as the hypothesis space $H : \{f(\mathbf{x}, \Theta)\}$, where the parameter Θ indexes the space. With the hypothesis space defined, the supervised learning task can be recast as a numerical optimization problem where we seek to minimize a cost function $\Phi(\mathbf{X}, \mathbf{Y}|\Theta)$ that penalizes each incorrect classification.

Definition 1. *Training and test data*
Training data $(\mathbf{X}, \mathbf{Y})$ *is the data used in the optimization process, whereas the testing data* $(\mathbf{X}_*, \mathbf{Y}_*)$ *is a data set that is used to evaluate the predicative performance. The two data sets are disjoint.*

In order to find an approximate minimum for $\Phi(\mathbf{X}, \mathbf{Y}|\Theta)$ one can introduce a proxy objective function $E(\Theta)$ representing the empirical risk. $E(\Theta)$ is a linear combination of two dueling objective functions: a loss function ℓ that approximates the true cost, and the regularization term ρ that penalizes overfitting:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, E(\Theta) = \underset{\Theta}{\operatorname{argmin}} \sum_{(x,y)} \ell(y, f(x)) + \lambda\rho(f) \tag{5.1}$$

An example of a supervised algorithm, which we have used extensively as part of our work, is the support vector machine (SVM) class of methods. The multiclass SVM works by searching for functions (one for each of the C classes) whose score for the correct class $f(w_{y_i}, x_i)$ is at least 1 higher than the other classes. For the linear SVM we have:

$$f(w_j, x_i) = \mathbf{w}_j^T \mathbf{x}_i \quad \text{for } j \in \{1, ...C - 1\} \tag{5.2}$$

Mathematically this is expressed by the non-convex minimization problem, which scales to very large data sets [13]:

$$\min_{\mathbf{W}} L(\mathbf{W}) = \underbrace{\frac{1}{M} \sum_{i} \sum_{j \neq y_i k} \max(0, \mathbf{w}_j^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + 1)}_{\text{data loss}} \quad + \quad \underbrace{\frac{\lambda}{2} \|\mathbf{W}\|^2}_{l_2 \text{ regularization loss}} \tag{5.3}$$

When the classes are not linearly separable in the input feature space we can project the features to a higher dimensional space where the target classes are more linearly separable. Instead of directly transforming each feature, we can employ the *kernel trick* where we directly compute the inner-product $K(x, x')$ (similarity) between two data points in said higher dimensional space [13]. One choice for such a kernel is the Gaussian RBF kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{5.4}$$

where the hyperparameter $\gamma = \frac{1}{2\sigma^2}$ must be determined empirically during cross-validation. For the one-class SVM the training set consists of instances of only the positive class and the goal thus becomes to separate these data points from the origin. In lieu of target classes, the one-class SVM inference process takes a hyperparameter ν which specifies an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors with respect to the training set size. [67].

For subsequent predictions (probability) confidence values can be computed by using Platt scaling effectively mapping $(-\infty, \infty)$ to $[0, 1]$ [69]. Other commonly used supervised machine learning methods include: Random Forest [14], Adaboost [87] and logistic regression [13].

5.2 Secure machine learning

The central assumption behind statistical machine learning is that the inputs and labels are independent and identically distributed random variables from an unknown joint probability distribution $P(\mathbf{x}, \mathbf{y})$ [30]. Furthermore, it is assumed that this distribution is stationary [65]. Under these conditions, minimizing the empirical risk on the smaller training data set, which has often been painstakingly hand labeled, is equivalent to minimizing the risk on the larger evaluation data set. However, this assumption is violated for security tasks such

as intrusion detection and DLP systems, where one must take into account the possibility of attackers actively seeking to by-pass detection by manipulating the classifier itself. A machine-learning algorithm is said to be *secure* if it performs adequately when deployed in adversarial conditions. Security assessments of machine-learning systems may be conducted with respect to the three axis [11] as follows:

- ***Influence***: A user can influence the learning system by conducting either: a causative or an exploratory attack. A causative attack (interchangeably: poisoning) refers to manipulating (parts of) the training data with the intention of exerting control over the learning process. An exploratory attack involves inducing and exploiting a misclassification by rewriting a classified document to circumvent detection by the content-checker, for example.

- ***Security Violation***: Security violations, with respect to a machine learning based DLP system, take on one of two forms: integrity (e.g. sensitive content being incorrectly classified, which in the DLP setting would result in a policy violation and a form of confidentiality breach) and availability (e.g. non-sensitive data being misclassified en masse, thus rendering the system useless similar to a denial of service attack).

- ***Specificity***: This refers to the scope of the attack, which can either be a targeted or an indiscriminating attack.

We will in the following sections further elaborate on the different forms of attacks, i.e., the causative, exploratory and model inversion attack variants.

Causative attacks

Proposed mitigation measures against causative attacks, i.e., attacks in which the adversary tries to influence the training procedure, include the use of algorithms that effectively sanitize the data by modifying the learning process to dynamically discount those data points in the training set that have a significant negative impact on the performance [59]. A competing class of defense mechanisms recasts the problem as one of anomaly detection, such as whether the model parameters of one user deviate dramatically from the model parameters of other users. Another proposed approach draws on methods from the field of robust statistics [40]. However, this approach assumes that only a small fraction of the training set has been contaminated.

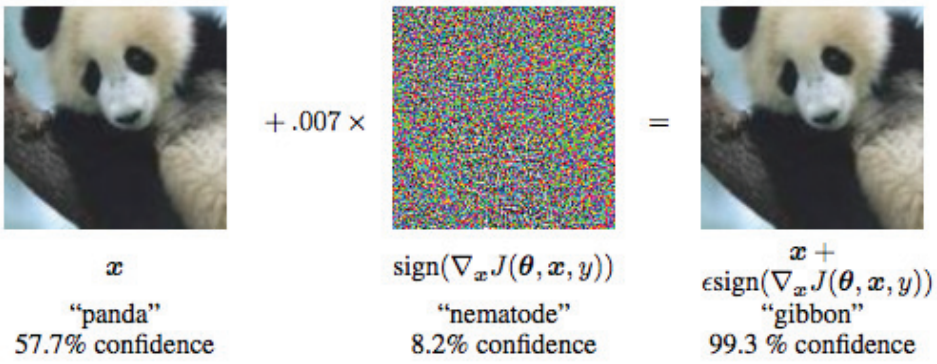

Figure 5.1: Generating adversarial images. Figure from "Explaining and Harnessing Adversarial Examples" by Goodfellow et al. [31].

Exploratory attacks

The analysis of exploratory attacks has attracted an increasing interest from researchers in recent years. This surge can at least partly be attributed to the rise and popularity of deep neural networks, especially convolutional neural networks applied to computer vision related tasks, and the discovery of "adversarial images". An adversarial image (more generally known as an "adversarial example") is an image that has been synthetically altered so that it is misclassified with high confidence, while still appearing unaltered to a human observer. Figure 5.1 shows an example of how the image of a panda can be perturbed to be misclassified as a gibbon (a smaller ape). Further investigations have shown how adversarial images can be automatically constructed [66] and that, contrary to early misconceptions, their existence is not caused by non-linearities [31]. Successful black-box attacks have been carried out by exploiting transferability, whereby the adversarial examples generated using one classifier is likely to carry over to another classifier [63, 64]. One proposed defense mechanism suggests generating and using adversarial samples during training as an additional type of data augmentation transformation (scaling, rotation etc.). It is more difficult to automatically generate adversarial examples for textual data, as for this type of data it is problematic to enforce the constraint for the altered document or sentence to be semantically identical with the source document or sentence while also being syntactically meaningful. That being said, it is still feasible to perform manual exploratory attacks, and automatic attacks are

routinely carried out by spammers, but humans can often detect these easily.

Model inversion attacks and data leaks

A separate class of attacks, which is of particular concern for classifiers trained using sensitive or classified data sources, is the possibility of recovering data from the training set by performing probing attacks using the inferred model. Proof of concept attacks have demonstrated the possibility of extracting (parts of) data points from the training set [8, 28, 29, 33, 38]. These attacks are possible because the internal structure; that is the model weights, capture information about the training data. A proposed defensive approach would be to alter the learning algorithm to become privacy preserving. However, this latter approach remains an active research area [41], and has mostly focused on protection against personally identifiable information (PII) leaks. Studies on attacks have mostly been in the context of collaborative machine learning tasks, such as systems in which each participant contributes to a common learning task. An attack on such a collaborative system in which a digit classifier was collaboratively trained using the MNIST data set (handwritten numbers) is discussed in depth in [38].

6 Generative processes and probabilistic models

As we rely on a generative modeling and probabilistic programming approach in our work to detect insiders (Paper V), we will provide a brief introduction to this topic.

Generative models

A mathematical model is a simplified representation of a system expressed using mathematical terms. Probabilistic modeling is a sub-branch of mathematical modeling in which statistics is used to express all uncertainty (and noise) that is associated with the model's components and the strength of the relationships between, both unobserved and observed, variables. In contrast with deterministic models the inference procedure does not result in a single best-estimate solution, but describes the (model) world in terms of probability distributions.

A generative model tries to capture how the underlying processes have generated the observed values. These generative processes are described in terms of joint probability distributions over the possible states of the simplified world we model. After having defined a model we can use the resulting joint probability distribution to perform generic queries (e.g. given that these specific conditions hold, then what is the likelihood of this other set of conditions) and we can update the distribution (i.e. compute the posterior distribution) once new data becomes available. Furthermore, such a model can be utilized to generate (sample) new data points, which is what gives rise to term *generative model*.

A special instance of a generative model is the Bayesian network (BN). In a BN the directional dependencies between the random variables (nodes) are explicitly defined (and often signify causal relationships). For each node in the graph, one must specify the conditional probability distribution (CPD) given the values of the parent nodes. Figure 6.1 shows a graphical representation of such a model with the random variables: (I)nsider, (M)islabel, (U)ser, (L)abel and (S)core. In order to complete the BN model, we must determine the functional form of the CPDs and the prior distributions:

$P(\text{Mislabel}\|\text{Insider})$	(CPD for Mislabel given Insider)
$P(\text{User}\|\text{Label}, \text{Mislabel})$	(CPD for User given Label and Mislabel)
$P(\text{Score}\|\text{Label})$	(CPD for score given L)
$P(\text{Insider})$	(Prior for Insider)
$P(\text{Label})$	(Prior for Label)

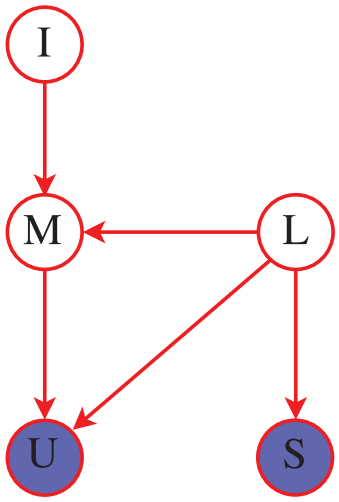

Figure 6.1: A graphical model describing the dependencies between a set of random variables. The purple nodes are those we observe. Inference algorithms can recover the latent distributions from the set of observed values.

the inference algorithm then infers the posterior, which allows us to execute generic queries. The implementation of complex models is an error-prone, time-consuming and specialized task [26]. As our work on the detection of insider behaviour using score discrepancies is a novel and exploratory research question, we decided to use a probabilistic programming language instead of manually implementing each model that we experimented with.

Probabilistic programming

While in the past each model had to be painstakingly implemented by hand or using a domain specific language created for certain sub-classes of models, it is now possible to utilize a probabilistic programming language (PPL) instead. Using this PPL, the probabilistic model can be expressed using a general-purpose Turing complete programming language and utilize all the familiar programming constructions such as object oriented programming and recursive functions. The analogy here is that the probabilistic model is expressed as a programming language that is subjected to noise, i.e., the program is the generative process. This paradigm allows the implementation of more flexible models, that are much easier to integrate with any existing/external system modules. Furthermore, it

allows non-expert machine learning users who posses a high degree of domain expertise to create data-driven models.

7 Summary of papers

In this section we summarize each paper and highlight our contributions. For further details, readers should refer to the papers themselves.

Paper I - Policy-Based Labelling: A Flexible Framework for Trusted Data Labelling

Security labels are utilized for several applications including information flow control in cross-domain information exchange. In order to better assure the correctness of the security label in such as setting, Paper I introduces the concept of policy-based labelling as an attribute-based access-control (ABAC) inspired approach to data labelling. This approach deploys a policy-based labelling solution that specifies a set of rules (i.e. polices expressed in terms of the attributes associated with the entities involved, such as the user and document in question). These rules clearly state the conditions for when or how a given security label is appropriate for a specific data object (i.e. document) and help to increase the trust in the correctness of the assigned security labels.

The outcome decision for an ABAC request is a question of whether or not the access is permitted, while the policy-based labeling system either 1) determines if the value of the security label for the data object is acceptable or not, or 2) produces a new set of output attributes, e.g., a less lenient security label for the data object, through the use of the obligation mechanism. Policy-based labelling allows us to express data-labelling polices using object, subject and environment attributes, and the resulting system can be used both in a fully automated labeling setting, determining the label automatically for all data objects; or as a detection mechanism that inspects and asserts the validity of user-assigned labels and then drops or flags suspicious export requests as necessary. The flexibility of the proposed framework and the underlying policy language allows for the creation of general purpose programming constructions that can extract attributes (using attribute modules), impose conditions on the labeling decision, perform the data object and label binding or notify/alert the security teams.

A proof-of-concept implementation of the framework was developed in order to experiment with these ideas. It was built using the open-source XACML 3.0 implementation Balana [84] and implemented in Clojure so as to run on the JVM. Policies and attribute modules were created for a wide range of tasks including: dirty-word-list based content-checking, XML schema validation, digital signature verification (implemented using a custom attribute module). Obliga-

tion handlers were created for sending emails (security alerts) and for changing, i.e., increasing, the level of the proposed security label.

Paper II - Automatic Security Classification by Machine Learning for Cross-Domain Information Exchange

In Paper II we discuss how to build machine learning-based classifiers for predicting the security classification level of text documents. As a data source for the training, test and validation data sets we used scanned PDF documents retrieved from the online repositories of declassified historical U.S. documents that have been made available through the ongoing efforts of the U.S. Digital National Security Archive. The documents were then transformed into plain-text format through the use of a commercial OCR provider. The subset of data we used contained documents regarding different domains (countries) over separate time-periods. While the security labels for the documents range from Unclassified to Secret we conflate the classification levels into Classified and Unclassified in order to make the task amenable to a binary classifier as is appropriate for a guard setting in which we seek to determine whether or not to release a data object. For the classifier we use a linear SVM, which we train using the documents' TF-IDF weights and a BoW model as features.

Experiments were carried out where we compared the performance measured in terms of accuracy, when we train and evaluate on each subset (domain) in isolation without the presence of "leaky keywords" (e.g. *Secret*, *Classified*, *Unclassified*), while preserving the chronological order and when using only a plain-text abstract for each document. Finally, we experiment with a two-phase algorithm in which the documents are first clustered, and then classifiers are trained on each cluster independently in an attempt to simulate the approach where we train on a per-domain (country) basis.

The relationship between Paper I and II is that the proposed methods can be utilized as attribute modules, whose output values or predictions can be subsequently used in the policies.

Paper III - Automatic Security Classification with Lasso

Paper III extends Paper II by replacing the SVM model with a logistic regression model with an added l_1 regularization term. The use of the L1 norm induces sparsity in the resulting weights, which means that most of the weights of the trained model are (nearly) zero. Consequently, we (ideally) end up with a set of

non-zero weights whose associated words or tokens can be considered as data-derived dirty-words. It can also be considered as a type of feature selection routine that aids the human analysts to interpret the end model.

We also compare the performance, measured in terms of accuracy, against that of the k-NN, SVM and Naive Bayes models. Experiments show that the l_1-regularized logistic regression model yields comparable performance, and even outperforms the other models under some conditions.

When varying the strength of the regularization term, the number of non-zero weights changes, and so we can analyze the trade-off between the number of (effective) features and the performance of the classifier. Analysis shows that we need to retain at least 459 features (as compared with the *thousands* for the l_2 regularized counterparts) to achieve optimal performance, and that even with as few as 50, the performance is shown to be surprisingly good. These resulting words can be considered as the components of dirty (and clean) word lists. Further, we experiment using multiple security classification levels, such as Unclassified, Restricted, Classified and Secret by replacing the sigmoid with the softmax function.

Paper IV - Data Loss Prevention Based on Text Classification in Controlled Environments

In Paper IV we adapt the machine-learning classifiers introduced in Paper II (and extended in Paper III) to the cross-domain information exchange and DLP scenario and we also experiment with information retrieval (similarity-based) methods. This setting differs from the more generic one studied previously, in which the goal was merely to infer sensitivity or classification levels for the textual content itself. Instead, we here look at the specific scenario where the users of a computer network are able to 1) import documents of known classification in what we call *controlled environments* and 2) export (potentially new and/or modified) documents after having assigned a user-specified security label. A controlled environment refers to any environment where we have control on all imported documents and their respective security classification, and it could be a computer, virtual machine, network domain or an isolated process or application, e.g., container.

For the two previous papers (as well as in other past studies) the annotated sets of documents were separated into disjoint training, test and validation data sets, implying that each document was a member of one such set. Here, on the other hand, we simulate how documents (or rather the information they contain) are transformed into new documents as a result of combining information from

multiple sources into a new report or by summarizing a larger block of work, for example. This simulation gives a more realistic view of how (un)classified material is processed. As cross-domain content checking can be considered a sort of hybrid task that combines the task of extrinsic plagiarism detection with sensitivity level estimation, it is beneficial to try and exploit this factor in our solutions. By keeping track of the data (and its security classification levels) that a user has accessed within the context of their controlled environment, we use these subsets of documents to train per-user/controlled environment classifiers. Experiments indicate that the more detail we have on the information a user has accessed the better the predictions are. This interesting and somewhat surprising results seems to contradict the conventional machine-learning wisdom that more data is always better; and this phenomena is explained by the hybrid nature of the task, which is mixture of sensitivity estimation and extrinsic plagiarism detection.

In order to validate that our results also hold water in an operational setting, i.e., that the observed boost in performance is not attributed to the artificial way we generate output documents, we conducted experiments using a separate data set consisting of a private repository of classified and unclassified reports from our research institution. For each of these reports we assumed that the references were the documents imported into the controlled environment while the report itself was the output at the end of a session. We also compared datasets derived from the DNSA collection for the which the granularity of the associated controlled environment varied from: 1) a per-user level (tight control) 2) domain specific (topic) and to a 3) global setting (all available documents).

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

While the machine learning-based content checking approaches have yielded promising results, there were still a significant number of false alarms even when the controlled environment setting (introduced in Paper IV) was deployed. Furthermore, with the increase in reports of insider-related security incidents, it would be helpful to be able to equip security teams with quantitative risk estimates on a per-user granularity level. For these reasons, in Paper V we derive a meta-score (insider/internal threat score [ITS]) that uses the aggregated output from DLP systems to detect and flag any behavior indicative of data leakage. This method also allows us to better handle the large number of incorrect classifications (false alarms) that one would encounter in an operational environment.

In order to model the way users choose security labels, we used a generative

model whose random variables represent: the user-assigned label (U), the predicted score (S), the true label (L), whether or not the document was mislabeled (M) and a variable that describes the user's overall propensity to mislabel any document (I). The observed values (i.e. U and S) are then used to infer the distributions for the latent (unobserved) variables. In doing so we can use the expected value of I as the likelihood of a user mislabeling a document.

Experiments were conducted that showed how using the inferred values of L (estimated label) resulted in an enhanced performance when compared with using the predicted scores (e.g. using the score with a 0.5 threshold as in Paper II) and/or the user-assigned labels. We also studied the security aspects of the model and proposed and experimented with an anomaly detection-based anti-manipulation algorithm. Further, the practical utility of the score was demonstrated through a discrete-event simulation and visualization that provided security teams with a situational overview of day-to-day changes in the computed insider threat score.

Paper VI - Data Leakage Prevention for Secure Cross-Domain Information Exchange

The paper is a journal article that ties together the contributions of our previous work. It also puts that work into context by providing a discussion of a proof-of-concept data guard that was developed in a joint research effort with Thales Norway, and it offers a more comprehensive treatment of the security aspects of machine-learning based content checking security mechanisms.

Paper VII - Data Loss Prevention for Cross-Domain Instant Messaging

For Paper VII we introduced an anomaly detection based data validation security mechanism for cross-domain chat and we investigated the effect of incorporating latent author traits in similarity based DLP methods. More specifically, we created a cascading classifier for inspecting and validating the payload of chat messages in (military) instant messaging, with the first step being an anomaly detection-based method whose purpose is to ensure that the message channel is not used to exfiltrate non-message data such as images, documents, binary files or encrypted content. The messages that pass this initial filtering phase then proceed to have their contents analyzed for the presence of known sensitive information. By first constructing authorship verification models which

we combined with output from the DLP model we were able to significantly improve the accuracy for sparse messages.

Experiments were conducted using message traffic that was generated during a field training exercise conducted by members of the armed forces, an internal repository of classified documents and a myriad of non-message based data sources. The results demonstrated that our proposed traffic filtering classifier was successful in distinguishing between legitimate and illegitimate traffic. Further, the experiments proved that it is possible to derive per-user stylometric profiles from sparse messages and that incorporating this signal in a traditional data loss prevention solution leads to a significant increase in the predictive performance. This is an important result as it suggests that further improvements in DLP, in such scenarios, can be obtained by introducing other latent author traits.

8 Conclusion and future work

The topic of the thesis falls under the goal of ensuring the security of cross-domain information exchange across different security domains. The ability to efficiently and securely exchange information in this manner is a crucial capability and requirement for conducting modern coalition-based military operations. Also within the parts of the civilian sector where sensitive data is being processed, these solutions can be deployed to network previously isolated computer systems in a secure way or to better secure existing interconnections.

While there exists high-assurance data guards that enforces security policies by validating the integrity and appropriateness (with respect to the release criteria) of user-assigned security labels, only rudimentary security mechanisms exists for asserting their correctness. The crux of the thesis lies in establishing improved methods for validating the correctness of the assigned security label. In particular, we have investigated the use of rule-based and data-driven methods for detecting instances where export requests have been too leniently classified and would represent a data leakage. These detection mechanisms are primarily based on machine learning methods that learn both how to predict the sensitivity level of textual data and how to keep track of from where the data originates. The prediction confidence scores, produced by the algorithms, provides an indication of the likelihood that the respective data objects contains sensitive data. A high score can either result in the export request being instantaneously dropped or alternatively, the scores can be used for triaging the labeling decisions for subsequent manual review by trained personnel.

We have adapted and analyzed machine learning algorithms to the cross-domain information exchange and DLP scenario, in which the methods are deployed in adversarial environments where the security mechanism itself is liable to be a target for adversaries.

We have also investigated how the models' output predictions can be used to construct meta-models that analyze how a user assigns security labels to documents. By observing long-term behaviors and discrepancies between the user- or service-assigned and the predicted security labels, we can detect insider behavior and also better handle the number of incorrect classifications (false alarms), thus further assisting in the triage effort.

Lastly, the data-driven approach has been extended to encompass the assertion of the data format validity of the exported objects. More specifically, we have developed methods for performing authorship verification on short, informal and conversational chat messages and we have developed anomaly detection based traffic filtering algorithms for cross-domain instant messaging systems.

Future work

In this section we outline some ideas on potential topics for promising future research directions.

Deep learning

Deep neural network learning (DNN) methods have revolutionized how machine learning is done for many computer vision [46] and NLP tasks [21]. Instead of manually crafting features, we can instead learn these representations directly from the data. However, in order to achieve state-of-the-art performance, it is necessary to have access to very large data sets [47]. Unfortunately, for the classification tasks discussed in this paper, on security level classification and more generally on content-based DLP, no such collections were available[4]. Furthermore, the data sets used were quite noisy and much of the sentence structure was lost during the OCR process. Although the data set we used were significantly larger than those previously studied, the DNN methods are known to thrive only when trained with especially massive data sets. While there are larger collections of higher-quality data available (most notably WikiLeaks), the ethical and legal ramifications of using these are unclear and often controversial [53]. Recently, very large collections of declassified documents have been made available [19], but these are still only available in PDF format and thus need to be processed by OCR before they are amenable to further machine learning-research. While preliminary experimentation using DNN architectures such as recurrent neural networks (RNN) together with related concepts such as word-embedding layers (i.e. word2vec and GloVe [56, 68]) did not result in any noticeable performance increase (compared with a baseline generalized linear model with TF-IDF features), exploring the application of DNN to DLP is still a worthwhile endeavour.

Multiple modalities

Neither the content checking nor the authorship verification models were able to perform classifications flawlessly, which in practice often means there will be a lot of false alarms. It is unclear how well these machine learning models can be made, and if we assume that an upper-bound is approximately equal to that of the human performance, there will be (at least for the authorship task) a substantial amount of misclassification. A similar problem is faced in the field of

[4]Refer to the introduction for an overview regarding the publicly available data sets.

active authentication. In order to address this problem researchers have opted for a multimodal approach; combining different tasks in order to increase the predictive performance.

Security and privacy

Secure machine learning is a field still in its infancy and much is still desired before classifiers trained using classified data can be routinely deployed in practice. Most research in this field has been conducted either using image data or with respect to preserving the privacy of certain PII information. It would therefore be interesting to investigate how much training data can be recovered when the model is trained using textual data in a generic DLP setting and how this amount changes when using a differential privacy preserving classifier.

References

[1] Digitial National Security Archive. "http://nsarchive.chadwyck.com/home.do", 2016. Accessed: 2015-03-26.

[2] A. M. E. T. Ali, H. M. D. Abdulla, and V. Snasel. Survey of plagiarism detection methods. In *Modelling Symposium (AMS), 2011 Fifth Asia*, pages 39–42. IEEE, 2011.

[3] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy. A semantics-aware classification approach for data leakage prevention. In W. Susilo and Y. Mu, editors, *Information Security and Privacy*, volume 8544 of *Lecture Notes in Computer Science*, pages 413–421. Springer International Publishing, 2014.

[4] K. Alzhrani, E. M. Rudd, T. E. Boult, and C. E. Chow. Automated big text security classification. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 103–108. IEEE, 2016.

[5] K. Alzhrani, E. M. Rudd, C. E. Chow, and T. E. Boult. Automated us diplomatic cables security classification: Topic model pruning vs. classification based on clusters. *arXiv preprint arXiv:1703.02248*, 2017.

[6] Amazon. https://aws.amazon.com/macie/, aug 2017. URL https://aws.amazon.com/macie/.

[7] A. Armando, M. Grasso, S. Oudkerk, S. Ranise, and K. Wrona. Content-based information protection and release in nato operations. In *Proceedings of the 18th ACM symposium on Access control models and technologies*, pages 261–264. ACM, 2013.

[8] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[9] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[10] J.-P. Bao, J.-Y. Shen, X.-D. Liu, H.-Y. Liu, and X.-D. Zhang. Semantic sequence kin: A method of document copy detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 529–538. Springer, 2004.

[11] M. Barreno, B. Nelson, A. D. Joseph, and J. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[12] M. Bhargava, P. Mehndiratta, and K. Asawa. Stylometric analysis for authorship attribution on twitter. In *International Conference on Big Data Analytics*, pages 37–47. Springer, 2013.

[13] C. M. Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *ACM SIGMOD Record*, volume 24, pages 398–409. ACM, 1995.

[16] M. L. Brocardo, I. Traore, S. Saad, and I. Woungang. Authorship verification for short messages using stylometry. In *Computer, Information and Telecommunication Systems (CITS), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[17] J. D. Brown and D. Charlebois. Security classification using automated learning (scale): Optimizing statistical natural language processing techniques to assign security labels to unstructured text. Technical report, DTIC Document, 2010.

[18] J. D. Brown and D. Charlebois. Security classification using automated learning (scale). Technical report, DRDC Ottawa CR, 2010.

[19] Central Intelligence Agency. Crest: 25-year program archive, Jan 2017. URL `https://www.cia.gov/library/readingroom/collection/crest-25-year-program-archive`.

[20] K. Clark. Automated security classification. Master's thesis, Vrije Universiteit, 2008.

[21] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

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

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

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

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

[26] K. Fisher. Probabilistic programming for advancing machine learning. URL `http://ppaml.galois.com/wiki/raw-attachment/wiki/Presentations/PPAMLKickoffOverviewSlides.pdf`.

[27] M. Franco-Salvador, I. Bensalem, E. Flores, P. Gupta, and P. Rosso. Pan 2015 shared task on plagiarism detection: Evaluation of corpora for text alignment. In *Volume 1391 of CEUR workshop proceedings CLEF and CEUR-WS. org*, 2015.

[28] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security*, pages 17–32, 2014.

[29] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[30] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[32] T. Gottron. External plagiarism detection based on standard ir. technology and fast recognition of common subsequences. In *Notebook Papers of CLEF 2010 LABs and Workshops*, 2010.

[33] J. Graham-Cumming. How to beat an adaptive spam filter. In *Presentation at the MIT Spam Conference*, 2004.

[34] D. Gugelmann, P. Studerus, V. Lenders, and B. Ager. Can content-based data loss prevention solutions prevent data leakage in web traffic? *IEEE Security & Privacy*, 13(4):52–59, 2015.

[35] R. Haakseth, N. A. Nordbotten, Ø. Jonsson, and B. Kristiansen. A high assurance cross-domain guard for use in service-oriented architectures. In *Military Communications and Information Systems (ICMCIS), 2015 International Conference on*, pages 1–8. IEEE, 2015.

[36] D. J. Hand et al. Classifier technology and the illusion of progress. *Statistical science*, 21(1):1–14, 2006.

[37] F. He, Y. Liu, X. Zhu, C. Huang, Y. Han, and Y. Chen. Score level fusion scheme based on adaptive local gabor features for face-iris-fingerprint multimodal biometric. *Journal of Electronic Imaging*, 23(3):033019–033019, 2014.

[38] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. *arXiv preprint arXiv:1702.07464*, 2017.

[39] T. C. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. *Journal of the Association for Information Science and Technology*, 54(3):203–215, 2003.

[40] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[41] Z. Ji, Z. C. Lipton, and C. Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

[42] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[43] J. Kasprzak and M. Brandejs. Improving the reliability of the plagiarism detection system. *Lab Report for PAN at CLEF*, pages 359–366, 2010.

[44] A. Keromytis. Active authentication, apr 2017. URL `http://www.darpa.mil/program/active-authentication`.

[45] M. Koppel, J. Schler, and E. Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(Jun):1261–1276, 2007.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[47] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

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

[49] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[50] B. Lindauer, J. Glasser, M. Rosen, K. C. Wallnau, and L. ExactData. Generating test data for insider threat detectors. *JoWUA*, 5(2):80–94, 2014.

[51] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 413–422. IEEE, 2008.

[52] C. D. Manning, H. Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[53] G. J. Michael. Who's afraid of wikileaks? missed opportunities in political science research. *Review of Policy Research*, 32(2):175–199, 2015.

[54] Microsoft. How dlp rules are applied to evaluate messages. 2015. https://technet.microsoft.com/en-us/library/dn329050(v=exchg.150).aspx.

[55] A. S. Mikhayhu. Anomaly detection at multiple scales. 2012.

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[57] R. Mogull and L. Securosis. Understanding and selecting a data loss prevention solution. *Technicalreport, SANS Institute*, 2007.

[58] R. M. A. Nawab, M. Stevenson, and P. Clough. External plagiarism detection using information retrieval and sequence alignment-notebook for pan at clef 2011. In *Proceedings of the 5th International Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse*. Sheffield, 2011.

[59] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 8:1–9, 2008.

[60] NEXOR. Information exchange gateways: The evolving storycopyright, may 2017. URL `https://www.nexor.com/wp/wp-content/uploads/2017/05/Information-Exchange-Gateways-The-Evolving-Story.pdf`.

[61] G. Oberreuter, G. LHuillier, and J. D. Velásquez. Approaches for intrinsic and external plagiarism detection.

[62] G. Oberreuter, G. LHuillier, S. Ríos, and J. Velásquez. Outlier-based approaches for intrinsic and external plagiarism detection. *Knowlege-Based and Intelligent Information and Engineering Systems*, pages 11–20, 2011.

[63] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[64] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[65] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[66] N. Papernot, P. McDaniel, A. Swami, and R. Harang. Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 49–54. IEEE, 2016.

[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12 (Oct):2825–2830, 2011.

[68] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[69] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[70] F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast, and B. Stein. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. *Working Notes Papers of the CLEF*, 2016.

[71] M. Richter and K. Wrona. Devil in the details: Assessing automated confidentiality classifiers in context of nato documents. 2017.

[72] P. Rosso. Author profiling 2017, 2017. URL `http://pan.webis.de/clef17/pan17-web/author-profiling.html`.

[73] C. C. Serena, I. R. Porche III, J. B. Predd, J. Osburg, and B. Lossing. Lessons learned from the afghan mission network: Developing a coalition contingency network. Technical report, DTIC Document, 2014.

[74] A. Shabtai, Y. Elovici, and L. Rokach. *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.

[75] X. Shu, J. Zhang, D. Yao, and W.-C. Feng. Fast detection of transformed data leaks. *IEEE Transactions on Information Forensics and Security*, 2016.

[76] N. Singla and D. Garg. String matching algorithms and their applicability in various applications. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(6), 2012.

[77] A. Stolerman. *Authorship verification*. PhD thesis, Drexel University, 2015.

[78] T. Takebayashi, H. Tsuda, T. Hasebe, and R. Masuoka. Data loss prevention technologies. *Fujitsu Scientific and Technical Journal*, 46(1):47–55, 2010.

[79] B. Vielmetti. Researcher in medical college theft case is sentenced, Aug 2013.

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

[81] K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *RAID*, volume 4, pages 203–222. Springer, 2004.

[82] Z. Wang. The applications of deep learning on traffic identification. *BlackHat USA*, 2015.

[83] K. Wrona, S. Oudkerk, A. Armando, S. Ranise, R. Traverso, L. Ferrari, and R. McEvoy. Assisted content-based labelling and classification of documents. In *Military Communications and Information Systems (ICMCIS), 2016 International Conference on*, pages 1–7. IEEE, 2016.

[84] WSO2. WSO2 Balana implementation. `https://github.com/wso2/balana`, 2014.

[85] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 250–257. IEEE, 2005.

[86] M. Zechner, M. Muhr, R. Kern, and M. Granitzer. External and intrinsic plagiarism detection using vector space models. In *Proc. SEPLN*, volume 32, pages 47–55, 2009.

[87] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.

Paper I

Policy-Based Labelling: A Flexible Framework for Trusted Data Labelling

Policy-Based Labelling: A Flexible Framework for Trusted Data Labelling

Kyrre Wahl Kongsgård, Nils Agne Nordbotten, and Stian Fauskanger

Abstract

Security labels are utilized for several applications. For instance, cross-domain information exchange can be enabled by associating security labels with data objects and enforcing cross-domain information flow control based on these labels (e.g., using guards). The correctness of the security labels is critical to the overall security of such solutions. To assure the correctness of security labels, this paper proposes a flexible framework for trusted information labelling. The proposed solution represents a novel application of attribute based access control (aka. policy- based access control) principles to data labelling. The proposed framework can utilize content verification/analysis, user/application input, information flow monitoring, and contextual information as a basis for its policy-based labelling decisions.

1 Introduction

Security labels provide the means to associate a set of security attributes with a specific information object [7]. Examples of label attributes may include the confidentiality classification of a data object or its handling restrictions. While security labels are often concerned with confidentiality, such as the XML Confidentiality Label [3], security labels can also be used for aspects such as integrity and availability. In the view of attribute based access control (ABAC) and object-level security, a security label may contain any type of attribute relevant for security decisions.

Consequently, security labels may be utilized for many purposes, including access control and cross-domain information exchange. In the latter case, a guard may be used to enforce cross-domain information flow control by releasing or blocking data objects based on their associated security label. Independent of the exact type of properties specified by the label attributes, security labels are trusted to be correct when security decisions are based upon them. For

instance, if a guard releases information from a classified domain to an unclassified domain, based on security labels stating that the information is unclassified, those security labels are trusted to be correct. Thus, the overall security of such a cross-domain solution does not only depend on the security of the guard, but also on the correctness of the security labels.

While the integrity and authenticity of a security label (and its associated data object) can be protected during transport and storage, e.g., using a cryptographic mechanism such as a digital signature, this does not guarantee that the security label is correct in the first place.

In general, one can say that there are two main bases for determining the correct label attributes (e.g., confidentiality classification) of a data object:

- One is to assess the actual content of the data object to determine the correct attributes of the security label. This can either be based on human judgement (e.g., by the author of the data object or an operator performing a human review) or be based on some type of automated content inspection (e.g., dirty word scanning or a more advanced content analysis).

- The other is to base the properties on the origins of the information in the data object. For instance, it may be known that a specific sensor only produces data of a specific type with a given classification. Likewise, a Restricted domain should not contain any data classified above Restricted, and a subset of an Unclassified text should be Unclassified.

In some scenarios, it may be acceptable to simply have the originating user or application specify the label attributes of the data object. However, if there is a significant security risk (e.g., interconnecting two domains with a large security span), higher assurance in the correctness of security labels is required.

One issue is that human users or applications may mislabel data by mistake, or even that an insider could intentionally mislabel data. A more fundamental problem with regard to the assurance of the solution itself is that the system could be subverted, e.g., by malware. In that case, the system may intentionally mislabel data, e.g., to enable exfiltration of confidential data. Note that such mislabelling may happen in different ways, e.g., through an automatic process not requiring human input or by deceiving a user into believing that either the data object or the security label being attached is different from what is actually the case. The latter might be achieved by displaying a data object (e.g., document) different from the one being labelled, or if additional information is hidden within the data object (e.g., as non-visible metadata or through

steganographic techniques).

The use of high assurance systems resistant to subversion could mitigate this threat. However, there is often a requirement for users to be able to use commodity applications, or custom applications built on top of commodity operating systems, and no high assurance commodity operating system is likely to become available in the near future.

High confidence in security labels may in some scenarios be achieved by having a high assurance gateway in front of a given system or security domain, labelling information originating from that system/domain with a given label. For instance, all information originating from a Restricted domain may be labelled as Restricted without risk of leaking classified data. While such functionality is available in some guards (e.g., [4]), it provides limited flexibility. For instance, it does not enable unclassified data produced at a Restricted workstation to be exported to an unclassified domain.

Thus, if one are to use one of the previous approaches, one basically has to choose between low assurance in the correctness of security labels (but high flexibility) or very limited flexibility (but potentially high assurance). Policy-based labelling (PBL), as proposed in this paper, aims to strike a balance between these two extremes, by using policies to specify rules for how data objects can be labeled. As in Attribute Based Access Control, the policies are specified based on attributes, i.e., typically of the subject, the resource (i.e., the object to be labelled in our case), the action (i.e,. the specific label requested if any), and/or the environment. The proposed framework makes use of plug-in modules for evaluating the data to be labelled and for obtaining other attributes relevant to the policy decision. The attributes obtained by these plug-in modules are then used as input to a policy evaluation. This way, the policies determines how data objects are allowed to be labelled, while checks customized for specific scenarios or applications can be provided through plug-in modules.

The framework may be used to automatically label data objects, to enforce that the labeling performed by a subject (e.g., application/user) is according to policy, or some combination thereof. The framework may be implemented as a custom standalone device for higher assurance or be integrated on client machines (potentially using some type of separation mechanism for increased protection).

While the proposed framework may be deployed in different ways, some interesting possibilities arise if it also processes incoming data objects. The framework can then be set to effectively monitor the incoming data objects and their associated label attributes, and this information may then be used to help decide the label attributes of outgoing data objects. This, in addition to subject

and environmental attributes, means that the proposed framework can utilize both previously identified bases for determining the label attributes of a data object.

It should be noted that PBL, depending on the exact policy configuration, can be used to create highly secure as well as completely insecure labelling configurations. Its power as such lies in the fact that it enables the policy creator(s) to specify policies providing an acceptable risk for a given deployment, enabling finely calibrated security checks to be added through custom plug-in modules. Due to the use of plug-in modules, the framework is also extensible with regard to object formats (e.g., file types or message types), although our implementation effort so far has focused on XML/SOAP.

The rest of this paper is organized as follows: Section 2 introduces the proposed policy-based labelling framework. We then briefly discuss various deployment options in Section 3. Example use cases are given in Section 4, including a description of how monitoring of incoming data objects can be used to assist labelling. Section 5 describes our prototype implementation. Section 6 presents a more detailed discussion of the policy aspects of PBL, including an overview of the XACML policy language. Section 7 provides a survey of related work, before we conclude in Section 8.

2 The proposed labelling framework

The basic idea of the proposed framework is that various information about the data object to be labelled and the circumstances under which this labelling is performed can be represented as attributes and be used as input for a policy decision. Apart from information about the data object itself, the attributes may, for instance, convey information about the subject (e.g., user or application) for which the data object is being labeled and the environment, including what type of information has previously been received or originated by the environment (e.g., terminal, virtual machine, etc.) originating the data object.

A functional high level design of the proposed labelling framework is shown in Figure 2.1. We will first provide a brief overview, before a more detailed description of each functional component is provided. As can be seen, a data object for labelling is taken as input. This data object is first processed by one or more Attribute Modules, each returning a set of zero or more attributes. The set of attributes returned by the Attribute Modules are then provided to the Policy Decision Point (PDP) for a policy decision. In the normal case, the PDP returns its policy decision in the form of permit or deny and a set of

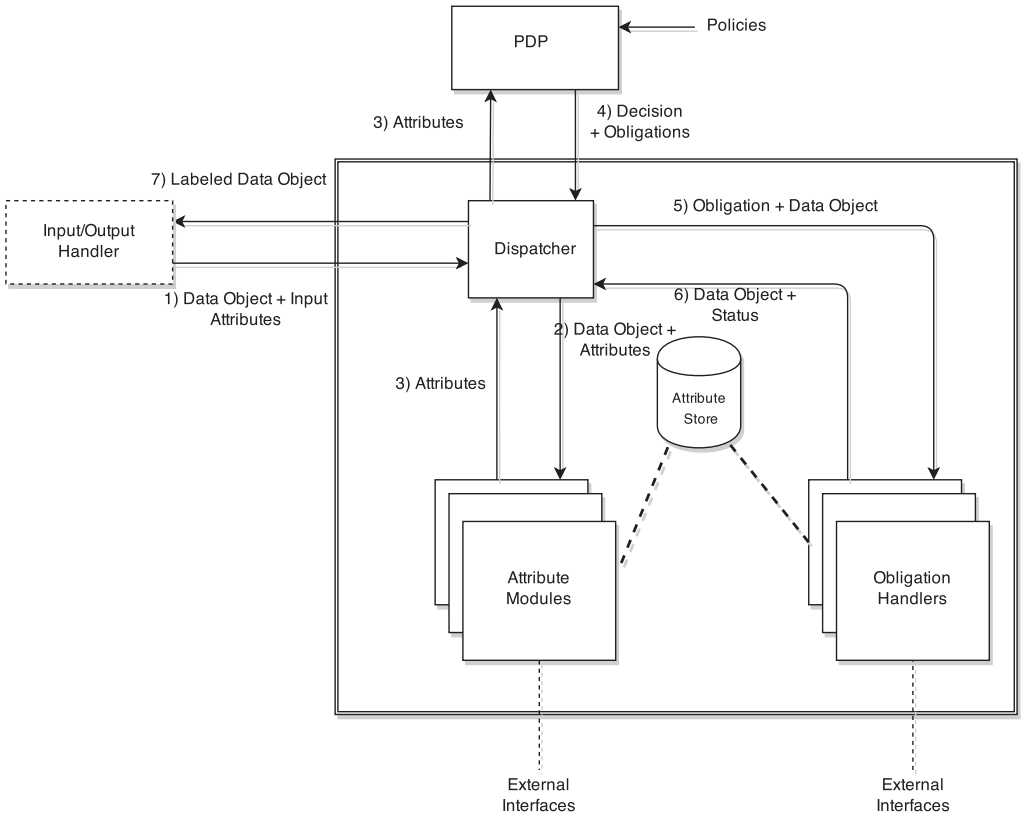

Figure 2.1: A functional view of the policy-based labeling framework. explanation of the role of each component.

obligations. A permit decision means that processing (i.e., labelling) of the given data object can proceed, while a deny decision means that further processing (i.e., the intended labelling) of the given data object is not allowed. In any case, any obligations returned by the PDP are to be handled by Obligation Handlers. The Attribute Modules and Obligation Handlers are plug-in modules that can be included with the framework, providing flexibility for different usage scenarios and applications, while the configured policies of the PDP provide the glue between them.

2.1 Input/Output Handler

The Input/Output Handler is an optional component that is not part of the core framework, but that may be used to provide adaptations for various deployments. In addition to the data object, the Input/Output Handler may optionally also supply some input attributes to the framework. Such attributes may for instance include information about on what interface the data object has been received or label attributes requested to be set by the subject. The latter may also be achieved by having a label included together with the data object upon submission, where the subject suggest/request the label attribute(s) to be set. Input attributes could also include the identity of the subject, e.g., if authentication has been performed at a lower layer.

2.2 Dispatcher

The dispatcher is responsible for receiving each data object and then calling the applicable Attribute Module(s) according to configuration. It then sends a policy decision request to the PDP, containing the input attributes and the set of attributes returned by the Attribute Module(s). After receiving the decision from the PDP, the Dispatcher is responsible for calling any applicable Obligation Handlers and releasing or discarding the final data object in the case of a permit or deny respectively. The Dispatcher is also responsible for ensuring that each Attribute Module is only allowed to write to its own attribute name space in order to enforce separation of privilege between Attribute Modules.

2.3 Attribute Modules

There can be one or more Attribute Modules,[5] each returning a set of zero or more attributes. Each Attribute Module has read access to the data object and the set of attributes returned by any previously called Attribute Modules. Which Attribute Modules are called and in which order are specified by the configuration.[6] Examples of Attribute Modules may include content checkers (e.g., dirty-word and XML schema validation), subject authentication modules, subject attribute retrievers (e.g., based on SAML), and security label parsers. Depending on their implementation, Attribute Modules may also have access to external information, e.g., to retrieve information from a directory service.

[5] There may in principle be special cases with zero Attribute Modules, in which case one would need to use Input Attributes alone for the policy decision.

[6] For performance optimizations, one might also specify modules that can be called in parallel independent of other modules, but this is not supported by our current implementation.

2.4 Policy Decision Point (PDP)

The PDP is responsible for making the policy decision based on the attributes in the submitted decision request and the applicable policies. At the high level, the PDP has a set of policies and takes a set of attributes as inputs, and outputs a decision (permit or deny) together with a set of zero or more obligations. Although the exact language in which the policies are expressed is ultimately an implementation choice, we have chosen to utilize the XACML 3.0 policy language for this purpose. Our use of XACML 3.0 policies is further discussed in Section 6.

2.5 Obligation Handlers

There may be one or more obligation handlers,[7] each responsible for handling one or more obligation types, where each obligation is identified by an obligation attribute. An attribute value associated with the attribute is used to convey any additional information needed to carry out the obligation when necessary. Examples of possible Obligation Handlers include handlers for security labelling, content filtering/removal, logging, alerts, error messages, and label binding. A special obligation handler can also be used to enable policy writers to specify that additional Attribute Modules are to be run under specific conditions, thereafter resubmitting the policy decision request to the PDP including the new attributes.

2.6 Attribute Storage

The attribute storage is an optional component used for persistent storage of attributes. The attribute storage can be written to (including deleted from) by one or more Obligation Handlers, and can be read by Attribute Modules. Thus, the policies can specify that certain attributes are to be written to persistent storage given certain conditions, thereby enabling a historical context for later policy evaluations (for other data objects). It should be noted that each Attribute Module and Obligation Handler may also implement or have access to separate storage, e.g., an Obligation Handler for security logging may have append only access to a separate storage device.

[7] There may in principle be special cases with zero Obligation handlers, e.g., corresponding to a case where labelling is performed external to our framework and where the framework is only used to block data objects with a non-compliant label without performing further action.

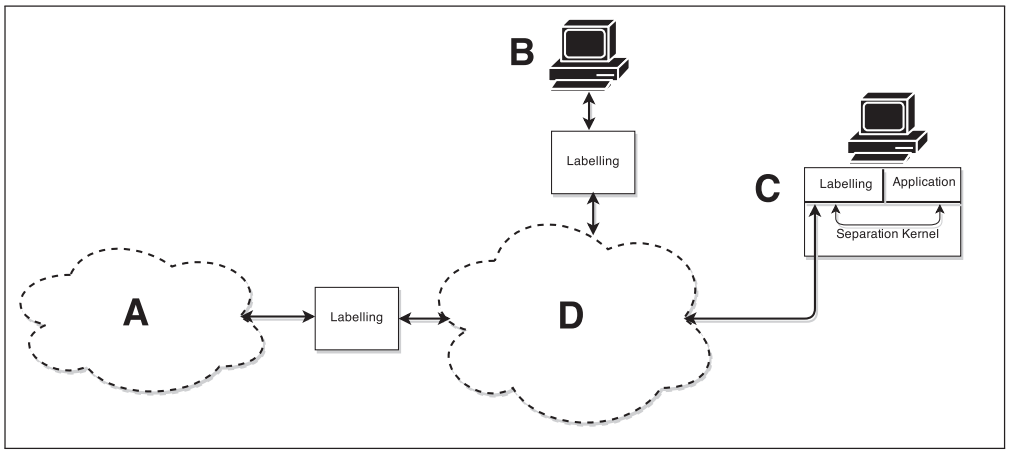

Figure 3.1: Illustration of different deployments of the policy-based labelling framework.

3 Deployment alternatives

The framework can be deployed in different ways. One alternative is to deploy it on the same machine as the application originating the data to be labeled (e.g., a user machine or an application server). There are multiple ways in which this can be performed, the most basic one being that the application program uses the labelling framework as a library. For SOAP Web services, another option is to deploy the framework as a handler being applied to all messages by the Web services platform.

Another natural deployment option is to deploy the labeling framework as a service that can be called over the network. Depending on the policy, this may require authentication of the subject requesting labelling to be performed.

The framework could also be deployed as a gateway or proxy (as the labelling gateways for network A and node B in Figure 3.1). An advantage of the gateway approach is that the framework can be made non-bypassable in both directions, which as we will see facilitates some interesting applications. A gateway could for instance be positioned in front of each system for which labelling is to be performed (e.g., server, sensor, or workstation), in front of a set of systems, or in front of a network domain. While such a labelling gateway may typically be implemented as a separate device, it could also be integrated with a guard or in a separate non-bypassable MILS partition on the client/server (as node C in Figure 3.1).

When deploying the framework on a user machine or similar (e.g., application

server) one may consider separating the labelling framework from the remainder of the system for better security protection, e.g., using a separation kernel. Assuming labels are cryptographically protected, smart cards, hardware security modules, or similar may be used for protecting the cryptographic keys.

4 Illustration Examples

The following will illustrate the functionality of the framework through a few simple examples.

4.1 Basic Labelling Examples

Let us first consider an example where we have a user who has been working on a data object (e.g., document) within a NATO Secret domain, and now wants to label this data object as NATO Restricted. The user or her application would then submit the data object to the labeling framework, depending on how the framework is deployed (as discussed in the previous section). The user would indicate that she wants to label the document as Unclassified, either by attaching an Unclassified label to the document or by supplying this information as input attributes to the labelling framework. Other input attributes supplied by the Input/Output Handler could typically include the subject identity (e.g., if authenticated by SSL/TLS over the network or by the operating system in the case of local invocation). Otherwise, means for subject authentication could also be included within the data object itself, e.g., by including a digital signature.

Upon receiving the data object and input attributes, the labelling framework first invokes the configured Attribute Modules. Each Attribute Module is provided as input the data object, the input attributes, and any attributes returned by previously called Attribute Modules. In our example case, there might for instance be one Attribute Module retrieving additional information about the subject (e.g., role, labelling privileges, etc.) from a directory service. Alternatively, this information could be supplied by a Security Assertion Markup Language (SAML) assertion embedded within the data object, in which case an Attribute Module would be responsible for validating the SAML assertion and retrieving its attributes. Another Attribute Module could determine the type of the data object (e.g., document format or message type). Depending on type, the data object could for instance also be subject to a dirty word check, returning an attribute indicating the severity or category of any dirty-words found.

All attributes (both input attributes and those obtained by the Attribute Modules) are then submitted to the PDP for a policy decision. In this example we will assume that we have a policy stating that the default action is to accept the user's proposed label given that the data object validates to a known allowed type, no dirty-words were found, the subject is authorized to perform such labelling, and the label is within a specified valid label range (e.g., NATO Unclassified, Restricted, or Secret, with a specified set of categories). Assuming these requirements are fulfilled, the policy evaluation will result in a "Permit" decision, i.e., meaning that the labeling process should proceed with executing the obligations specified (there could also be obligations specified for the case of "Deny"). In our example the obligations could include to attach the security label (unless this was already supplied by the subject), digitally sign the label and data object (i.e., confirming that the security label is attached according to policy), and to log the result for audit purposes. All these obligations would need to be performed by the respective obligation handlers before the labelled data object is returned.

It may be noted that custom Attribute Modules may be included for specific applications. For instance, an Attribute Module could be included for NATO Friendly Force Information (NFFI) messages.[8] Such an Attribute Module could for instance retrieve the the positional data from the NFFI message, so that this information may be used during policy evaluation. This way, one could for instance specify that only NFFI messages relating to a certain geographic area are to be marked as releasable to a given coalition partner.

4.2 Information Flow Assisted Labelling

If we deploy the labeling framework in such a way that all incoming and outgoing traffic (e.g, to the node, domain, or virtual machine) can be monitored, we can potentially maintain complete records of all the content that each environment receives and sends out. This is for instance the case in Figure 3.1, where the labelling component has been deployed such that all in- and outbound traffic to/from computer B, the application partition (e.g., virtual machine) within computer C, and the network partition A can be controlled. If we consider D the larger network, all information would typically be labeled before entering D, although there may also be policies for labeling unlabeled information origi-

[8]The NATO Friendly Force Information (NFFI) standard defines a message format and protocol for exchanging information on friendly forces during coalition operations [9]. A NFFI message is an XML file that contains at least the mandatory fields Positional data (longitude, latitude and altitude), Velocity, ID (name and text string), and Operational status.

nating from D. The information collected through such monitoring can later be utilized to determine the label attributes of new data objects, for instance:

1. "High Water Mark" - Through continuous monitoring, we can keep track of the highest classification level of the data that the environment originating the data object to be labelled has received. We can then use this value to enforce a "high water mark" policy, in which we restrict a node to only sending out data with the security classification set to the same (or higher) level as that of the most highly classified data object it has received. This may be particularly be useful when combined with what we refer to as instantiated environments, that can be (re)started from a known information state for each worksession. For instance, the separation kernel on computer C could be configured to allow the labelling framework to restart the application partition clearing its state.

2. Misusability Weight - A less rigid alternative to the high water mark policy is to take a risk based approach. Such an approach could be based on an assessment of the potential damage that could be caused if an environment/user leaks the information it possess. This could for instance be performed by computing a misuseability weight for each environment, similar to the Textual Misuseability score (TM-Score) proposed in [13]. The security labels of the information exchanged would be used to assist the calculation of the misuseability weight, and the misuseability weight would then be used as an input attribute to the PDP for labeling decisions.

3. Similarity Checking - If we index all data objects passing through the labelling mechanism, we can compute the similarities of incoming and outgoing data objects, using for example the cosine similarity measure [2]. This knowledge about similarity with other data objects, whose label attributes are known, can then be utilized as input to the policy decision. Such signature checking might be applicable to media data as well, e.g., videos, images, and audio files.

5 Implementation

We have implemented a prototype of the proposed framework in a mixture of Clojure and Java, utilizing a XACML 3.0 PDP for the policy decision point.

As the framework needs to be able to specify obligations on a per-rule basis, we rely on the latest version of XACML (3.0), which introduced rule-level

obligation expressions. Previous versions of XACML only supports obligations at the policy level. Although it is technically possible to achieve the same effect by using multiple policies instead of multiple rules, this would in quickly result in an impractical large number of polices. At the time of writing, there are several robust open-source Java implementations of XACML 3.0 available, e.g., Balana by WSO2 [16] and a framework made available by AT&T [1]. For our prototype implementation, we opted for using the Balana version. This decision was primarily based on the fact that the AT&T version had just been released when we started the project.

The Attribute Modules and Obligation Handlers are implemented as classes of the interfaces "AttributeModule" and "ObligationHandler" respectively. Thus, if a user wants to adapt the framework to a specific use case then she would only need to provide custom implementations of these two sets of classes and alter the configuration settings accordingly. With modularity, flexibility, and extendability in mind, we designed the system such that the set of Attribute Modules and obligation handlers that need to be executed are specified in a configuration file, which is read once on start-up and then re-read on any subsequent modifications.

So far, we have implemented several Attribute Modules to be able to test the basic functionality of the framework:

1. Dirty word - This module extracts all the text from a XML file, tokenizes the result and matches the tokenstream against a set of predefined categories of "dirty words." The attribute value is a list of "dirty word" categories.

2. Schema validation - Attempts to validate a provided XML file against a set of pre-defined schemas. The resulting attribute value is a list of matched schema names.

3. Maxium security label - Extracts all the security classifications present in a file that uses the "XML Confidentiality Label". The highest security classification encountered is returned as the attribute value.

4. User - Dummy module that returns hardcoded user data such as role, username, security-clearance, age, gender, and phone-number.

5. NFFI positional data - Parses an NFFI message and returns the positional data as the attribute value.

6. Signature verification - validates the signature of an XML/SOAP document.

We have also currently implemented some Obligation Handlers:

1. Email - Sends an email to the security administration team.
2. Change Label - Changes the security label.
3. Reevaluate - Initiate a policy reevaluation.
4. Attribute persistence - Write attributes to the attribute store.
5. Insert - Inserts an XML element (e.g., an XML Confidentiality label) at specified position in an XML document.

6 Policies, attributes, and obligations

Attributes are key-value pairs that serve as the main building blocks of ABAC systems, where it represents the properties of the entities involved in an authorization request. By utilizing a well-established and much studied access control paradigm in the policy-based labelling framework, we achieve the benefit of not having to re-engineer libraries and code. It also enables the security administrators to express the labeling logic in terms of familiar concepts, potentially using existing tools for policy management.

All rules are evaluated based on the subject, data object (i.e., resource), action (i.e., the label attributes requested by the subject), and environment attributes. In terms of the set of features that must be provided by a framework in order to support rules of this form we needed:

1. A component that extracts the attributes from the involved entities (i.e., Attribute Modules)
2. A policy evaluation engine that uses the attributes to express when a rule should be triggered (i.e., XACML PDP and policies)
3. Operation sets associated with each rule that are executed depending on the outcome of the rule (i.e., Obligation Handlers)

How these components are implemented in the framework, i.e., as attribute modules and obligation handlers, is outlined in Section 2, and in the following sections, we will provide a brief review of the XACML 3.0 standard before giving a more in-depth description of how policies, obligations and attributes are utilized in the framework.

[8] One can also provide custom attribute categories.

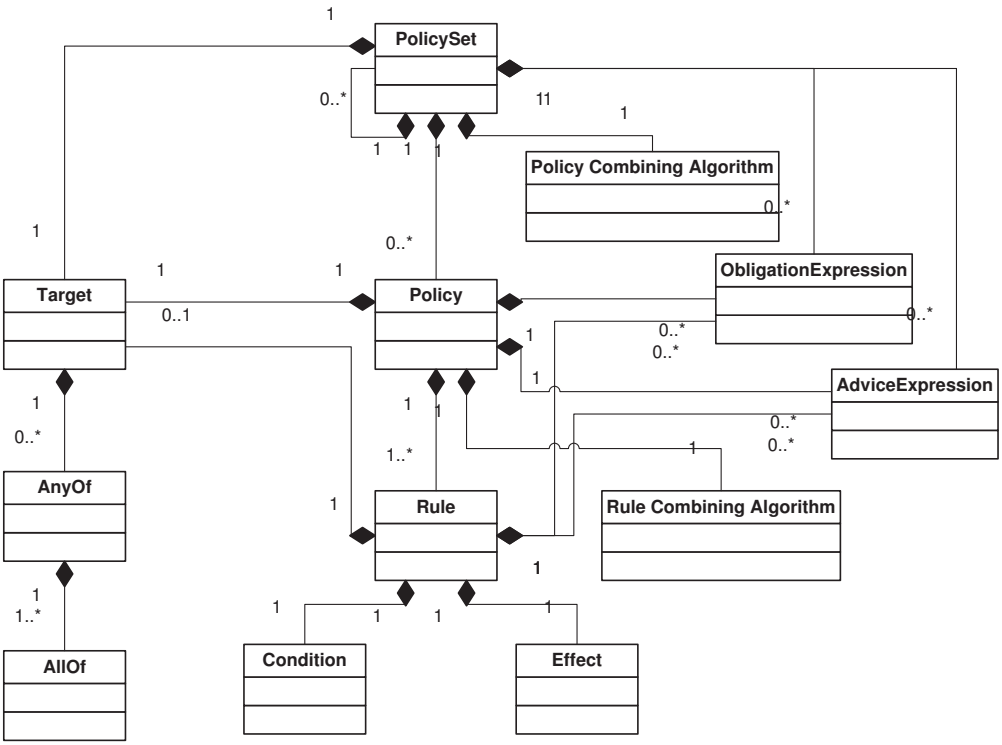

Figure 6.1: XACML 3.0 policy language model [10].

6.1 XACML

The eXtensible Access Control Markup Language (XACML) is a specification for defining access control policies using XML. As shown in Figure 6.1, the XACML rule constitutes the basic building block for defining policies in XACML. Each rule has an effect, either permit or deny. Furthermore, each rule may specify a target. The target of a rule defines the subjects, resources, actions, and/or environments to which the rule applies (i.e., who may, or may not, do what to which resource given the environment).

A rule may also contain a condition, further restricting the applicability of the rule. Such a condition may involve attributes of the subject, resource, action, and/or environment, and can make use of arithmetical, comparative, set, and Boolean operators. Thus, XACML provides high granularity for defining rules and allows rules to be made context sensitive. A condition may for instance involve the role of the subject, the time of day, and/or previous events.

A XACML rule is not to exist on its own, but instead as part of a XACML

policy. In case there is more than one rule in a policy, these are interrelated by a rule-combining algorithm. Three different rule combining algorithms are defined: deny-overrides, permit-overrides, and first-applicable. In addition, custom algorithms can be defined.

The target of a policy may be determined from the targets of its rules or specified explicitly. If no target is specified by a rule, the target of the rule is taken to be the same as the target of the containing policy. The target is used by the PDP to determine if the policy/rule is applicable to a given request. Consequently, the effective target of a rule is at least as strict as the target of the containing policy (i.e., PDPs only consider the rules within applicable policies).

A policy may also specify obligations. Such obligations may for instance be that an e-mail should be sent to the resource owner if access is granted or that denied requests for access should be logged. In order to ensure that the obligations are fulfilled, any obligations should be carried out before granting access. In the latest XACML specification (3.0) rules may also specify obligations.

Another newly introduced feature is the *Advice* statement, which is a type of obligation that may be ignored.

Policies may again be combined into a policy set in the same way that rules are combined into policies. The algorithms for this are equivalent to the ones for combining rules, with the addition of an only-one-applicable algorithm, where only one policy is to be applicable to a given request/ target.

A PDPs response to a request is either permit, deny, not applicable (i.e. if no policy/rule was applicable), or indeterminate (i.e. if an error occurred). One or more obligations (and/or advices) may also be specified, and access must be denied unless all the obligations can be fulfilled.

6.2 Usage of XACML Policies, Attributes, and Obligations

Policies are what make up the logical component of the framework and can be thought of as the connection between the Attribute Modules and the Obligation Handlers, in the sense that the attributes of a labeling request are effectively mapped to a set of obligations by the policies. This mapping is specified through the use of XACML policy rules as described in Section 6.1. Whereas in XACML, the rules are used mainly to derive a final Permit/Deny outcome, we are primarily interested in the effects of the accompanying obligations. Each rule/obligation pair describes an operation to perform for a request whose attributes satisfies the rule's condition statement. All obligations of applicable

rules whose effect matches the final evaluation outcome (permit/deny) are to be carried out.

Returning to the content checking example discussed earlier in Section 4, we can now express how to check for the presence of dirty words in terms of a XACML condition. Let us assume that we have an Attribute Module that extracts the textual contents of the document, tokenizes the resulting text, matches the token stream against a set of known "dirty words," and then maps these words to categories before finally taking as the attribute value this list of matched categories. Our XACML rule, shown in Figure 6.2, creates a bag of categories by using the *string-bag* function and checks if any of the "dirtyword-category" attribute values are members of this bag by applying the *string-at-least-one-member-of* function. If there is at least one match, the rule is applicable and evaluates to "Permit". Assuming there are no further policies or rules that causes the policy decision to evaluate to "Deny", the obligations associated with a "Permit" decision must be fulfilled. In this example, we want the presence of a "dirty word category" to imply that the appropriate security level is "NATO Secret," and as such we can represent this by an obligation (and obligation handler) that when executed changes the label of the document to "NATO Secret". We can achieve this effect through the use of a rule-level obligation in combination with either an "AttributeAssignment" or "AttributeAssignment-Expression" obligation element, which we simply use to pass arguments to the obligation handler class from the policy. In Figure 6.2 we show this by calling the obligation handler "SetSecurityLabel" with the attribute "securityLabel" set to "NATO Secret."

As XACML policies can become quite complex, a suitable tool should be used for policy management.

7 Related Work

The proposed solution represents an application of attribute based access control (ABAC) principles to the process of information labelling. While ABAC is concerned with deciding whether access should be granted or not (i.e., permit or deny), our objective is to decide what security label to apply to a given data object. In particular, our work makes use of the XACML 3.0 [10] ABAC policy language. While XACML also provides an architectural access control framework, the PDP is the only component that we actually utilize from this. Still, one could say that the Dispatcher and Obligation handlers, in our framework, loosely corresponds to the Policy Enforcement Point (PEP) in XACML

```
<Rule Effect="Permit" RuleId="dirtyword-categories">
    <Condition>
        <Apply
            FunctionId="string-at-least-one-member-of">
            <Apply FunctionId="string-bag">
                <AttributeValue
                    DataType="string">
                    DirtyWord-Category-1
                </AttributeValue>
                <AttributeValue
                    DataType="string">
                    DirtyWord-Category-2
                </AttributeValue>
                <AttributeValue
                    DataType="string">
                    DirtyWord-Category-3
                </AttributeValue>
            </Apply>
            <AttributeDesignator
                MustBePresent="true"
                AttributeId="dirtyword-category"
                Category="object"
                DataType="string"/>
        </Apply>
    </Condition>

    <ObligationExpressions>
        <ObligationExpression FulfillOn="Permit"
            ObligationId="SetSecurityLabel">
            <AttributeAssignment
                DataType="string"
                AttributeId="securityLabel"
                Category="object">
                NATO Secret
            </AttributeAssignment>
        </ObligationExpression>
    </ObligationExpressions>
</Rule>
```

Figure 6.2: An example XACML 3.0 policy that checks whether there are any classified acronyms in data object and if so sets the security label to "NATO Secret".

and that our Attribute Modules to some extent resembles a Policy Information Point (PIP) in XACML (i.e., providing attributes of the subject, object, environment and action). The information flow between the components do however also differ from that in XACML.

A variation of ABAC, Content-based Protection and Release (CPR) [15], has been proposed for future use in NATO. In CPR attributes within a content label are used to convey the properties of an information object. Access decisions are then based on protection and release policies effectively expressing requirements (in terms of attributes) on the user and her terminal and/or environment in order to be granted access to information objects with such properties. CPR depends on the ability to assign content properties to information objects, and the work proposed in this paper may as such be used with CPR to determine the content properties of data objects (either automatically or in cooperation with the user).

The CPR paper [15] also present the NATO Metadata Binding Service (NMBS). NMBS can be used to bind specified metadata (e.g., a security label) to an information object, using a binding mechanism of particular strength (e.g., digital signature). Similar services are also available as commercial products, e.g., [5]. Our proposal differs from these solutions by providing a framework for determining what label attributes are to, and should not, be included within a given data object's security label according to policy. The binding of the security label to the data object is in our framework to be performed within an Obligation Handler (or alternatively by the Input/Output Handler), and may in principle be realized by invoking a binding service if for some reason not to be implemented locally. By providing a service interface, our framework may also be deployed as a labelling service itself.

Part of the motivation for the work in this paper is that it is difficult to assure that a data object being labelled does not contain unintended information or has a wrong label attached. This is particularly the case on a commodity platform providing limited assurance. This problem is related to the the problem of knowing whether what you actually sign with a digital signature is what you see when you sign an electronic document, which is a well known problem (e.g., [14]). Labelling of data objects is nevertheless different from that of users electronically signing juridical contracts and approving financial transactions. Because data objects to be labelled may be generated at a relatively high rate, and potentially are large and in a format unsuitable for human review, it is not practical to have the user review and confirm the action on a separate device. For many labelling applications (e.g, sensor data), there may not even be a human user present.

This is also reflected by the fact that, apart from the internal labelling used for mandatory access control in some high assurance operating systems, existing approaches to higher assurance data labeling are typically based on the use of "labelling gateways". An example of this include the XML/SOAP guard presented in [4], which may be configured to label outgoing information with a specific label. While the framework proposed in this paper may also be deployed as a gateway (e.g., as part of a guard), it has the advantage of providing the means for determining the label attributes of different data objects (i.e., through the use of Attribute Modules combined with attribute based policy evaluation).

If there are less stringent requirements for assurance, e.g., for scenarios involving relatively low risk, labelling may also be performed by the user on a commodity system. Several applications for this are offered by Titus [12], e.g., enabling a user to label files on Microsoft Windows and providing plug-ins for use within Microsoft Office applications. A similar plug-in for Microsoft Outlook is also available from SMHS [11], utilizing a security label service from Isode [6].

A planned solution for automatic confidentiality classification of non-structured text is discussed in [8]. It focuses on the use of machine learning techniques to determine the confidentiality classification of unstructured data (e.g., documents). The results of this work could potentially be utilized for creating Attribute Modules within the framework presented in this paper. While [8] anticipate the use of Sun's XACML 2.0 implementation for policy-based classification, this is not further explored. The anticipated use of XACML in their solution would however apparently be quite different from our. There is for instance no mention of XACML obligations, which our use of XACML is completely dependent upon. As discussed, this is also the reason why XACML 2.0 is not suitable for use as part of our framework, as it lacks support for rule-level obligations. Furthermore, our proposal is fundamentally based on ABAC/XACML principles, utilizing Attribute Modules to obtain attributes to be used as input to the policy decision for performing the classification, and then Obligation Handlers for fulfilling the policy outcomes. The potential monitoring of incoming data objects to help determine the label attributes of a data object is another differentiating factor of our proposal that has the potential to significantly improve the decision basis.

8 Conclusion

In this paper, we have introduced the concept of policy-based labelling as an ABAC inspired approach to data labelling, and proposed a framework for for policy-based labelling of data objects. A prototype of the proposed framework has been implemented as a proof-of-concept and will be used for further experimentation. Furthermore, we have provided several examples of how the policy-based labelling framework can be utilized.

Fully automatic labelling of data objects is just one potential application of the proposed solution. The use of the framework to enforce that the label attributes suggested by the subject (e.g., user or application) is according to policy may be at least as important. A variation of the latter could also be to utilize the framework for aiding the subject as to what label attributes to apply. Alternatively, one might choose to generally accept the labels suggested by the subjects, but utilize the framework for selecting suspicious cases for additional audit. Due to its flexible plug-in architecture and rich policy language, the framework is also well suited for performing translation between different security policies and label formats.

While our current prototype implementation has not been implemented with assurance in mind, a main property of the proposed solution is that it enables the enforcement of the labelling policy to be separated from low assurance commodity user/application platforms. This enables the labelling mechanism to be implemented with higher assurance (e.g., on a separate platform with a high assurance operating system or using some type of isolation mechanism on the user/application platform). This way, it can be assured that data objects are in fact labelled according to the labelling policy. As part of a larger solution, the overall assurance that for instance classified data is not leaked will however also depend on the specific labeling policy and Attribute Modules, seen in combination with what is known about the potential input. While some information sources (e.g., sensors) produce data that is relatively easily assessable with regard to label attributes, less may be known in advance about other sources producing potentially unstructured and diverse content (e.g., written documents). The ability to monitor incoming data objects to the environment from which labelling is being requested is as such considered an important element for improving the decision basis in such more general scenarios.

Continuing, we plan to further investigate the use of the policy-based labelling framework for determining the labelling attributes of data objects, including the use of more advanced Attribute Modules. In particular we want to further investigate information flow assisted labelling and risk based approaches

to labelling as discussed in Section 4.

References

[1] AT&T. AT&T XACML 3.0 Implementation. https://github.com/att/XACML, 2014.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[3] A. Eggen, R. Haakseth, S. Oudkerk, and A. Thummel. XML confidentiality label syntax. FFI-rapport 2010/00961, 2010.

[4] R. Haakseth, N. A. Nordbotten, Ø. Jonsson, and B. Kristiansen. A high assurance guard for use in service-oriented architectures. International Conference on Military Communications and Information Systems, 2015.

[5] Infodas. SDoT labelling-service: XML security labels for cross-domain information exchange. URL http://www.infodas.de/download/SDoT_Labelling_Service_eng_130422_II.pdf.

[6] Isode. Isode security label server. URL http://www.isode.com/products/security-label-server.html.

[7] R. Kissel. Glossary of key information security terms. NIST IR 7298 revision 2, 2013.

[8] A. Magar. Automatic security labelling prototype system architecture. DRDC Ottawa CR 2011-134, 2011.

[9] R. Malewicz. NATO Friendly Force Information (NFFI)(version 1.2) Interface Protocol Definition IP3. *NC3A Working Document*, 2006.

[10] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0. OASIS Standard, 2013.

[11] SMHS Ltd. Braid. URL http://www.smhs.co.uk/products/braid.

[12] Titus. Titus classification. URL http://www.titus.com/software/desktop/index.php.

[13] A. Vartanian and A. Shabtai. TM-Score: A misuseability weight measure for textual content. 2014.

[14] A. Weber. See what you sign secure implementations of digital signature. In *Proc. Intelligence in Services and Networks: Technology for Ubiquitous Telecom Services*, pages 509–520, 1998.

[15] K. Wrona and S. Oudkerk. Content-based protection and release architecture for future nato networks. In *Proc. Military Communications Conference*, pages 206–213, 2013.

[16] WSO2. WSO2 Balana implementation. `https://github.com/wso2/balana`, 2014.

Paper II

Automatic security classification by machine learning for cross-domain information exchange

Automatic security classification by machine learning for cross-domain information exchange

Hugo Hammer, Kyrre Wahl Kongsgård, Aleksander Bai, Anis Yazidi, Nils Agne Nordbotten and Paal E. Engelstad

Abstract

Cross-domain information exchange is necessary to obtain information superiority in the military domain, and should be based on assigning appropriate security labels to the information objects. Most of the data found in a defense network is unlabeled, and usually new unlabeled information is produced every day. Humans find that doing the security labeling of such information is labor-intensive and time consuming. At the same time there is an information explosion observed where more and more unlabeled information is generated year by year. This calls for tools that can do advanced content inspection, and automatically determine the security label of an information object correspondingly. This paper presents a machine learning approach to this problem. To the best of our knowledge, machine learning has hardly been analyzed for this problem, and the analysis on topical classification presented here provides new knowledge and a basis for further work within this area. Presented results are promising and demonstrates that machine learning can become a useful tool to assist humans in determining the appropriate security label of an information object.

1 Introduction

Security labels are used by the military, government agencies, international organizations and private corporations to associate security attributes to a specific information object [16]. These labels are intended to convey for instance the sensitivity of the contents of the information object. Traditionally, the information objects were typically *paper documents* with printed security markings indicating the confidentiality classification of the documents. In a military setting, examples of such security markings include the labels "Unclassified", "Restricted", "Confidential", "Secret" and "Top Secret". The security label mandates how

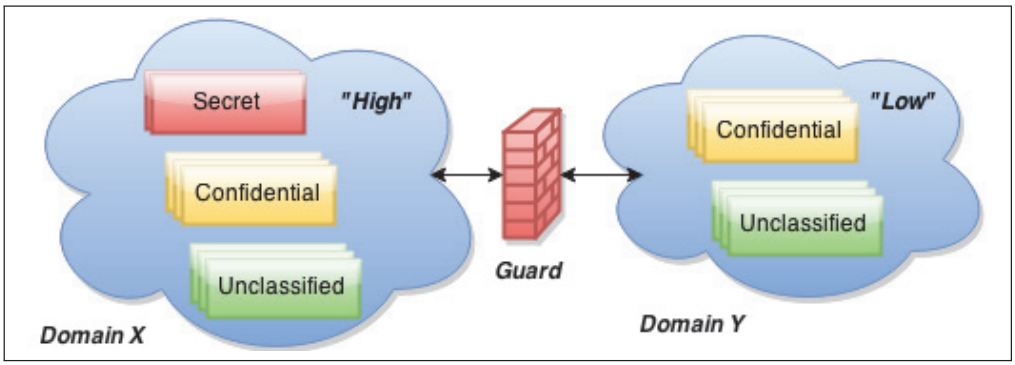

Figure 1.1: Example of cross-domain information exchange across a Guard between Domain X and Domain Y, where both domains contain various Unclassified (U) and Confidential (C) information objects. Only Domain X contains Secret (S) information objects. Here, the Guard should ensure that the Secret objects in Domain X are not leaked into Domain Y.

the information in the document shall be treated according to the governing security policy. For example, the "Unclassified" security label might mean that the information in the document does not need any particular protection, while the other markings might indicate that the information is classified and that the information in the document must be handled accordingly.

In modern environments the information objects are more likely to consist of digital information, e.g., Word documents, text messages and e-mails. If the purpose of the security labeling is concerned with preserving confidentiality, a security label such as the XML Confidentiality Label [9] can be used. The security label can be digitally attached and bound to the data object, e.g., by using a cryptographic mechanism such as a digital signature. This will also protect the integrity and authenticity of the security label (and its associated data object) during transportation and storage. However, the digital signature does not guarantee the correctness of the originally assigned security label.

Within a military setting, there are many information domains (e.g. networks, information systems, etc.) that have not implemented a mechanism to attach digital security labels to the digital information objects residing within the domain. The security attributes of the information objects will then typically be determined implicitly by the context, e.g., that "all digital information objects residing within Domain X shall be treated as Secret (S)" (Figure 1.1).

The first problem with this approach is that it can easily introduce inconsistent classification and massive over-classification of the information objects

within the domain, which is considered a practical challenge today [19]. For instance, Domain X in the example above might contain only a small number of information objects that require the security label Secret (Fig. 1). However, to preserve the confidentiality of the few Secret information objects, the remaining vast majority of information objects in the domain, which all deserves a lower-level security label, have to be considered as Secret as well.

The second problem of not having an explicit mechanism for assigning security labels is that it easily renders the information domain into an inflexible and isolated information silo. Even though Domain X in the example above contains mostly security information of low security classification, the information objects within the domain cannot easily be exchanged with systems that are not accommodating Secret information or be accessed by systems or personnel without a corresponding security clearance.

In the advent of the information age, these information silos are exactly what most organizations have been struggling to avoid and to move away from over the past 10-20 years. For instance, within this period of time the NATO allies have identified the need to move from the old paradigm of "need-to-know" to a new paradigm of "need-to-share" [7], [18], [22]. This means shifting away from single-domain information silos and move towards cross-domain information exchange (Fig. 1), without losing the ability to protect the information appropriately. Information sharing is considered a strategic capability and a steppingstone to obtain information superiority by ensuring that all allies have the newest and most pertinent information at hand at any time [6].

Two-way cross-domain information exchange is usually accommodated by a guard that is responsible for the information flow control between two domains (Figure 1.1). The guard inspects the security label of any information object that is requested moved from one domain to the other. The guard will only permit the information object to be passed to the other domain if it carries a security label that authorizes this action. The use of guards will be outlined in further detail in Section 2.

A critical point is to determine which security label to assign to an information object in the first place. A simple approach is to attach an explicit security labels based on the implicit security classification of the origin of the information object. In many cases this is acceptable. For instance, it may be known that a specific sensor only produces data of a specific type with a given classification. However, often the origin of the object will not give an accurate classification (e.g., if the origin is Domain X in the example above). Thus, in many other scenarios, one would prefer to assess the actual content of the data object instead, to determine the correct attributes of the security label.

The traditional way of assigning security labels is based entirely on human judgment, e.g., the security classification of a document is determined solely by the author of the document or by another evaluator performing a review. While security labeling is paramount to cross-domain information exchange and future information superiority in the military domain, undertaking the actual labeling is a tremendous challenge. The most optimistic advocate the introduction of fully automated tools. However, one may argue that tools are primarily needed to support, assist and partly offload the humans in their effort of assessing the security classification of an information object. Results presented in this paper are applicable to both approaches.

The work in this paper is applicable to both human-assisted (semi-automatic) and a fully automatic the security labeling process. The three main use-case scenarios include:

1. Proactive labeling (human-assisted): Security labeling is undertaken by a combination of human effort and automated tools. The tools can be used proactively, e.g., the software is assisting the human in determining the correct security label for the information object.

2. Reactive labeling (human intervention): In a setting where all the data has been manually assigned a security label, we still want to detect if data, when it leaves a secure domain, has been previously mislabeled either due to human error or malicious agents. These reactive label checkers can trigger actions depending on the policy, such as denial of access or a notification that calls for human intervention.

3. Fully-automated labeling: A fully automated solution would be desirable, and could be used in some settings. However, in many military settings and other security-dependent scenarios, it might not be realistic to expect it implemented in the near future.

While we acknowledge that a completely automated security labeling process might be unrealistic in near future, solutions for automatic labeling is a corner stone also for the scenarios above that includes human assistance or intervention.

Even though cross-domain information exchange could benefit from methods for automatic security labeling, little has been published on the topic. The contribution of this paper is putting this topic on the agenda, and exploring techniques for content analysis that is more sophisticated than the dirty word scanning techniques that are currently deployed. In this paper, we employ a machine learning approach to the problem, and to the best of our knowledge, this

paper is together with our previous works in [12], [10] and [11] the only published work addressing this issue. Other relevant available information we have found is a Master thesis from 2008 [8], as well as a military (thus, difficultly accessible) technical report [5]. All three works make comments on the surprising lack of published data. As noted in Section 2 on related work below, there is a need for specific methods tailor-made for the problem and commercially available products are not sufficient to this end.

In summary, methods and algorithms on the specific problem of security labeling need to be explored in the open literature, and their accuracy in determining the correct security classification of an information object should also be assessed openly. This paper attempts to bring the problem space and the technology development into the open literature domain.

2 Related Work

The following sections provides a brief overview of related work in the field.

2.1 Cross-Domain Information Exchange and Guards

Figure 1.1 shows a simple scenario requiring information exchange between different domains. As indicated in the figure, it relies on placing a data guard between the two domains. A number of commercial cross-domain information transfer solutions/guards, e.g., Lockheed Martin's Radiant Mercury (RM), BAE's DataSync Guard and Boeing's eXMeritus Hardware Wall, have been certified and officially approved for use by the Department of Defense (DoD) in the US [21]. These data guards facilities the secure information transfer between networks operating at different levels of classifications and with different security policies. Common for most of the commercially available guards is that they perform basic checking of the format and metadata of the information objects, such as checking that the information object is formatted according to required specifications or that it carries the required metadata/label. In terms of content scanning, the guards typically support some type of basic "dirty word" checking. As discussed in Section 1 there is a need for more advanced content scanning techniques. The security labels that are checked by the guard before it permits information to be released, can be first set by the guard itself (e.g. in a separate pre-processing software module), by a module on the clients, or by a labeling gateway (e.g., see [13]). If there are less stringent requirements for assurance, e.g., for scenarios involving relatively low risk, labeling may also be performed

by the user on a commodity system. Titus [3] offers several applications for this, e.g., enabling a user to label files on Microsoft Windows and providing plug-ins for use within Microsoft Office applications. SMHS [20] provides a similar plug-in for Microsoft Outlook, utilizing a security label service from Isode [15].

2.2 Frameworks for Security Labeling of Information Objects

Kongsgård et. al. [17] provide a security labeling framework for determining what security label attributes are to, and should not, be included within a given data objects security label according to policy. They present a solution for the use of attribute based access control (ABAC) principles to the process of information labeling. In particular, the framework provides support for pluggable attribute modules (e.g., content checkers) whose output serve as input to the policy decision. The work conducted herein is as such highly relevant to the work in [17], by providing a potential attribute module.

A variation of ABAC, Content-based Protection and Release (CPR) [23], has been proposed for future use in NATO. In CPR attributes within a content label are used to convey the properties of an information object. Access decisions are then based on protection and release policies effectively expressing requirements (in terms of attributes) on the user and her terminal and/or environment in order to be granted access to information objects with such properties. CPR depends on the ability to assign content properties to information objects, and the work proposed in this paper is therefore relevant. The CPR paper [23] also presents the NATO Metadata Binding Service (NMBS). NMBS can be used to bind specified metadata (e.g., a security label) to an information object, using a binding mechanism of particular strength (e.g., digital signature). Similar services are also available as commercial products, e.g., [14].

2.3 Content Scanning for Automatic Security Labeling

Since the content scanning of existing cross-domain information exchange solutions is typically limited to some form of "dirty-word checking", there is a need for exploring more advanced scanning techniques. A review of commercially available content tools is undertaken in [5]. The conclusion is that there might exist commercial content analysis tools that could be appropriate for the task, but these are proprietary. Without knowledge of their implementations or reports on their performance for this problem, they are of little use in an academic setting. Note that a product that is designed for one general task

(e.g. determining the topic of a document) might not perform well in a specific task (e.g. determining the security classification of a document). Furthermore, most research in document categorization focuses on identifying the topic of a document (topical classification), while security classification is non-topical, and usually a more challenging problem. Moreover, often the methods need to be tailor-made for the specific tasks (e.g. for the type of security attribute addressed) and optimized for a specific context (e.g. for the type of information object or for the exact topic of the information content). Thus, knowledge of specific methods is more important in the long run than availability of generic proprietary products or research results from related or more general problems. It was not before 2008 that it was suggested in a Master thesis to use general machine learning techniques to the problem [8]. The thesis did not make attempts to apply machine learning, but proposed an architecture for the problem. In the architecture, the document is first checked for compliance with policy and classified by topic, before the actual security classification is undertaken. General machine learning methods were first applied to the problem in a military technical report from 2010 [5]. The three general classifiers - Nearest Neighbor (NN), Naive Bayes (NB), and Support Vector Machine (SVM) are applied for security classification and compared. The corpus, which is collected from the Digital National Security Archive [2], is manually separated by topic in advance. Different pre-processing methods commonly used for text analysis, such as word stemming, term weighting and dimensionality reduction, are applied, and the effects of applying them are also analyzed. Among different contributions, this paper investigates the effects of topic classification prior to security classification, as only assumed in previous works [8], [5]. To the best of our knowledge, the works in [12], [10] and [11] were the first published papers on automatic security classification and the application of machine learning techniques to this problem.

3 Experiments, Results and Discussions

3.1 Lexical Corpus

The Digital National Security Archive contains the most comprehensive collection of declassified US government documents available to the public [2]. We base our analysis on documents from the same collection that were used in [12], [10] and [11]. This collection was originally chosen because it contains a mix of both classified and unclassified documents from three unrelated domains:

Table 3.1: Documents used in the experiments (after removal of short documents).

	Total docs	Unclas-sified.	Confi-dential	Sec-ret	Top Secret
AF	834	333	395	102	4
CH	948	322	247	286	93
PH	1023	424	514	85	0
Sum	2805	1079	1156	473	97

- AF, Afghanistan: The Making of U.S. Policy, 1973-1990
- CH, China and the United States: From Hostility to Engagement, 1960-1998
- PH, The Philippines: U.S. Policy during the Marcos Years, 1965-1986

Of the 5853 documents available within these three topics, we do not use duplicate documents and documents classes that are very small or have an unsuitable classification [12], [10]. Then we remove documents with 30 words or less (after having also removed some keywords as explained below), and we end up with 2805 documents in total (Table 3.1).

To simplify the analysis, we reduce the security classifications into two main classes used for the further analysis. The first class is *Unclassified,* which comprises 1079 documents. The second class is *Classified*, which is an aggregation of the document classes "Confidential", "Secret" and "Top Secret" in Table 3.1 [12], [10] [11]. However, then we remove some of the latter documents to get an approximately equal share between classes, ending up with around 2158 documents. This is only approximate, because we pick randomly documents from the aggregate "Classified" class with an expectancy of remaining with 1079 documents.

Further details about the corpus can be found in

The documents on the DNSA website are scanned pdf documents of poor quality, and the content text extracted from OCR (optical character recognition) has many errors. However, each document has also a meta-data in plain text format containing further information about the pdf documents. Many of the documents have an abstract (i.e. extract of the content of the document), i.e. the abstract is given in plain text format in the meta-data attached to the pdf documents. The abstracts are short texts that are assumed to contain high

quality information about the document content. Such meta-data was used in [5], but details are not specified, except that the limited number of abstracts used indicates that they selected a small subset of the abstracts for their analysis.

3.2 Data processing and machine learning

To go further in our analysis we extracted the raw textual contents using OCR techniques, using the OCR service provided by Abbyy [1].

For all the experiments we resorted to the simple bag-of-words model [4], in which any word order was discarded and a document was represented merely by a vector of term frequency-inverse document frequency (tf-idf) weights.

For our "base case" analysis we utilized spell checkers and auto-correction to mend many OCR errors and performed some analysis on this processed material. The processed/auto-corrected text material is more concise, and less verbose than the raw text (i.e. mis-spelled words are merged into the same word for the analysis). However, some important information might be lost, such as abbreviations that might be significant for the classification. Thus, we also undertook analysis on the raw uncorrected text material. (The latter analysis, which is not part of the "base case", is referred to as "raw" later.)

The raw material has the advantage of containing all information, while the quality of the information is poorer, e.g. in terms of many words with spelling errors etc. This trade-off indicates that in a real scenario where one does not have to rely on OCR, results would be generally better than presented in this paper.

We have not performed word stemming before machine learning. It turns out that this does not affect the performance considerably in neither positive nor negative way, but it reduces the size and sparsity of the vectors and gives easier computation.

Furthermore, as part of the pre-processing in our "base case" analysis we remove/ignore any keywords of the type "SECRET, "UNCLASSIFIED" etc. from all the textual contents prior to training the machine learning algorithm. If we were to leave these types of words in the text, it would potentially result in the classifiers yielding artificially good results, that effectively would only determine the security label based on absence or presence of such words.However, we also do some non-base-case analysis on material where these keywords are not ignored (referred to as the "keyword" analysis later) .

As for the actual machine learning we apply Support Vector Machine (SVM) for classification of classified vs unclassified documents. 70% of the documents are used for training set, while the remaining 30% constitutes the test set. As

Analysis case	All docs together	95% conf.int.	Split by country	95% conf.int.	Three clusters	95% conf.int.
Base case	0.78	(0.75, 0.81)	0.80	(0.76, 0.83)	0.78	(0.75, 0.81)
With keywords included	0.89	(0.87, 0.92)	0.88	(0.85, 0.91)	0.90	(0.88, 0.92)
Chronologically ordered	0.75	(0.71, 0.78)	0.66	(0.62, 0.70)	0.70	(0.67, 0.73)
Raw text (no auto-corr.)	0.87	(0.84, 0.89)	0.84	(0.80, 0.86)	0.85	(0.82, 0.88)
Short text abstracts	0.72	(0.66, 0.78)	0.72	(0.66, 0.78)	0.69	(0.61, 0.76)
Base case w/ 2 clusters	0.79	(0.76, 0.82)	0.79	(0.75, 0.82)		
Base case w/ 4 clusters	0.76	(0.73, 0.80)				
Base case w/ 8 clusters	0.77	(0.73, 0.80)				

Table 3.2: Main results of base case analysis and different variants of this

a starting point, we try to avoid any historic effect in terms of change within the topic over time. Thus, in our base-case analysis documents are not ordered chronologically, so both the training set and the test set contain documents that span the entire time period for the given topic.

3.3 Analysis of Base Case

As outlined above, the base case comprises full-text OCRed documents that are post-processed with various spell checking and auto-correction techniques. Classification-related keywords, such as "Secret" and "Unclassified", are removed from the text, and all documents are shuffled into an non-chronological order before the documents are split into a training set and a test set that is analysed with SVM. Results are summarized in the first line entry of Table 1.

A result of 78% classification accuracy (in the "All docs together"-column) indicates that the application of machine learning is a promising solution to the problem addressed in this paper. (The other columns will be discussed later in relation to clustering.)

3.4 The effect of keywords

In the second line entry we use the same documents as in the base case, but do not remove words like "Secret" and "Unclassified". We see that this has a clear effect on the results where the accuracy jumps from 78% to 87%. The second line entry indicates that our precautious concern about ignoring keywords (such as "Secret" and "Unclassified") to avoid unrealistically positive results, was well-founded.

3.5 Chronological Order

In a real setting the classification of new documents will be based on machine learning performed on historical documents. To test this we arranged the documents in chronological order, and performed training on the first part of the range. Then we tested on the second part. Note that this is an almost unrealistically difficult task. The training documents are used to classify documents that are created 10 to 15 years after the end of the training period.

We see that chronological aspect has a clear effect on the results with the accuracy dropping from 78% (base case) to 71%.

3.6 Analysis of Raw Text

The "Raw text (not auto-corr.)" row in Table 1 shows results for the base case machine learning undertaken on the raw output of the OCR, i.e. that is not subject to auto-correction. By performing auto-correction many words are corrected to the correct spelling (e.g. Chima is change to China) and one may expect that this is important. On the ohter hand, onemight also reduce the amount of data and eliminate important part of the information (e.g. such as abbreviations).

The results show that the accuracy increasing from 70% to 78% when auto-correction is performed. Auto-correction is very important for the classification performance of scanned documents. By visual inspection of the effect of the auto-correction, this is not a very surprising result.

In a real scenario where OCR is not needed, these results indicate that machine learning will probably perform even better than demonstrated in this paper, because there will not be such a large amount of spelling errors, and at the same time no information need to be dropped to fix them.

3.7 Analysis of Abstracts Only

To analyze machine learning on more sparse information, Table 1 summarizes also results for machine learning undertaken only on the abstract (meta-data) of the documents. The abstracts are high-quality information that is available in text format without requiring OCR (i.e. few spelling errors seen), assumed to be formulated "to the point" and containing a brief overview of the essential information of the documents.

The results indicate a little drop in performance. This may indicate that the amount of information is often important for the applicability of machine

learning. It might give an indication that the machine learning approach is well applicable in scenarios where large information objects (e.g. text documents) are exchanged, while less applicable to scenarios with smaller information objects (e.g. exchange of short emails or other text messages).

3.8 Clustering

The fourth column of Table 1 shows results where documents are split per country (manually split into 3 clusters), before the SVM is applied. In the sixth column we have used k-means clustering to split into three clusters automatically, disregarding the fact that documents are divided into three parts from the start (countries). The three lower entry lines shows results for other number of clusters, and also the use of two topical clusters within each of the three manual country clusters.

These results indicate that manual clustering (see the results for "split per country" in Table 1) is not necessarily better than automatic clustering (in the lower part of the table). We also observe there are great potential in clustering, as separate security classes might form separate clusters.

The sparse amount of previous work has assumed that manual topical clustering (here: per country) should be undertaken before the security classification without supporting these claims. Our results indicate, however, that this is not necessarily the case. This indicates that often the advantage of learning from more documents might outweigh the disadvantage of learning from other topics that are less relevant.

4 Concluding Remarks

This paper is aiming to put machine learning on the research agenda for cross-domain information exchange. Experiments confirm that machine learning approaches perform well and might become a valuable tool in this setting. Even though the performance can probably be increased further from an accuracy of 86% achieved in this paper, the remaining 14% inaccuracy indicates that without further research and improvements the method is applicable as an automatic tool targeted at assisting humans in determining the appropriate label or at detecting potential mislabeling of data objects.

The paper provides a number of learning-points compared to the sparse amount of previous work. In [5] is was assumed that manual topical clustering (here: per country) should be undertaken before the security classification, while

the work in [8] assumed automatic topical clustering. None of the works support these claims. Our results indicate, however, that this is not necessarily the case. This indicates that often the advantage of learning from more documents might outweigh the disadvantage of learning from other topics that are less relevant.

Furthermore, our results indicate that manual clustering is not necessarily better than automatic clustering. We also observe that there is a great potential in clustering, as separate security classes might form separate clusters. Exploring more along the line of more fine-grained topical classification is an interesting issue for further work.

We also argue that the chronological aspect should be taken into consideration in an analysis, since this comes natural with a practical use of automatic security classification. Our results confirms that the temporal order of the documents does make a difference, but the reduction in the performance was limited, indicating that machine learning still is a applicable and promising method. The development of a machine learning method that takes the temporal aspects into consideration is an issue for further investigation.

Finally, this paper has been limited to the use of classification of two security classes. Expanding the analysis presented in this paper to a scenario with more classes is a natural next step.

References

[1] Abbyy. "http://www.abbyy.com/". Accessed: 2015-03-26.

[2] Digitial nation security archive. Accessed: 2015-03-26.

[3] Titus classification. "http://www.titus.com/". Accessed: 2015-03-26.

[4] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[5] J. D. Brown and D. Charlebois. Security classification using automated learning (scale). Technical report, DRDC Ottawa CR, 2010.

[6] N. Brown. *Statement for the Record Before the 108th Congress Committee on Armed Services*. US House of Representatives, 2003.

[7] C. Cavas. *Petraeus: US must share more info with allies*. [Online], http://www.defensenews.com/story.php?i=4623591, 2015. URL http://www.defensenews.com/story.php?i=4623591.

[8] K. Clark. Automated security classification. Master's thesis, Vrije Universiteit, 2008.

[9] A. Eggen, R. Haakseth, S. Oudkerk, and A. Thummel. XML confidentiality label syntax. FFI-rapport 2010/00961, 2010.

[10] P. E. Engelstad, H. L. Hammer, A. Yazidi, and A. Bai. Advanced classification lists (dirty word lists) for automatic security classification. *Proceedings of The 7th IEEE International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC, 2015), Cyber Security and Privacy (CSP), Xian, China, Sept 17-19*, 2015.

[11] P. E. Engelstad, H. L. Hammer, A. Yazidi, and A. Bai. Analysis of time-dependencies in automatic security classification. *Proceedings of The 7th IEEE International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC, 2015), Cyber Security and Privacy (CSP), Xian, China, Sept 17-19*, 2015.

[12] P. E. Engelstad et al. Automatic security classification with lasso. *Proceedings of The 16th International Workshop on Information Security Applications (WISA 2015), Jeju Island, Korea, August 20-22*, 2015.

[13] R. Haakseth, N. A. Nordbotten, Ø. Jonsson, and B. Kristiansen. A high assurance guard for use in service-oriented architectures. International Conference on Military Communications and Information Systems, 2015.

[14] Infodas. SDoT labelling-service: XML security labels for cross-domain information exchange. URL `http://www.infodas.de/download/SDoT_Labelling_Service_eng_130422_II.pdf`.

[15] Isode. Isode security label server. URL `http://www.isode.com/products/security-label-server.html`.

[16] R. Kissel. *Glossary of Key Information Security Terms*. DIANE Publishing Company, 2011. ISBN 9781437980097. URL `http://books.google.no/books?id=k5H3NsBXIsMC`.

[17] K. W. Kongsgård, N. A. Nordbotten, and S. Fauskanger. Policy-based labelling: A flexible framework for trusted data labelling. International Conference on Military Communications and Information Systems, 2015.

[18] P.-P. Meiler and M. Schmeing. *Secure Service Oriented Architectures (SOA) supporting NEC, (Technical Report TR-IST-061) NATO*. 2009.

[19] U. S. G. P. Office. *Too Many Secrets:Overclassification as a Barrier to Critical Information Sharing*. Hearing Before the Committee on Government Reforms, US House of Representatives, 2004.

[20] SMHS-Ltd and Braid. *[Online]*. URL `http://www.smhs.co.uk/`.

[21] UCDMO. Cross domain wiki. `http://www.crossdomain.org`. Accessed: 2015-03-26.

[22] U. Wolf. Does NATO meet the challenge of the information era? In *23rd International Workshop on Global Security, Berlin, Germany*. 2006.

[23] K. Wrona and S. Oudkerk. *Content-based protection and release architecture for future NATO networks*. in Proc. Military Communications Conference (MILCOM), 2013.

Paper III

Automatic security classification with lasso

Automatic security classification with lasso

Paal E. Engelstad, Hugo Hammer, Kyrre Wahl Kongsgård, Anis Yazidi, Nils Agne Nordbotten and Aleksander Bai

Abstract

With an increasing amount of generated information, also within security domains, there is a growing need for tools that can assist with automatic security classification. The state-of-the art today is the use of simple classification lists ("dirty word lists") for reactive content checking. In the future, however, we expect there will be both proactive tools for security classification (assisting humans when creating the information object) and reactive tools (i.e. double-checking the content in a guard). This paper demonstrates the use of machine learning with Lasso (Least Absolute Shrinkage and Selection Operator) [7, 11] both to two-class (binary) and multi-class security classification. We also explore the ability of Lasso to create sparse solutions that are easy for humans to analyze and interpret, in contrast to many other machine learning techniques that do not possess an explanatory nature.

1 Introduction

Security classification is about classifying information objects, such as documents and text messages, into different groups, e.g. "Secret", "Confidential", "Unclassified" etc. The concept is not only used by the military, government agencies and international organizations, but also by private corporations, e.g. see [10]. The classification indicates the sensitivity of the contents of different information objects and mandates how the information object shall be treated according to the governing security policy.

The content of an information object is typically classified by human inspection and assessment, and given a security label (e.g. "Public" vs "Confidential" or "Business Internal"). However, with an increasing amount of generated information, there is a need for tools that can assist with automatic security classification, which is the problem this paper addresses.

Organizations that implement security classifications might also use *guards* to enforce information flow control according to the policy. The guard is typically located on the border between two domains, e.g. a "high" domain and a "low" domain. It is responsible to protect the confidentiality of the "high" domain, by denying objects of a too high classification to leave the "high" domain and be released into the "low" domain.

There is a number of commercial guard solutions available, e.g., Lockheed Martin's Radiant Mercury (RM), BAE's DataSync Guard and Boeing's eXMeritus Hardware Wall, have been certified and officially approved for use by the Department of Defense (DoD) in the US [12]. In terms of content scanning, the guards typically support some type of basic "dirty word" checking, i.e. the classification lists in these guards are not very advanced. A review of such content scanning tools is undertaken in [4].

In the future, we expect there will be both proactive tools for security classification (assisting humans when creating the information object) and reactive tools (i.e. double-checking the content in the guard). However, the only tool that resembles automatic security classification today, is the content checking that takes place in the guard itself, which is based on "Dirty Word Lists".

However, in this paper we propose to use machine learning as a starting point instead of "Dirty-word lists". In doing so, we put focus on Lasso (Least Absolute Shrinkage and Selection Operator) [7, 11], which has not been considered for this problem in previous work.

Lasso has a number of compelling features that are explored in this paper. For instance, it is easily expanded to multinomial regression.

However, perhaps the most attractive feature is the sparseness of the solution that Lasso provides, which makes it able to derive interpretable classifications lists as a outcome of the machine learning. A classification list, such as a "Dirty Word List", is with current state-of-the-art typically configured manually, because good human control with the list is critical. The disadvantage of manually constructed lists is that it is hard to construct lists with advanced functionality (e.g. introducing a weight factor to each word or an overall bias-term) which would force humans to take complex decisions (e.g. which values to assign). This paper demonstrates the use of machine learning to create more advanced classification lists automatically. A major obstacle for machine learning to be used is that they would create long lists that are difficult to inspect, analyze and control by humans. First, some of the most efficient machine learning techniques, such as Support Vector Machines (SVM), k-Nearest Neighbor (kNN) or Naïve Bayes (NB) [6] are hard to interpret by humans. SVM, for instance, is known to have notoriously poor interpretability, as noted by Kotsiantis [8].

The main contributions of this paper are as follows:

- Putting *Automatic Security Classification* as a topic on the research agenda. As pointed out in the next section, surprisingly little has been published on Automatic Security Classification.
- Constructing a well-described two-class benchmark experiment for a binary security classification problem, and exploring features of this experiment.
- Exploring the Lasso machine learner [11] as a method that has not yet been analyzed in literature for this problem.
- Exploring the multinomial regression of Lasso for situation with more than two security classes. This problem has not yet been analyzed in the literature.
- Taking false positives and negatives into account in the evaluation, which has not yet been done in the literature for this problem.
- Showing how machine learning in general, and Lasso in particular, can automatically create classification lists that are more advanced than the simple "dirty word lists" that are used today and that are still easy for humans to understand and assess.

2 Background

2.1 Introduction to Lasso

The main focus of the paper is exploring the use of Lasso as a novel approach to the automatic security classification problem. In terms of classification accuracy, we will compare Lasso with other state-of-the art machine learners, such as SVM.

For the sake of clarity, we should first briefly clarify how Lasso can be adopted for classification as it is mainly designed for solving regression problems. Lasso deals with feature selection by shrinkage methods which allow a variable to be partly included in the regression model.

We want to learn a text classifier, $y = f(x)$, from a set of n training examples $D = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)\}$. For text categorization, x_{ij} represent an entry in the Document-Term Matrix (DTM). That is, each element in the vector $x_i = [x_{i1}, ..., x_{ij}, ..., x_{id}]^T$ contains the word frequency of term j in document i,

and d is the total number of terms. Usually d is a huge number. The values $y_i \in \{-1, +1\}$ are class labels that indicate non-membership or membership to a class. The concept can easily be generalized to a multi-class problem. Due to space limitations, here we will focus on explaining how Lasso can be applied to solve a two-class/binary classification problem.

Logistic regression is a conditional probability model of the form:

$$p(y_i = +1|\beta, x_i) = \frac{1}{1 + exp(-\beta^T x_i))} \tag{2.1}$$

To estimate the unknown parameters in the Lasso model, we compute the following minimization

$$\widehat{\beta} = \arg\min_{\beta}\{-\sum_{i=1}^{n} \ln(1 + \exp(-\beta^T x_i)) + \lambda \sum_{j=1}^{d} |\beta_j|\} \tag{2.2}$$

where $\widehat{\beta}$ is the parameter estimates.

Lasso is a regularized regression method based on the L1-norm (second sum in equation 2.2). As we see, the expression consists of two sums. The first sum is for maximizing the likelihood estimation of the parameters $\beta_j, j = 1, 2, \ldots, d$ and the second sum is for controlling the sparsity of the solution. λ is a regularization parameter controlling the degree of sparsity.

2.2 Related Work on Automatic Security Classification

Little has been published on Automatic Security Classification, and all works we have seen make comments on the surprising lack of published data.

To the best of our knowledge, the only *published* work mentioning this issue dates to 2005. However, here Rhetorical Structure Theory is applied to the problem, which is not pertinent to the our work presented here [9]. In this paper, on the contrary, we employ a machine learning approach to the problem.

Other relevant available information - although not in the form of *published papers* - that we have found is a Master thesis from 2008 [5] and a (difficultly accessible) technical report [4] from 2010. The Master thesis did not make attempts to apply machine learning, but proposed an architecture for the problem. However, we use the technical report as a starting point for the work presented in this paper. The corpus used in this report was retrieved from the Digital National Security Archive - DNSA [2]. We will look closer at the corpus and their work in the following Section 2.3.

2.3 Experiments on Policy Documents from DNSA

The Digital National Security Archive (DNSA) contains the most comprehensive collection of historic and declassified US government documents available to the public [2]. These were chosen because they contain a mix of both classified and unclassified documents from three unrelated domains:

- AF, Afghanistan: The Making of U.S. Policy, 1973-1990
- CH, China and the United States: From Hostility to Engagement, 1960-1998
- PH, The Philippines: U.S. Policy during the Marcos Years, 1965-1986

The technical report from 2010 [4] used only 686 of the 5853 documents available DNSA documents within these three topics, but did not describe why so few documents were selected and what were the selection criteria. A number of other issues and parameters were also unclear from the report. We contacted the authors for further information, but due to the 5 years that have passed since the report was written, they informed us that most detail information about the experiment was now lost. Thus, a starting point of our introduction of automatic security classification into the public domain of published works was to re-conduct the experiments - however with less strict selection criteria - and thus with more documents in the corpus. Furthermore, we document in detail how the experiment is done. The experiment is presented in Section 3.1. Before we get to this, we will summarize the main pertinent information about their experiment as follows:

In the experiment of the technical report the "Unclassified" documents comprised one class, while all documents of higher classifications were aggregated into one class, thus reducing the security classification problem into one that only deals with two classes, i.e. into a *binary* classification problem. Being restricted to only the use of two different security classes is strictly limiting the applicability of the results presented in the report. To address this limitation, we will present a novel multi-class solution to the security classification problem later in this paper, in Section 4.

In these kinds of experiments, document texts are treated as bag of words, where punctuation marks, white spaces etc. are removed, and each document gives a term-vector with a dimensionality corresponding to the number of available words, and with each vector component providing the frequency of the word

being used, i.e. it is typically a sparse vector. Vectors of all documents are comprising the Document-Term Matrix (DTM), where each row in the matrix is a word-frequency vector corresponding to one specific document.

In the technical report, three general classifiers, namely "k Nearest Neighbor" (kNN), "Naïve Bayes" (NB), and "Support Vector Machine" (SVM) [6] were applied for security classification and compared. They concluded that SVM performed best, with a best performance around 85% classification accuracy. Other classifiers, such as Lasso, was not mentioned.

The report explored stemming of all the words in the DTM, i.e. every word in the documents are replaced with its word stem [4]. This means that sets of multiple columns in the DTM (corresponding to different word forms with the same stem) are aggregated into a single column corresponding the stem. This reduction of dimensionality of the DTM will increase calculation speed. In terms of classification accuracy, stemming might contribute positively by removing some words that introduce "noise", but might on the other hand also contribute negatively if the aggregation of words into the word stems leads to less precision in terms of conserving the exact meaning of each word form. The analysis in [4], however, concluded that word stemming does not influence classification accuracy considerably.

They also explored the use of "term frequency-inverse document frequency" (tfidt) [4] for term weighting (e.g. transforming the word frequencies metric in the original DTM into another metric), which is also a common technique in such experiments. Also on this point, they concluded that such term weighting does not affect the performance significantly.

3 Two-Class Benchmark Experiment

3.1 Experiment with Standard Machine Learners

The starting point of our analysis is to conduct a benchmark experiment. We base our experiment on the documents in the DNSA database from the same three topics as outlined in Section 2.3. Of the 5867 documents available within these three topics, we skip documents that are not very useful to the scenario we are targeting on to the analysis itself (Table 3.1).

For instance, 9 documents are removed since they are duplicates of other documents already in the set, while 620 documents are not useful for classification, since their classification is unknown (i.e. they are classified/marked as "UNKNOWN"). 1612 documents are removed because these documents are

Table 3.1: Documents used in the experiments

	Total docs	Dupl.	UN-KNOWN	EX-CISED	LIM.-OFF.	PUBL. USE	REST-RICT.	NON-CLASS.	UN-CLASS.	CONFI-DENT.	SEC-RET	TOP-SECR.
AF	2010	1	359	449	300	0	0	37	331	420	109	4
CH	1989		227	706	80	3	1	196	131	253	299	93
PH	1868	8	35	457	308	0	0	220	223	528	89	0
Sum	5867	9	621	1612	688	3	1	453	685	1201	497	97

marked as "EXCISED" (i.e. the classified text sections are removed). The "PUBLICUSE" documents could be considered as an unclassified class and the "RESTRICTED" document as classified, but since their sizes are so limited (3 docs and 1 doc, respectively), they are omitted. Furthermore, 688 documents are within the borderline classification "LIMITEDOFFICIAL", and therefore not considered. Thus we end up with 2933 documents in total, where 1138 documents (453 "NONCLASSIFIED" and 685 "UNCLASSIFIED") are aggregated to the "Unclassified" class of our analysis. The remaining 1795 documents (classified as either "CONFIDENTIAL", "SECRET" or "TOPSECRET") are aggregated into the other "Classified" class of the experiment. The number of documents per classes and per country (AF:Afghanistan, CH:China and PH:Philippines) is summarized in Table 3.1.

As part of our analysis algorithm, we also remove very short documents with only 30 word stems or less extracted. As a baseline, we remove 128 documents and end up with a corpus of 2805 documents constituting the 2805 rows of the DTM that is used for further analysis. As for the actual machine learning 70% of the documents (i.e. 1964 documents) are used for the training set, while the remaining 30% (i.e. 841 documents) constitute the test set. However, if we remove some additional keywords (as explained below in Section 3.2), a few more documents get to the limit of 30 word stems. Then we end up with 2793 documents (i.e. a DTM of 2793 rows), with 1955 documents in the training set, and 838 documents in the test set.

As DNSA contains pdf documents, we extracted the raw textual contents using the optical character recognition (OCR) service provided by Abbyy [1]. Since the scanned pdf documents are of poor quality, we used auto-correction to mend the many OCR errors occurring in the extracted text. The processed/auto-corrected text material is more concise, and less verbose than the raw text (i.e. mis-spelled words are merged into the same word for the analysis).

For all the experiments we used word stemming and resorted to the simple bag-of-words model [3], in which any word order was discarded, ending up with

Table 3.2: Classification performance for different machine learners.

Machine learner	Keywords ignored	Classification Accuracy	95% conf.int.
SVM	No	0.84	(0.81, 0.87)
5 kNN	No	0.70	(0.66, 0.73)
Naive Bayes	No	0.65	(0.61, 0.69)
SVM	Yes	0.77	(0.74, 0.81)
5 kNN	Yes	0.66	(0.62, 0.69)
Naive Bayes	Yes	0.65	(0.61, 0.68)
Lasso	No	0.90	(0.88, 0.92)
Lasso	Yes	0.78	(0.75, 0.82)

a corpus of 23477 word stems. Infrequent word stems, occurring less than 15 times in the entire corpus were also omitted to speed up calculations, by reducing the number of word stems (or columns in the DTM) from 23477 to 5840. Our analyses showed that this *term-frequency limitation* did not affect classification accuracy noticeably. Finally, each document (row in the DTM) was represented in the DTM merely by a vector of term frequency-inverse document frequency (tf-idf) weights [4].

Results are shown in Table 3.2. The upper pane of the table (i.e. upper three rows of results) confirms the conclusion in [4], i.e that SVM displays an accuracy around 84% (compared with around 85% in [4]). Furthermore, our results confirm that SVM has better performance than k-Nearest Neighbor (kNN) and Naïve Bayes (NB), even though the performance difference is higher here than in [4]. Nevertheless, based on these results, SVM will be used as a benchmark machine learner throughout the rest of this paper, while we will focus less on kNN and NB in the following. (We also note that Lasso performs better than SVM, as discussed in more detail in Section 3.3 below).

3.2 Removing Favourable Keywords

In addition, we did analysis where we removed/ignored any keywords relating the classification labels/words ("SECRET", "UNCLASSIFIED" etc.) from all the textual contents prior to training the machine learning algorithm, i.e. we removed the word stems of such words.

The reason we did this is that we suspected that the policy documents might include words that describe explicitly if a topic is classified or secret, such as the label itself. If we were to leave these types of words in the text, it would potentially result in the classifiers yielding artificially good results, that effectively would only determine the security label based on absence or presence of such words.

The number of word stems is thus reduced from 23477 to 23473, and after applying the minimum term-frequency limitation from 5840 to 5836 word stems (cf. Section 3.1). Removing keywords also reduces the number of documents in the corpus from 2805 to 2793, as pointed out earlier in Section 3.1.

By removing these keywords, we think that our analysis results are more amenable to many other types of information objects where this is not the case. Not surprisingly, the classification performance is a little lower when reducing these keywords, as shown in the second pane of Table 3.2 (i.e. in rows 4-6), even though very few words are removed. Indeed, we observe that the classification accuracy of SVM drops from 84% down to 77%.

3.3 Experiment with Lasso

In addition to testing the standard machine learners explored in a previous technical report, we also explored the use of Lasso for the automatic security classification problem according to the model presented in Section 2.1.

Table 3.3: Confusion matrix of the two-class (binary) classification

		Predicted classes		*Row-sums*
		Classified	Unclassified	
Actual classes	Classified	460	59	519
	Unclassified	123	196	319
	Column sums	255	583	838

The confusion matrix derived from the analysis is shown in Table 3.3, showing how the 838 test documents are classified ("Predicted class") compared to their actual classification ("Actual class"). Correct classification is along the diagonal, i.e. the *classification accuracy* is of $(460 + 196)/838 = 0.78$. Knowing the entire confusion matrix, other performance metrics can also be calculated, e.g. the *precision* is of $460/(460 + 123) = 0.79$, while the *recall* is of

$460/(460 + 56) = 0.89$. We could go further into exploring ROC and AUC curves, but due to space limitations this is outside scope of this paper and left for future work. This paper focuses primarily on the classification accuracy.

For comparison with other methods, the Lasso results are summarized in bottom pane of Table 3.2. Here, we observe that Lasso has a classification accuracy of 90%, and performs better compared to SVM at 84%. However, in all analyses in the rest of the papers, the keywords related to the security classifications or security labels are ignored (cf. Section 3.2). Then, Lasso yields a classification performance of 78%, compared to SVM at 77%.

3.4 False negatives

With a spam filter, which protects the confidentiality of a site from the outside world, one is often concerned with the problem of false positives. This is a legitimate message mistakenly marked as spam, which is critical if users miss critical information. A false negative, on the other hand, is usually not critical but mostly inconvenient, in terms of a spam message that reaches the inbox.

For guard, on the contrary, which protects against information leakage, false negatives must often be avoided at all costs. For instance, if a guard mistakenly releases a "Classified" document, due to the fact that it is mistakenly classified as "Unclassified", this false negative might have severe consequences for the organization. On the other hand, if the guard mistakenly blocks an "Unclassified" document from being released, this false positive is inconvenient - although probably not critical.

Analysis of false negatives of automatic security classification has not yet been tackled in previous work. From the confusion matrix, the false positives are given by the lower triangular part of the matrix, and is of $123/838 = 0.15$, while the false negatives are given by the upper triangular part of the matrix, and is of $59/838 = 0.07$. Thus, if we are only concerned with information leakage, and find a solution to tackle the inconvenience of false positives, the guard performs at 93%.

4 Multi-Class Security Classification with Lasso

While previous work has only considered two classes, here we consider automatic security classification with multiple classes. Just as we used Lasso for the binomial logistic regression of the binary (two-class) classification problem above, we can easily use Lasso for multinomial logistic regression of the multi-class

problem of automatic security classification. While we aggregated the classes "CONFIDENTIAL", "SECRET" and "TOPSECRET" into one common "Classified" class above, now we keep these as separate classes. Results of the Lasso multinomial regression are shown in Table 4.1.

Table 4.1: Confusion matrix of multi-class classification

		Predicted classes				*Row-sums*
		Top Secret	Secret	Confidential	Unclassified	
Actual classes	Top Secret	14	13	4	3	34
	Secret	2	55	52	30	139
	Confidential	0	16	268	62	346
	Unclassified	0	14	71	234	319
	Column sums	16	98	395	329	838

Now the classification accuracy is of $(+14 + 55 + 268 + 234)/838 = 0.68$. The fact that it drops compared to the binomial analysis is not surprising, as we now have considerably fewer documents available of each class. However, if we are only concerned with information leakage, and find a solution to tackle the inconvenience of false positives, the guard performs at 80%, derived from the classification accuracy on the diagonal (contributing 68%) and the false positives in the lower triangular part of the matrix (contributing 12%). Again, the confusion matrix allows us to calculate other metrics as well, but we leave an exploration of this to future work.

5 Lasso Feature Selection for Short Classification Lists

Lasso is a regularized regression method based on the L1-norm (second sum in equation 2.2 in Section 2.1.) L1 regularization is a compelling feature for creating brief classification lists, because it leads to sparse solutions where an additional automatic feature selection is performed within the Lasso learning algorithm. Lasso maximizes the likelihood, while constraining (i.e. penalizing) the sum of the absolute values (i.e. L1-norm) of the regression coefficients. Due to the L1-based regularization, we end up in a situation where some of the β_i-estimates becomes exactly zero and are removed from the solution. This leads to sparse solutions with many zero beta-values, effectively reducing the dimension of the solution.

Table 5.1: The real classification list lengths generated by Lasso, when adjusting the λ parameter.

MinIndex	54	50	45	40	35	30	25	20	15	10	5
$log(\lambda)$	-4.60	-4.41	-4.18	-3.95	-3.72	-3.48	-3.25	-3.02	-2.79	-2.55	-2.32
ListLength	**479**	404	306	206	139	93	55	23	15	7	3
Class.Acc.	0.78	0.78	0.78	0.77	0.77	0.75	0.73	0.7	0.68	0.66	0.62
Low Conf.Int.	0.75	0.75	0.75	0.74	0.74	0.71	0.69	0.67	0.65	0.63	0.59
High Conf.Int.	0.81	0.81	0.81	0.80	0.8	0.77	0.76	0.73	0.72	0.69	0.66

Indeed, while other machine learners easily would create solutions with thousands of words corresponding to the thousands of columns in the DTM, the optimal solution of Lasso to our experiment is a list of only 479 words, or 479 non-zero β_i values. Table 5.1 shows that the optimal solution of 479 words correspond to a MinIndex of 54, where the MinIndex indicates the size of the λ parameter. (A value of 54 corresponds to $\lambda = 0.01005$, i.e. $log(\lambda) = -4.60$.)

The size of the penalty is controlled by the λ parameter; the larger the penalty applied (or the lower the MinIndex is in Table 5.1), the sparser is the solution. In Table 5.1, we explore doing feature selection in Lasso, by selecting a non-optimal λ parameter that increases the penalty, i.e. Lasso creates sparser solution at the expense of a reduced (non-opitmal) classification accuracy.

Table 5.1 shows that Lasso demonstrates good preservation of the classification accuracy when reducing dimensions by increasing the L1 norm penalty. As observed in the table, Lasso preserves its accuracy of around 75% down to a classification list length of only 93 words, and the accuracy drops to only around 70% with a list length of 23 words. The ability of Lasso of preserving classification accuracy even when going to quite sparse solutions is also illustrated in Figure 5.1. In other words, Lasso is well adapted to creating advanced classification lists that are easily understood and inspected by humans, and the length of the classification list can be adjusted down at the expense of a moderately lower classification accuracy.

6 Conclusions and Future Work

This paper introduces Lasso into the solution space of automatic security classification. First, we observed an indication that Lasso might perform better than SVM and other commonly used machine learners in terms of classification accuracy (e.g. as shown in Table 3.2).

The paper confirms that with Lasso a classification accuracy around 80-

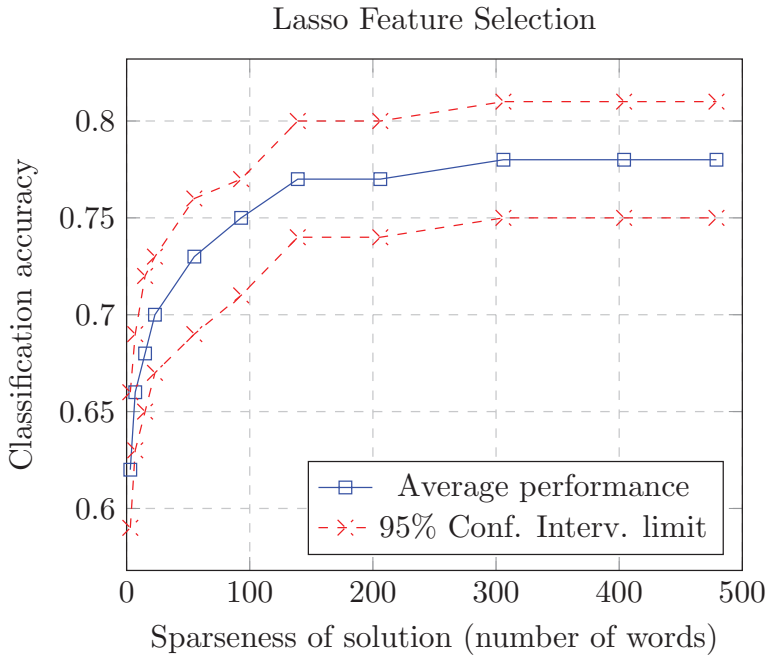

Figure 5.1: The blue curve shows the classification accuracy (y-axis) as a function of number of words (with non-zero β-value) in the solution (x-axis)

90% is realistic using machine learning for automatic security classification. However, we identified that some strict removal of some keywords (not identified in previous work) might make our analysis more amenable to other types of corpus. The requirement reduced classification accuracy to around 78%, and we used this as a starting point for our general analysis.

In addition to the performance benefits of Lasso for this problem, Lasso is easy to extend to multi-class classification problems where more than two different security classes are present. This has not yet been studied in the literature, and the paper demonstrated that multi-class security classification is feasible with Lasso. We also shed light on the false negative problem of automatic security classification and guard functionality, which is opposite to the false positive problem that have been studied for spam filters. Both the topic of multi-class security classification and the false negative problem are issues for future work.

Due to the fact that Lasso tend to create sparse solution, we demonstrated that Lasso is able to automatically create advanced classification lists that might partly replace the simple "Dirty Word Lists" used in guards today. Creating

short lists is an advantage since brief lists are easily to interpret by humans.

While other machine learners would easily create a classification list of thousands of words, Lasso automatically reduced the size to only 479 words (Table 5.1). Furthermore, we showed that by adjusting the λ parameter of the L1 norm, we could create even considerably shorter lists with good preservation of the classification accuracy. Comparing this technique with other feature selection techniques is an issue for future work.

References

[1] Abbyy. "http://www.abbyy.com/". Accessed: 2015-03-26.

[2] Digitial national security archive. "http://nsarchive.chadwyck.com/home.do". Accessed: 2015-03-26.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] J. D. Brown and D. Charlebois. Security classification using automated learning (scale). Technical report, DRDC Ottawa CR, 2010.

[5] K. Clark. Automated security classification. Master's thesis, Vrije Universiteit, 2008.

[6] C. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli. Comparison of classification methods based on the type of attributes and sample size. *Journal of Convergence Information Technology*, 4(3):94–102, 2009.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/.

[8] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[9] H. Mathkour, A. Touir, and W. Al-Sanie. Automatic information classifier using rhetorical structure theory, in intelligent information processing and web mining, advances in soft computing. 31, 2005.

[10] W. Nicolls. Implementing company classification policy with the s/mime security label. RFC 3114, IETF, May 2002.

[11] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[12] UCDMO. Ucdmo cross domain baseline list. see: `http://www.crossdomain.org`, 2011. Accessed: 2015-03-26.

Paper IV

Data Loss Prevention Based on Text Classification in Controlled Environments

Data Loss Prevention Based on Text Classification in Controlled Environments

Kyrre Wahl Kongsgård, Nils Agne Nordbotten, Federico Mancini and Paal E. Engelstad

Abstract

Loss of sensitive data is a common problem with potentially severe consequences. By categorizing documents according to their sensitivity, security controls can be performed based on this classification. However, errors in the classification process may effectively result in information leakage. While automated classification techniques can be used to mitigate this risk, little work has been done to evaluate the effectiveness of such techniques when sensitive content has been transformed (e.g., a document can be summarized, rewritten, or have paragraphs copy-pasted into a new one). To better handle these more difficult data leaks, this paper proposes the use of controlled environments to detect misclassification. By monitoring the incoming information flow, the documents imported into a controlled environment can be used to better determine the sensitivity of the document(s) created within the same environment. Our evaluation results show that this approach, using techniques from machine learning and information retrieval, provides improved detection of incorrectly classified documents that have been subject to more complex data transformations.

1 Introduction

Organizations and companies are handling increasing amounts of digital sensitive information, such as personal (e.g., health) data, military classified documents, trade secrets, and so forth. It is critical that such sensitive data is not leaked to unauthorized parties.

Many vendors offer data loss prevention (DLP) solutions that automatically monitor the storage, network, and users to detect and prevent such leakages [19, 25]. Among the mechanisms employed by these solutions, data classification, where data objects are labelled according to their sensitivity, is recognized as

an important enabler to improve their effectiveness [27]. Furthermore, given correct classification, data can be protected using appropriate access control and other security controls.

For instance, in a military or governmental context, a correct security label forms the basis for conducting information flow control using a so called guard (e.g., [12]), that ensures that only data allowed by policy (e.g., non-sensitive data) is released to external parties. If a Confidential domain and an Unclassified domain are connected, the guard will typically be configured to only allow data marked with a security label of Unclassified to pass from the Confidential domain to the Unclassified domain. While there may be additional security classifications (e.g., Restricted and Secret), the main concern to prevent data loss in such a scenario is to be able to detect when a document is incorrectly claimed to be releasable (i.e., Unclassified in this case).

Since security decisions are based on the classification specified by a data object's security label, it is crucial that data objects are classified correctly. However, human users or applications may mislabel data by mistake. Furthermore, an insider, or malware, could intentionally mislabel data to bypass security controls. For this reason, it is important to be able to validate the classification specified by users/applications, in order to detect mistakes and exfiltration attempts.

In this paper we explore the use of automated methods, based on machine learning (ML) and information retrieval (IR), to detect misclassification that could result in data loss. Text classification for DLP purposes has usually been based on techniques like fingerprinting of documents, keyword matching, and regular expressions, but more recently methods like ML and IR have been recognized as important for automated classification of unstructured documents [27]. However, little research is to be found on the subject [2, 7, 13, 14, 18]. A common trait of these previous works is that the classifier or index is based on a set of documents with well-known classification, either intended to include all the sensitive documents to be protected or a subset of documents used as a training set. It may be noted that it is necessary to assume a set of correctly classified documents which can be used as the base to estimate the correctness of new classifications. However, the noise in such a large generic set of index documents may result in misclassification, especially if new documents have been generated using content from sensitive sources, but where this content has been re-phrased, summarized or modified in other ways. Apart from recent work based on sequence alignment techniques [26], intended solely for detecting inadvertent data leaks, there is little previous research on detection of modified/transformed data leaks, the most relevant being the use of synonym

substitution (spinning) [2].

In order to improve the classification accuracy also in these situations, we propose a controlled environment where all imported documents are dynamically monitored. The collected information constitutes the basis for classification of new documents created within the environment itself. The intuition is that in order to generate a new document, various sources are usually accessed and consulted, and the resulting work will share some common traits with such references. Also, if one wants to leak out a sensitive document, it would have to be imported first, and therefore be among the indexed ones. Compared to the usual approach where one classifier based on all sensitive documents available in a company (or a generic subset thereof) is used to classify any outgoing document, our results show that a classifier built dynamically for each controlled environment provides improved accuracy especially for the more difficult content transformations. While this is a surprising result given that more data generally provides higher accuracy, it illustrates that determining sensitive content is much more context dependent than for instance topic categorization. Also, we assume that any sensitive content can be traced back to the imported documents. Furthermore we study how different algorithms from both ML and IR performs in different controlled environments by varying the amount of imported documents, i.e., the noise in the training set/index, and by generating modified outputs that share a variable amount of information with the imported documents, both in quality and quantity. While we find that the classification accuracy of ML and IR algorithms is only moderately affected by document spinning, we find that there are other types of modifications (e.g., summarisation and mixing of content) that have a more severe effect on classification accuracy. To our knowledge the effect of these more difficult modifications has not been previously studied in the context of ML and IR for DLP. Our results show that the use of controlled environments improve detection of these less obvious data leaks.

For our tests we use documents from the U.S. Digital National Security Archive (DNSA) [1] and reports from our own institution for validation. While previous work has performed experiments using *tens* or *hundreds* of leaked classified documents (WikiLeaks, DynCorp and TM[9]) and public documents from sources such as Wikipedia, the Enron email dataset and various online PC magazines [2, 14], and a smaller subset of the DNSA documents [7], we believe the work presented here constitutes the first study of transformed data leak detection based on such a large number of actual classified documents (i.e.,

[9]Transcendental Meditation

thousands).

The rest of this paper is organized as follows. Section 2 describes the proposed solution, including the controlled environment, classification approaches, and deployment options. Section 3 describes the methodology used to evaluate the proposed solution, including: the corpora used as datasets and the methods used to simulate the generation of new documents. Section 4 presents the evaluation results; Section 5 provides a presentation of related work; and Section 6 provides some final conclusions and summarises the main contributions of the paper.

2 Proposed Solution

We introduce a *controlled environment* as an environment where we have control on all imported documents and their classification. The set of imported known documents is defined as *input*, and any new generated document(s) is defined as *output*. We assume that the classification (i.e., sensitivity) of the input documents are known, e.g., through existing cryptographically signed security labels.

A controlled environment could for instance be a single computer, a virtual machine, a network/security domain, or an isolated process/application. Our proposed solution inspects the information flow to the controlled environment as shown in Figure 2.1b, and estimates the classification of output documents based on the information about the input documents. This is contrary to the traditional approach where a larger generic index/training set is used to classify any output document, as illustrated in Figure 2.1.

Our guiding hypothesis is that by limiting the set of input documents to those relevant to an output document, thereby reducing the noise in the classification process, we can more accurately estimate the classification of output documents.

While there are different ways to implement controlled environments, our hypotheses suggests that the more tightly enclosed environments such as a virtual machine or isolated process have potential for better accuracy than the more loosely enclosed environments such as a network domain. As will be seen in Section 3, this is supported by our evaluation results. To fully take advantage of this, a controlled environment may be *instantiated* for a specific task (e.g., the writing of a single output document), thereby clearing all state (i.e., *input*) between tasks/instantiations. We refer to this as an instantiated controlled environment.

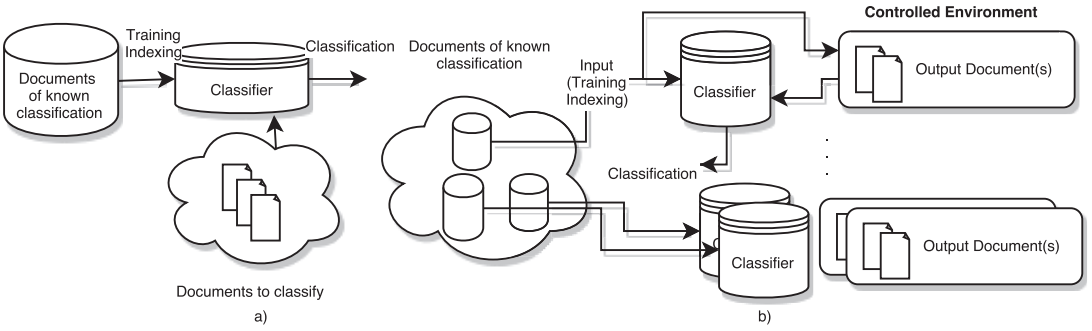

Figure 2.1: **a**) Usually, a classifier used in DLP is trained on all available documents **b**) With a controlled environment, only the documents of known classification accessed from the environment are used to train the classifier, which in turn is used to classify documents generated within the environment. Multiple controlled environments can exist simultaneously, each characterized by its own input and output.

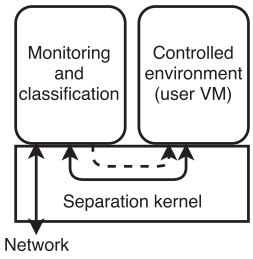

Figure 2.2: Illustration of how the proposed solution can be deployed on a separation kernel. The solid arrows represent the allowed communication channels while the dotted arrow indicates a control channel to restart the partition of the controlled environment (i.e., removing all input).

In the following two subsections we present first how the classification of output documents is performed and then discuss the different deployment alternatives.

2.1 Classification methods

To construct the classifier we use two approaches: one based on machine learning algorithms and one relying on information retrieval techniques (see Section 3.3.1 for additional details). In the case of machine learning, the input documents are used as the training set for the classifier, which is then used to determine the classification of the output document(s) directly. In the information retrieval approach, we build an index of all the input documents and their associated classification, so that we can query it to obtain a list of the input documents matching the output one(s), ranked by a similarity score. This list is then used to determine the more likely classification of the output.

2.2 Deployment

A controlled environment may in principle be any computing environment where the incoming information flow can be monitored. A monitoring mechanism similar to a guard, inspecting data at the application level, can be applied to control the information flow to a network/security domain, a virtual machine, a computer, a process, or some isolated container such as a sandbox. Cloud computing, with its extensive use of virtual machines, may also facilitate controlled environments.

Figure 2.2 shows a possible deployment of the proposed solution using a separation kernel. A separation kernel is a minimalistic type 1 hypervisor, running directly on the computer hardware, providing high assurance in the isolation between partitions (virtual machines) and controlled communication channels. As can be seen, monitoring of input and classification of output is performed within a separate partition, while the controlled user/application environment resides within another partition. The separation kernel ensures that all communication (e.g., documents from a file server) to/from the controlled environment has to pass through the monitoring and classification solution. The separation kernel can also be configured to allow the monitoring and classification partition to restart the controlled environment partition in order to clear its input state.

2.3 Applicable scope

Unlike the work in [7, 13], we do not aim to construct a classifier that is able to universally distinguish between what the government or some other organization considers to be classified and unclassified information; instead, we want to discover if sensitive information from a particular set of documents has been included (possibly in modified form) into a new document. Thus, the proposed method is not intended to be able to detect misclassification of output documents where the user has included sensitive content accessed through other channels (e.g., a second computer system or a paper document). Although this could potentially be addressed by combining our approach with those discussed in [7, 13], this is not pursued here.

2.3.1 Evasion and Poisoning

There are multiple ways to influence the machine learning classifier. By carefully choosing what to import, the training set can potentially be shaped in such a way that the algorithm later misclassifies a document containing sensitive content that the user wants to exfiltrate. This is an example of what is referred to in the literature as a *Causative* attack [4] in which the training set is intentionally poisoned. The Reject On Negative Impact (RONI) defense technique addresses this by modifying the learning process to dynamically discount those data points in the training set that have a significant negative impact on the performance [4]. Usage of procedures from the field of Robust Statistics has also shown to partially remedy the poisoning issue [4]. Also, performing a causative attack to exfiltrate larger amounts of data would likely result in detectable anomalies in the import patterns.

The IR approaches are not as susceptible to the *Causative* class of attacks because: 1) there is no training phase involved and 2) the presence of a classified document with a similarity score greater than the threshold value (see Section 3.3.1) will result in assigning the strictest label, e.g., "Classified", to the document.

Both the ML and IR methods remain vulnerable to the fact that if a highly skilled attacker has knowledge of the training set and hypothesis space, then he can shape the contents of the output document such as to stealthily avoid detection. This is known as an *Exploratory Attack*, in which the attacker does not influence the training phase [4]. This is a generic challenge in data loss detection, where this work advance compared to most previous work only considering unmodified data leaks.

The possibility of a skilled attacker evading the detection mechanism underlines why detection of malicious insider data leakage is considered a challenging research problem (e.g., [26]), and why said detection methods need to be deployed in combination with other countermeasures. As a controlled environment maintains control of the documents imported into the environment, we plan as further work to investigate how this information can be used to determine the risk an environment poses with regard to information leakage and to identify potential insiders. Also, it should be kept in mind that much of the data leakage by internal actors is due to accident (i.e., half of the serious incidents according to a recent study [15]).

3 Evaluation Methodology

In order to evaluate the effectiveness of the proposed solution, we needed to perform experiments using documents of known classification. As information from a sensitive document may leak by being included in a new document, we wanted to evaluate how modifications and introduction of external noise in output documents affect performance. We therefore introduced several methods to create new output documents based on manipulation of input documents and inclusion of external noise. This is further described in Section 3.1.1.

As the aforementioned approach does not accurately reflect how a document would be generated by a real user, we also use a second approach to validate our results. Using this approach we did not generate new output documents based on input documents (and noise), as in the first approach. Instead we used another set of documents with known classification as output documents, and used each of these document's reference list as an approximation of the input documents accessed during its creation. This latter approach is further described in Section 3.2.1.

Section 3.3.1 briefly describes the machine learning and information retrieval algorithms used as classifiers in the evaluation.

3.1 Evaluation approach 1

Here we present the first dataset and describe each of the document transformations that are used to generate the output documents.

3.1.1 DNSA dataset

The U.S. Digital National Security Archive contains the most comprehensive collection of declassified U.S. government documents available to the public [1]. From this repository we extracted three subcollections:

- *(Af)ghanistan: The Making of U.S. Policy, 1973-1990*;
- *(Ch)ina and the United States: From Hostility to Engagement 1960-1998* and
- *The (Ph)ilippines: U.S. Policy during the Marcos Years, 1965-1986.*

These were chosen because they contained a mix of both classified and unclassified documents from unrelated domains and from partially overlapping time periods.

Out of the 5853 retrieved documents, 1612 were dismissed because they had gone through a declassification process. The "PublicUse" (3 docs) and "Restricted" (1 doc) documents were removed due to their limited numbers. All documents marked either as "Unknown" (620) or "LimitedOfficial" (685) were removed. Documents that consisted of less than 30 words were also excluded. Post-filtering, the final data set consisted of 2884 documents. These documents were then partitoned into two categories depending on their original label. The 1117 documents marked "Unclassified" or "Non-Classified" were assigned the label *Unclassified*. While the remaining 1767 documents marked either "Confidential" or "Secret" were assigned the label *Classified*. In the final dataset, 525 out of the 883 documents belonging to the AF subset, 661 out of the 959 CH documents, and 610 out of the 1042 PH documents were labeled as *Classified*.

The final data set was divided into the *Input* (used as input documents) and the *Noise* datasets. The *Noise* dataset was used to introduce external noise into output documents.

For each PDF file we utilized the 3rd party OCR service Abbyy[10] to extract the textual contents. Any non-English words, i.e., artifacts of the OCR process, were then filtered out as part of a pre-processing phase as were any variations of tokens of the type *Secret*, *Unclassified* etc., as their inclusion would potentially result in the classifiers yielding artificially good results, that effectively would only determine the security classification based on the absence or presence of such words.

[10] http://www.abbyy.com/

3.1.2 Output generation

We create new output documents the following ways:

- **Existing Documents**: In this case an existing document from the DNSA dataset is used unmodified.
- **Mixed Documents**: In this case we randomly sample sentences from two documents of different classification, with each document contributing 50% of the output document. We take one document from the *Input* set and the other from the *Noise* set. The resulting document is then assigned the highest security classification of the two documents used in the mixing, the rationale being that the introduction of a single classified sentence or keyword into a unclassified document would imply that the correct security classification of the resulting document would be classified.
- **Rewritten Documents**: A document can be rewritten while still being semantically identical to the original, and should thus retain it's security label. We implement this by fitting a n-gram language model [17] for each document, and then use these probability distributions to generate new documents that contain semantically similar content. The correct security classification of the output document is assumed to be the same as that of the original document.
- **Abstract** For each document in the corpus there is an accompanying short abstract (not included as input) which we take as a condensed version of the document. The correct security label of the condensed document is assumed to be the same as that of the full document.
- **Spinner** An article spinner is a tool used in search engine optimization to generate documents for content farms and to circumvent plagiarism detection algorithms in general. It works by rewriting an article by replacing words, phrases and paragraphs with synonyms. We used the online commercial tool "Spin Rewriter 6.0"[11]. As this is proprietary software the exact algorithm used to "spin" documents is not known.
- **Translation** Using the online service Google Translate[12] we introduce noise into the document by performing a sequence of translation steps. In our experiments we utilize the translations steps:
  English → Simplified Chinese → English.

[11] https://www.spinrewriter.com/
[12] https://cloud.google.com/translate/docs

The "Exisiting", "Translation" and "Synonym" ("Spinner") transformation methods have also been used in the CLEF 2014 plagarism dection challenge [24]. It should be noted that "Abstract" is the most realistic transformation as it uses real abstracts created by human editors.

3.2 Evaluation approach 2

Here we present the second dataset used and then briefly describe how input documents were obtained based on the reference list of each output document in the dataset. As mentioned earlier, the intent behind introducing this second set of data and accompanying experiments is to further ensure that the results of the first approach is not attributed to the artificial way we generate output documents.

3.2.1 Private dataset

This dataset was based on an internal repository of technical reports at the Norwegian Defence Research Establishment (FFI). We collected a document set of 10 classified and 10 unclassified documents, to be used as output documents. The reference lists of these documents consisted of a mixture of both classified and unclassified reports, notes, and conference papers. The number of references per document ranged from 5 to 17, with a total set of 166 references (used as input documents).

For each document and its references we extracted the textual contents and then pre-processed the resulting text using the same procedure as for the first dataset, but with additional word filters customized to remove location and domain specific tokens that identifies the security label. For each of the documents we also removed the list of references from the text.

3.2.2 Input generation

As mentioned, this approach approximates the set of input documents by using the documents in the reference list of the output document instead. In this case the correct security classification of the output document is assumed to be the one stated on the original (output) document. Likewise, the security classification of an input document is also the one originally indicated on the document itself.

3.3 Algorithms

In this section we introduce the information retrieval and machine learning algorithms utilized in the experiments. For each algorithm, we performed 5-fold cross-validation and a grid-search to find the optimal configuration of tunable parameters.

3.3.1 Information Retrieval

In the field of information retrieval, a collection of indexed documents can be searched by computing the similarity between the documents and a query string. In the context of controlled environments, the similarity for each output/input pair can be used to detect if parts of the output document originates from the input document.

Each output document comes attached with the user's assigned label, and it is only interesting to run the detection routine for instances where this label may result in the document being released/leaked, i.e., *Unclassified* in our dataset. The following three-step algorithm computes the predicted security label:

- Compute the similarity between the output document and each of the input documents.
- Take the label of the input documents whose score is the highest as a *tentative* label.
- If the tentative label is *Unclassified*; we test whether the best *Classified* match has a higher score than some threshold θ. If this is the case: the final label is taken to be *Classified*. Otherwise we proceed with the *tentative* label.

The implementations provided by open-source search engine ElasticSearch [11] and the Python package gensim [22] were used for the experiments. As there are a number of different retrieval and scoring methods, we perform initial experiments with algorithms belonging to each of the four families:

- **TF-IDF/VSM** In the tf-idf vector-space model (VSM), the cosine similarity between the term frequency inverse-document (tf-idf) vector representations of a document and the query term is computed (refer to Section 3.3.3). This model is typically used as a baseline ranking method [3].
- **BM25**: Okapi BM25 is a probabilistic algorithm built on top of the TF-IDF method and has been empirically shown to outperform the regular TF-IDF method in many experiments [23].

- **LMJelinekMercerSimilarity**: A method that uses a statistical language model, e.g., a multinomial distribution over a sequence of one or more words, to represent each document in the collection [21]. Ranking is then performed by computing the likelihood of a document generating a query. This particular implementation uses Bayesian smoothing with Dirichlet priors [28].

- **IB**: A family of information based retrieval models that builds on the core idea that a words' statistical behaviour differs on the document and collection level [9].

3.3.2 Machine Learning

Machine learning aided text classification builds on the ability of the underlying algorithm to correctly learn the essential features that characterize a certain category of documents from a set of given examples, i.e., the training set, so that it can automatically classify new documents in the right category among those it learned.

While the goal of the traditional machine learning classification task is to train a classifier that is able to handle out-of-sample data points, e.g., tasks such as the real-time detection of pedestrians, malicious binaries or OCR, there is a strong overlap between the in-sample and out-of-sample data points in our setting, as we assume that information in the input documents (used for training) are used to generate the output documents. Under these conditions one can argue that a certain degree of overfitting is not only unharmful but in fact beneficial.

All fitting was performed using the open-source Python machine-learning library scikit-learn [20].

3.3.3 Features

For features we compute the tf-idf weights, and represent each document by a high-dimensional sparse vector $\mathbf{x}$, whose x_i entry is the tf-idf weight of the word (token) associated with dimension i. For example, a document denoted by d containing the words 'man', 'missile' and 'aide' would be transformed into the vector:

$$\mathbf{x}_d = \left[\overbrace{x_{d,1}}^{\text{man}} \overbrace{x_{d,2}}^{\text{missile}} \ldots \overbrace{x_{d,N}}^{\text{aide}} \right] \tag{3.1}$$

where N is the vocabulary size, with the word *aide* being mapped to the last dimension. The entries $\mathbf{x}_{d,t}$ are the tf-idf weights defined as[13]

$$\mathbf{x}_{d,t} = \underbrace{\sqrt{f_{t,d}}}_{\text{tf}} \times \underbrace{\left(1 + \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}\right)}_{\text{idf}} \tag{3.2}$$

where $f_{d,t}$ denotes the frequency of term t in document d and D the set of all documents.

We performed experiments using the RandomForest[6], Adaboost[28], SVM [5] and logistic regression [5] packages implemented in sklearn [20]. However, we only discuss the linear SVM multiclass algorithm in further detail as it yielded the best performance. It works by searching for linear functions (one for each class) whose score for the correct class (e.g., *Secret*) is at least 1 higher than the other classes (e.g,. *Unclassified*, *Top Secret*, *Classified*). This idea is mathematically expressed by the non-convex minimization problem:

$$\min_{\mathbf{W}} L(\mathbf{W}) = \underbrace{\frac{1}{M} \sum_i \sum_{j \neq y_i k} \max(0, \mathbf{w}_j^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + 1)}_{\text{data loss}} + \underbrace{\frac{\lambda}{2} ||\mathbf{W}||^2}_{l_2 \text{ regularization loss}} \tag{3.3}$$

Because there is an inbalance between the classes in the dataset, which is a situation one must expect to handle in a real-life deployment, we experimented with the use of a pre-training reweighting mechanism where:

$$y_i^* = \frac{M}{|C| \times |\{y \in Y : y = y_i\}|} \tag{3.4}$$

with $|C|$ and M denoting the number of classes and samples respectively. This has the effect of automatically adjusting the weights proportional to the inverse class frequencies.

4 Evaluation Results

We here first provide evaluation results from using the DNSA dataset (evaluation approach 1), and then additional validation results using the private dataset (evaluation approach 2).

[13] It should be noted that there exists a great number of heuristic variations of the tf and idf formulas.

4.1 Evaluation Approach 1

The training and indexing is performed on the DNSA dataset, which is also used to generate the sets of output documents described in Section 3.1. In order to verify whether a controlled environment improves the detection of content from a classified input document being embedded in some transformed way in an output document, we set up the following experimental set-up with three types of classifiers:

1. **Global**: First we create a global classifier trained on all available documents from the DNSA dataset. This constitutes the baseline accuracy achieved by traditional ML and IR methods, where more information is supposed to give better results.

2. **Domain specific classifiers**: Then we create three more context-sensitive classifiers, where only documents pertaining one of the three subcollections AF, CH and PH are used for training/indexing. This is done as a first verification that a classifier/index set built on more context-specific documents can indeed give better accuracy than one built on possibly more, but less specific data, even when creating the output documents from a relatively diverse and large input set. That is, better results are not only an effect of creating output documents from a small and not very diverse set of input documents.

3. **Dynamic per-user**: Finally we create controlled environments where we gradually increase the amount of imported documents used for training/indexing. This to test our hypotheses that using additional documents for training besides those strictly relevant to the output documents does not necessarily increase accuracy, but rather constitutes noise.

For all three types of classifiers we test how they perform on the different transformations of documents from each separate subcollection as defined in Section 3.1. We also test how they behave when used both as a binary classifier (i.e. Unclassified and Classified labels) and a ternary one (i.e. Unclassified, Confidential and Secret labels). For IR algorithms we use a threshold value $\theta = 0.15$ and we measure also how often they are able to identify the input document used to generate the output document, i.e., the number of exact matches.

If our hypothesis is correct, then dynamic per-user classifiers should provide the best accuracy when used on smaller but relevant input sets, and their accuracy converge toward domain specific and global classifiers as more and more noise is introduced in the input set. Domain specific classifiers should

also provide some better accuracy than global ones. The results we obtained are summarized in Table 4.1 and partially illustrated in Figure 4.1, and confirm exactly this kind of behaviour.

4.1.1 Global Classifiers

We train a classifier using the complete set of documents as the training set, and then measure the predictive performance in terms of accuracy on each of three subcollections AF, CH and PH separately. For ML we only include results for SVM as it consistently outperformed the other ML methods included in our experiments (see Section 3.3) on all output classes.

4.1.2 Domain Specific Classifiers

For the domain specific classifiers we train three classifiers using each of the subcollections AF, CH, and PH separately as training sets. We then evaluate the classifiers on the corresponding subcollections that were used in the training phase. Each classifier provides slightly better accuracy than the global classifier when detecting the classification of output documents generated from the corresponding subcollection.

4.1.3 Dynamic Per-User Classifier

We predict that a smaller input set will give better results because of less noise from potentially unrelated sources. In order to test this, we perform experiments in which we initially start with a sample of 25 classified and 25 unclassified documents. We then iteratively increase the set of input documents (namely the noise) by sampling from the remaining dataset while testing the classifier on outputs generated only from the initial 50 documents. This sequence of steps is then repeated 200 times and the mean accuracy is computed. Figure 4.1 (top) shows how the accuracy, for the "Abstract" output documents, decays as the size of the set of input documents increases. This shows how the classification task becomes increasingly difficult as more noise is introduced into the environment, approaching the performance of the domain specific classifier as the size of the input set increases. For the "Mixture" class we have taken one classified document from the *Input* and one unclassified document from the *Noise* datasets and combined a sample section of (50%) from each. This is meant to illustrate the effect of introducing noise from an external source, i.e., content which is not present in the training set/index. When evaluating the performance for

the "Mixture" documents we measure the accuracy only for Classified output documents.

4.2 Discussion

Table 4.1 displays the accuracy for the ML and IR approaches when evaluated using a Global, Domain Specific and Dynamic Per-User classifier. It is clear that all approaches are robust with respect to the "Rewritten", "Spinner" and "Translation" transformations and further experiments show that only a minor benefit is achieved when deploying a controlled environment for these classes. The high accuracy can be traced back to how these output documents are generated from the input documents and how the SVM and TF-IDF methods works. When using a bag-of-word representation in which the order of words are discarded, the relative frequencies of the input documents remain invariant with respect to the transformations, resulting in what (for the algorithms) are very similar documents. For the "Translation" transformation the intermediate steps distorts the semantics while retaining the TF-IDF weights of the orignal document.

The "Abstract" and "Mixture" categories appear to be more challenging transformations. If we look at the accuracy plots displayed in Figure 4.1, we see how the introduction of the more context aware Domain Specific classifier offers a moderate increase in performance compared to the Global classifier. Proceeding one step further by deploying the Dynamic Per-User classifier yields a significant increase in performance, and the plot illustrates how the performance decays as the size of the input increases until it finally converges to the performance offered by the Domain Specific classifier.

From the graph it is clear that the introduction of a second source of noise further reduces the effectiveness of the algorithms, but that the use of a controlled environment still significantly improves accuracy compared to a global or domain specific classifier. There is a notable drop in accuracy as we go from two to three security classes and when we go from three classes to only measuring exact matches. However the table shows that we still achieve a bump in performance by using a more tightly controlled environment.

From Table 4.1 and Figure 4.1 we can conclude that the IR and ML methods yield comparable performance when deployed using a controlled environment. However, the IR approach offers a few advantages. Firstly, the threshold value θ (Section 3.3.1) limits the viability of *Causatitive* attacks. Secondly, since they work by comparing the output document with a set of input documents, it is straightforward to explain a classification outcome to the end user. Likewise, it

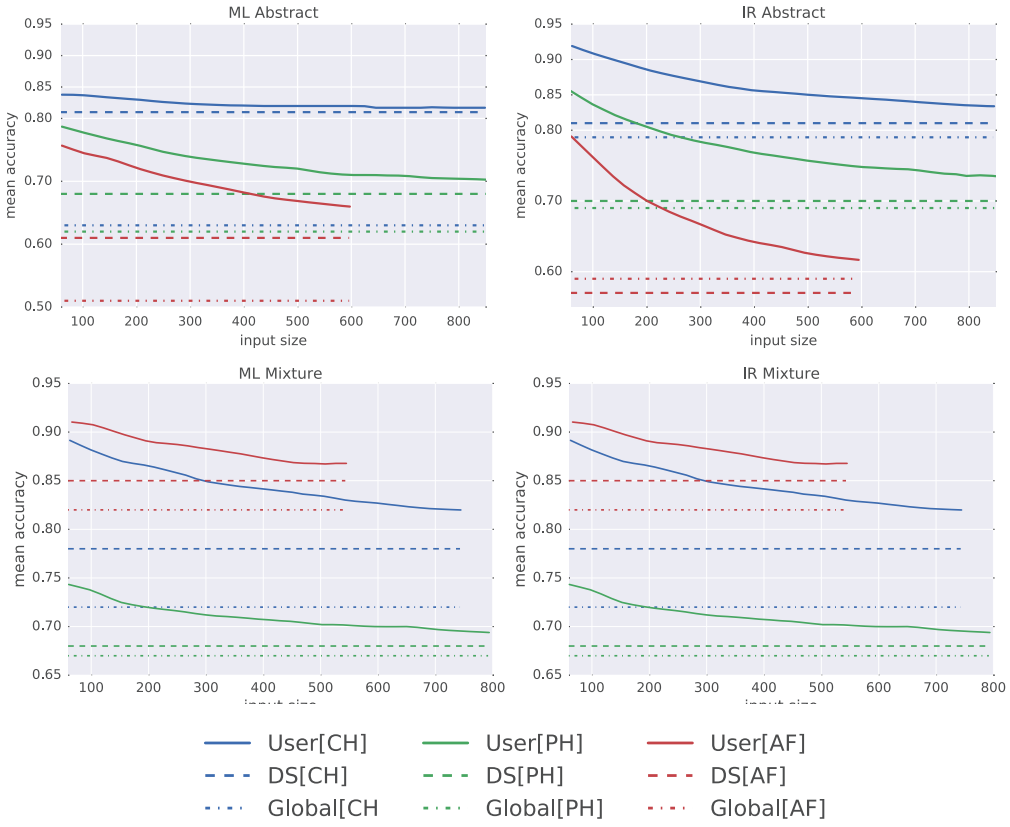

Figure 4.1: Accuracy as a function of the size of the controlled environment for the "Abstract" and "Mixture" output class. Left: Machine learning (l_2-regularized SVM) based classifier. Right: Information retrieval (TF-IDF VSM/Cosine) based classifier. **Legends**: *Global[XY]* a global classifier trained an the complete training set (AF, PH and CH) and evaluated on the XY dataset, where XY can be AF (Afghanistan), PH (Philippines) and CH (China). *DS[XY]* denotes a domain specific classifier trained and evaluated on the XY dataset. *User[XY]* a dynamic per-user classifier trained on the input for a controlled dataset from the XY dataset.

is easy to perform an error analysis by studying the nature of the documents it gets wrong. For example, in our study we randomly sampled a set of classified documents that were misclassified. We discovered that there were several instances in which the best match was an official invitation to an event or meeting, while the classified document, the one used as a query, was an internal memo detailing what the attending officials political position, talking, and view points should be.

4.3 Evaluation Approach 2

The primary motivation behind a second evaluation approach is to investigate whether or not the previous results can be attributed to the artificial way we generate output documents. E.g., if the generated output documents are much less diverse than those created by users of a real-life system, it could be that a larger but less specific training set might outperform the more context-aware given more diverse output documents.

For data we use the internal dataset described in Section 3.2.1, and we evaluate using the IR approach. We did not evaluate the ML classifiers on this dataset as many of the documents did not have enough references to perform the fitting. For each document we index all of its references (and their classification), and use the text of the document as the query string. From the set of results we take the security label of the document with the highest score to be the correct one. All three scoring algorithms have an accuracy of approximately 78%. This might seem like a lackluster result given that the small size of the input set should provide a more meaningful context for the classifier. However, an analysis of the set of misclassifications reveals that most of the incorrectly classified documents are unclassified reports whose best match is the larger and more extensive classified version, which naturally has a large amount of overlap. If we remove these documents, accuracy improves to 89%. As shown in Figure 4.2, there is significant benefit in using a controlled environment compared to a global classifier for this second set of documents, providing further evidence in support of our hypothesis.

5 Related Work

Despite the long history of applying machine learning and information retrieval to the text categorization task [16]; not much research has focused on using these methods to automate security classification in the context of DLP. This

		Transformation	**Binary**			**Tenary**			**Exact**		
IR			Global	DS	User	Global	DS	User	Global	DS	User
	China	Abstract	.79	.81	.93	.66	.68	.86	.33	.36	.75
		Spinner	.94	.94	.99	.99	.99	.99	.99	.99	.99
		Rewritten	.99	.99	.99	.99	.99	.99	.99	.98	.99
		Translation	.99	.99	.99	.98	.98	.99	.97	.97	.99
		Mixture	.72	.78	.89	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Afghanistan	Abstract	.58	.57	.85	.46	.47	.77	.22	.27	.60
		Spinner	.96	.96	.99	.93	.95	.99	.92	.93	.99
		Rewritten	.99	.99	.99	.99	.99	.99	.99	.99	.99
		Translation	.99	.99	.99	.99	.99	.99	.99	.98	.99
		Mixture	.81	.85	.92	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Philippines	Abstract	.69	.70	.85	.50	.56	.78	.30	.30	.66
		Spinner	.99	.99	.99	.99	.99	.99	.95	.95	.99
		Rewritten	.99	.99	.99	.99	.99	.99	.99	.99	.99
		Translation	.99	.99	.99	.99	.99	.99	.99	.96	.99
		Mixture	.67	.68	.75	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	.99	.99	.99
ML											
	China	Abstract	.63	.81	.84	.54	.59	.64	-	-	-
		Spinner	.98	.98	.99	.95	.95	.99	-	-	-
		Rewritten	.99	.98	.99	.96	.97	.98	-	-	-
		Translation	.96	.98	.99	.90	.97	.97	-	-	-
		Mixture	.76	.79	.92	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	-	-	-
	Afghanistan	Abstract	.50	.61	.76	.53	.54	.62	-	-	-
		Spinner	.96	.96	.99	.94	.93	.99	-	-	-
		Rewritten	.99	.99	.99	.97	.96	.99	-	-	-
		Translation	.95	.97	.99	.89	.95	.97	-	-	-
		Mixture	.81	.82	.96	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	-	-	-
	Philippines	Abstract	.62	.68	.80	.45	.53	.62	-	-	-
		Spinner	.97	.99	1.0	.96	.98	.99	-	-	-
		Rewritten	.97	.99	1.0	.98	.97	.99	-	-	-
		Translation	.96	.97	.99	.89	.96	.99	-	-	-
		Mixture	.59	.61	.70	-	-	-	-	-	-
		Unmodified	.99	.99	.99	.99	.99	.99	-	-	-

Table 4.1: Mean accuracy for the SVM and IR methods when deployed using a **global** (Global), **domain specific** (DS) and **dynamic per-user** (User) classifier. The performance is reported when operating with: two security classes (**Binary** - Unclassified/Classified), three classes (**Tenary** - Unclassified, Confidential and Secret) and exact match, i.e., counting only instances where the best match is the input document used to generate the output for the IR approach (**Exact**). The reported controlled environment accuracy is when using a classifier trained with 25 Unclassified and 25 Classified input documents.

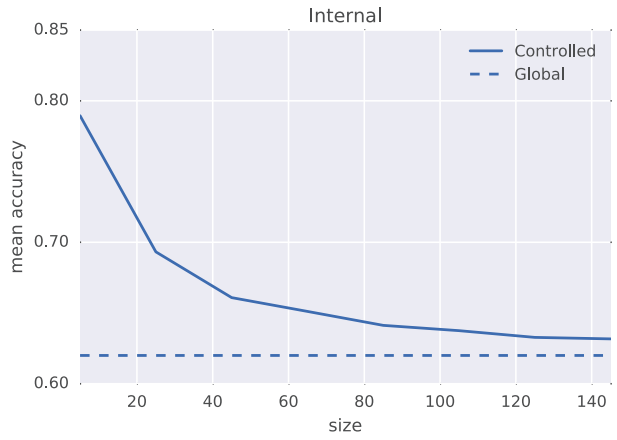

Figure 4.2: Accuracy as a function of the size of the controlled environment for the internal dataset compared to when using a global classifier.

despite being recognized as a relevant area by commercial vendors [27].

The first work we are aware of is that of Brown et al. [7]. This study used a smaller part of the DNSA corpus [1] to investigate whether a ML classifier trained on documents regarding a particular topic or time-period would generalize to other topics/time-periods. Their goal was to develop a tool to aid manual classification and declassification of documents, rather than detect leakages. These results were later validated and expanded upon in [13], and the classifiers then fine-tuned in [10]. In all these cases only unmodified out-of-sample documents were used as a test-set.

The concept of an automatic security classification framework based on machine learning has also been proposed by Kassidy [8], but without any implementation or validation. Lewellen et al. [18] proposed to use plagiarism techniques to detect data leaks. This is mostly a technical document on how to set up a system that uses standard indexing techniques to detect parts of sensitive text copied and pasted, for instance, in e-mails. Another related work uses N-grams statistical analysis to detect sensitive documents even after text spinning [2]. While our work is similar in that we investigate how the classifier precision changes when documents are manipulated or degraded, we use additional manipulation techniques and also utilize a large and more realistic dataset. Besides, their classification is based on topics rather than sensitivity, so it is not possible to directly relate our results. We argue that classifying documents per topic as they do, is easier as a topic can be well-characterized by specific words

and expressions, while sensitivity can be based, for instance, also on political or strategic motivations.

Hart et al. [14] first evaluate various ML algorithms as we do, but then settle on a SVM-based classifier and then develop a new supervised training algorithm that can automatically distinguish between secret, public, and non-enterprise documents. They use various supplement training and adjustment techniques in order to compensate for the imbalance and false positives due to the non-enterprise documents. While we operate with a data set whose security labels have been assigned by government personnel and has later gone through a declassification process, the data used in their study comes from several smaller private sources, and is a mixture of internal handbooks for religious organizations (LDS[14], TM[15]), corporate emails and private blogposts regarding software-engineering. Furthermore, they do not seem to account for possible manipulation of the documents to be classified either, but use only documents known to the classifier without any modification. Shu et al. [26] introduces a scalable sequence alignment algorithm for detecting data leaks in transformed documents. However, their focus is restricted to a special class of transformations, e.g., replacing white space characters with the "+" character.

6 Conclusion

This paper explored the idea of using controlled environments and dynamic classifiers to better determine the security classification of transformed documents, by providing a more relevant context. By deploying a controlled environment that monitors and registers all imported documents, i.e., the documents accessed by a user during a session, we can subsequently check if any documents that are later exported contain information whose origin is a classified source. In our work we proposed two ways to automate this process using methods from the fields of machine learning and information retrieval. A controlled environment can be instantiated per single user, virtual machine, network, or process, potentially for each work session or for longer time periods, and can be re-initialized when required.

In contrast to the traditional approach where a global classifier is trained using all the available data of an enterprise or organization, a classifier in a controlled environment has the advantage of a more targeted context, as it knows which data has been accessed during a session. Extensive evaluation

[14]The Church of Jesus Christ of Latter-day Saints
[15]Transcendental Meditation

presented in Section 4, using a dynamic per-user environment (trained on the documents accessed by the user), a domain specific (trained using documents from the same subcollection) as well as a global classifier (trained using all available data), supports the hypothesis that the introduction of a controlled environment is beneficial to DLP. Especially for difficult detection tasks there is substantial benefit in using a controlled environment as witnessed in Figure 4.1, where we see how the accuracy decays as more irrelevant data is imported into the controlled environment. For the easier detection tasks (results presented in Table 4.1) there was no significant gain to be achieved by deploying a controlled environment.

In order to validate that our results also hold water in a operational setting, i.e., that the observed boost in performance can not be attributed to the artificial way we generate output documents, we conducted experiments (Section 4.3) using a separate data set consisting of a private repository of classified and unclassified reports from our research institution. For each of these reports we assumed that the references were the documents imported into the controlled environment while the report itself was the output at the end of a session.

As part of future work we intend to perform a more fine-grained analysis in which we operate on the sentence or paragraph level, e.g., we want to investigate whether we can detect if a transformed chunk of text in an output document can be traced back to a classified document in the input set.

We also plan to use the predicted label probabilites or scores of the classifiers to build an insider threat meta score, which would help facilitate the detection of anomalous behaviour on the user level, thus mitigating the threat of *Causative* attacks. Furthermore, an insider score would mitigate false positives by analyzing user labeling trends over a longer time horizon instead of operating on a per-document level.

References

[1] Digitial National Security Archive. `"http://nsarchive.chadwyck.com/home.do"`. Accessed: 2015-03-26.

[2] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy. A semantics-aware classification approach for data leakage prevention. In W. Susilo and Y. Mu, editors, *Information Security and Privacy*, volume 8544 of *Lecture Notes in Computer Science*, pages 413–421. Springer International Publishing, 2014.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press, New York, 1999.

[4] M. Barreno, B. A. Nelson, A. D. Joseph, and D. Tygar. The security of machine learning. Technical Report UCB/EECS-2008-43, EECS Department, University of California, Berkeley, Apr 2008. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-43.html`.

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] J. D. Brown and D. Charlebois. Security classification using automated learning (scale): Optimizing statistical natural language processing techniques to assign security labels to unstructured text. Technical report, DTIC Document, 2010.

[8] K. P. Clark. Automated security classification. Master thesis Vrije Universiteit, 2008.

[9] S. Clinchant and E. Gaussier. Information-based models for ad hoc IR. In *Proc. ACM SIGIR conference on Research and development in information retrieval*, pages 234–241, 2010.

[10] P. E. Engelstad, H. Hammer, , K. W. Kongsgård, A. Yazidi, N. A. Nordotten, and A. Bai. Automatic security classification with lasso. In *Proc. International Workshop on Information Security Applications*, 2015.

[11] C. Gormley and Z. Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 2015.

[12] R. Haakseth, N. A. Nordbotten, Ø. Jonsson, and B. Kristiansen. A high assurance guard for use in service-oriented architectures. In *Proc. International Conference on Military Communications and Information Systems*, 2015.

[13] H. Hammer, K. W. Kongsgård, A. Bai, A. Yazidi, N. A. Nordbotten, and P. E. Engelstad. Automatic security classification by machine learning for cross-domain information exchange. In *Proc. IEEE Military Communications Conference*, volume 31, 2015.

[14] M. Hart, P. K. Manadhata, and R. Johnson. Text classification for data loss prevention. In S. Fischer-Hübner and N. Hopper, editors, *Proc. International Symposium Privacy Enhancing Technologies Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011.*, volume 6794 of *Lecture Notes in Computer Science*, pages 18–37, 2011.

[15] Intel Security. Grand theft data - data exfiltration study: Actors, tactics, and detection, 2015. URL `http://www.mcafee.com/us/resources/reports/rp-data-exfiltration.pdf`.

[16] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. 1398:137–142, 1998.

[17] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009. ISBN 9780131873216. URL `https://books.google.no/books?id=fZmj5UNK8AQC`.

[18] T. Lewellen, G. J. Silowash, and D. L. Costa. Insider threat control: Using plagiarism detection algorithms to prevent data exfiltration in near real time. Technical Report CMU/SEI-2013-TN-008, Carnegie Mellon University, 2013.

[19] E. Ouellet. Magic quadrant for content-aware data loss prevention. Gartner Inc., 2013.

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

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, 1998.

[22] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. pages 45–50, May 2010. `http://is.muni.cz/publication/884893/en`.

[23] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends in information retrieval*, 2009.

[24] P. Rosso and B. Stein. Overview of the 6th international competition on plagiarism detection.

[25] A. Shabtai, Y. Elovici, and L. Rokach. *A Survey of Data Leakage Detection and Prevention Solutions*. Springer, 2012.

[26] X. Shu, J. Zhang, D. Yao, and W.-C. Feng. Fast detection of transformed data leaks. *IEEE Transactions on Information Forensics and Security*, 2016.

[27] Symantec. Machine learning sets new standard for data loss prevention: Describe, fingerprint, learn, 2010. URL `http://eval.symantec.com/mktginfo/enterprise/white_papers/b-dlp_machine_learning.WP_en-us.pdf`.

[28] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.

Paper V

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

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

Kyrre Wahl Kongsgård, Nils Agne Nordbotten, Federico Mancini and Paal E. Engelstad

Abstract

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

1 Introduction

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

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

A more concrete example is shown in Figure 1.1 which depicts the architecture of a DLP system, in which a data guard connects two security domains, and where a machine learning classifier has been constructed using all labeled

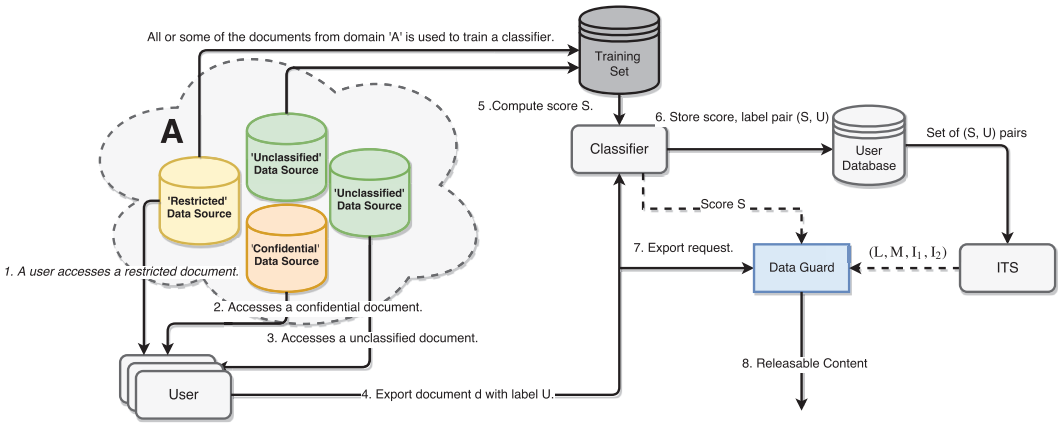

Figure 1.1: A network environment where users can access sensitive information. Each document exported out of the domain must have a security-label attached, which must satisfy the policy enforced by the data guard (e.g., only *Unclassified* content may be exported). By introducing a machine-learning classifier inferring the sensitivity level using textual contents (e.g. word frequencies), we can detect instances where the user knowingly or unknowingly violates the security policy by exporting classified material. The ITS component uses the set of user assigned labels and predicted scores to infer the misclassification rate for each user.

or known sensitive documents. Whenever a transfer request is made, the DLP algorithm first analyses the textual content of the data object and computes a predicted security label. If the predicted security label is more restricted than the one assigned by the user the transfer can be dropped and/or the security team notified. The method may also be applied in cases where security labels are not used, e.g., assuming a classifier trained on a set of known sensitive company documents, as any document exported from the network may be considered released by the exporting entity (i.e., implicitly labeled as non-sensitive).

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

high accuracy, detection of transformed data leaks is much more difficult with a higher rate of false positive/negatives [8]. In such cases, sensitive content may for instance have been rewritten or included as part of another document.

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

$$\mathbf{risk} = \overbrace{\mathbf{likelihood}}^{\texttt{Threat Score}} \times \underbrace{\mathbf{consequence}}_{\texttt{Misusability Score}} \tag{1.1}$$

Intuitively, the misusability score represents the damage an entity can cause if it leaks data, however, it says nothing about the likelihood of the entity actually leaking data. In a context where the data objects are textual documents, the "consequence" factor is represented by a misusability metric such as the TM-Score which can be readily computed [2], while the "likelihood" factor corresponds to the ITS, whose derivation and discussion is the topic of this paper.

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

2 An Insider Threat Score

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

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

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

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

2.1 DLP Algorithm

The ITS concept is generic, in the sense that it is applicable to any numerical DLP procedure[16]. As a basis for further experimentation we rely on the dataset and pre-processing done by by Hammer et al. [5]. Using this dataset derived from the Digital National Security Archive we train a binary class logistic regression classifier to distinguish between "classified" and "unclassified" documents. The trained classifier achieved an accuracy of 0.84 on the test set.

2.1.1 Error Sources

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

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

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

2.2 A Generative Approach

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

We can approach the problem by introducing and examining a simplified model of the process of users assigning security labels to data objects. Given a document X to be classified and a user, let $S \in [0, 1]$ denote the random variable whose value represents the probability that the document is classified (as estimated by the machine learning classifier), $U \in \{u, c\}$ the user assigned

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

label, $L \in \{u, c\}$ (i.e., unclassified or classified) the correct label (unknown), M a variable representing whether or not the document was mislabeled (unknown) and I the overall mislabel propensity. Figure 2.1 visualizes the dependency structure of these variables in terms of a directed graphical model. The links and their directions reflects a generative model of how a user performs the labeling process.

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

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

2.2.1 Bayesian Network Model

If we specify the conditional distribution of each random variable we get the Bayesian network shown in Figure 2.1. From the observed values of U and S we can infer the posterior density for I_2 and use its mean as the ITS.

In order to perform the fitting we utilized the software package Infer.NET and its implementation of the expectation propagation algorithm [11], while the data was generated using instantiations of the corresponding models implemented in PyMC3 together with the NDSA corpus [7]. By using a probabilistic programming framework like Infer.NET we were able to quickly implement and experiment with different models. The alternative, i.e., manually implementing each model, would have been a huge undertaking. However, by relying on Infer.NET we ran into a few restrictions when designing the model[17], e.g., it was not possible to create priors for beta distributions and it proved difficult to learn common Y_1 and Y_2 distributions directly from the complete set of observed user scores. In order to prevent situations in which a skewed distribution would completely dominate the model all users were made to share a common distribution L_λ.

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

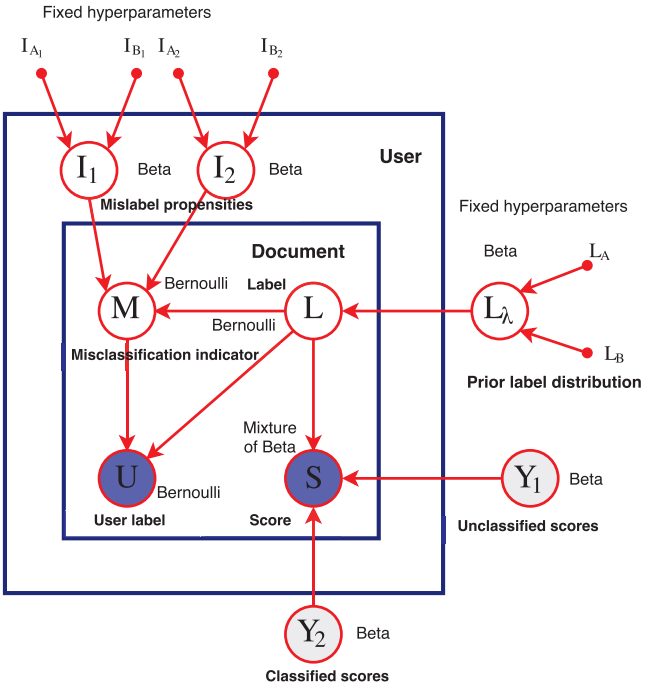

Figure 2.1: Graphical model expressing the dependency structure of the Bayesian network, and the distributions for each of its nodes. The node U denotes the user-assigned labels, S the predicted scores, M the document mislabel mislabel indicator, L the true label and I the mislabeling propensity. The two variables shaded gray Y_1 and Y_2 are the known score distributions for classified and unclassified documents respectively. Observed variables are shaded blue.

2.2.2 Manipulation

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

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

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

2.2.3 Evaluation

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

Per-document performance: We want to compare the difference in performance, for a per-document classifier, for the three strategies in which we use

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

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

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

Sample size: We also measure how many samples, i.e., the number of exported documents, that is needed to establish the true misclassification rate for a user. Figure 2.2 (left) displays the relative error of the estimated value of $\mathbb{E}[I_2]$ as a function of export size. For the depicted graph the insider had a misclassification rate of 0.10. As expected, the higher the misclassification rate is the less samples are needed. Although a significant number of documents (> 600) is required to completely determine the true value, even with a small number of documents the estimated mean is larger than those of regular users.

Insider detection: In order to test how effective the value $\mathbb{E}[I_2]$ is in detecting insiders we generate the set of users, compute their latent distributions and then rank them according to the size of their sample means. From this ranked list of users we can compute the mean average precision at k (MAP@k)[18].

The graph of Figure 2.2 (right) shows the MAP@k (for $k \in [1, 10]$) when the insider user has a misclassification rate of 0.10 and 100 exported documents. From a set of a 1001 users it is not unreasonable for a security team to investigate and audit a set of $k > 10$ users, in which case the performance is high (≈ 0.95).

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

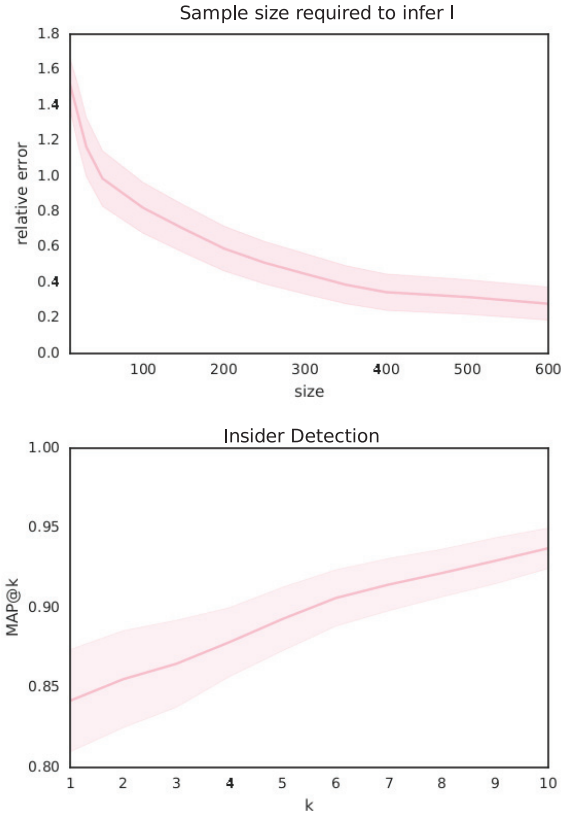

Figure 2.2: Left: Relative error of the inferred estimate $\mathbb{E}[I_2]$ plotted as a function of the sample size. **Right**: Mean average precision at k for detecting the insider user among a set of regular users. Y-axis corresponds to the shape parameter $I_2 \sim \texttt{Beta}(a, 10.0)$.

2.2.4 Counter-measures against Manipulation

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

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

For each user we construct two multidimensional vector representations for unclassified and classified score distributions respectively. The entries in these vectors belong to one of two classes:

- 1. summary statistics derived from the score distribution that represent the degree to which the user assigned and predicted security labels deviate and
- 2. metrics that mitigates the attempt by the attacker to manipulate one or more corresponding score components, e.g., the number of exported and imported documents.

As the number of entries increases so does the effort required for a malicious user to blend in with the regular ones. Ideally, the vector entries would be connected or paired in the sense that if a user was to manipulate one of them it would lead to anomalous values in the connected component.

The list of proposed summary statistics used in our experiments includes the: *mean*, *median*, *skew*, *kurtosis*, *standard deviation*, *variance*, *interquartile range*

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

Table 2.1: Manipulation Detection. Performance measured in terms of "Mean Average Precision at k" (MAP@k), for a set of 1001 users. Regular users characterized by $\mathbb{E}[I]\approx 0.01$ while the insider user has $\mathbb{E}[I]\approx 0.07, 0.13, 0.23$.

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

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

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

Table 2.1 displays the performance of the PoC manipulation detection method. The results in the table were achieved without using the monotonically non-decreasing (summation) class of sub-scores. As expected, it shows that the higher the intitial misclassification rate is, then the the easier detection becomes, and that at least certain forms (see description above) of manipulation attacks can be detected through anomalies in the observed user score distribution(s). Future work should seek to incorporate these defense mechanism within a single graphical model.

3 Identifying Internal Threats

Having introduced a method of computing internal/insider threat scores; we will in this section demonstrate how they can be applied in practice. We primarily

focus on how the ITS allows us to equip IT security personnel with visualizations/overviews that help ease the cognitive load of day-to-day operations by providing situational overviews of the risk each user poses to the organization,

Equation 1.1 shows how the ITS and misusability weight for a user can be combined to compute the data leakage risk he poses to the organization. These risk estimates facilitates the construction of interactive visualizations providing an overview of the risk each user poses to the organization.

We assume a scenario akin to the one presented in Figure 1.1, and a user behavior model in which malicious users are characterized by the following set of traits:

- 1. A high misclassification rate (captured by the ITS),
- 2. Sudden bursts in activity (periods of increased volume) and
- 3. Irregular activity, e.g., a user that suddenly starts coming into work on weekend mornings or outside of regular business working hours in general.

For each user and time period we can then compute the associated risk-estimate or ITS and display it in terms of interactive visualizations. Figure 3.1 displays a "calendar heatmap" of the daily ITS for three users in the period January-October. A darker shade of green signifies a higher ITS value. The underlying data was generated from experiments implemented and performed using the discrete-event simulation framework SimPy, in which the inter-arrival time between events (exported documents) is modeled by an inhomogeneous Poisson point process. The three users depicted represents a malicious insider, a regular user and an incompetent user.

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

4 Related Work

To the best of our knowledge no prior research has discussed the concept of an ITS built on the output of existing DLP systems. However, our work is influenced by related concepts such as misusability weights. The M-Score [1] and the TM-Score [2] algorithms provide a means of quantifying the damage an organization would incur if the data a user has accessed would leak to 3rd

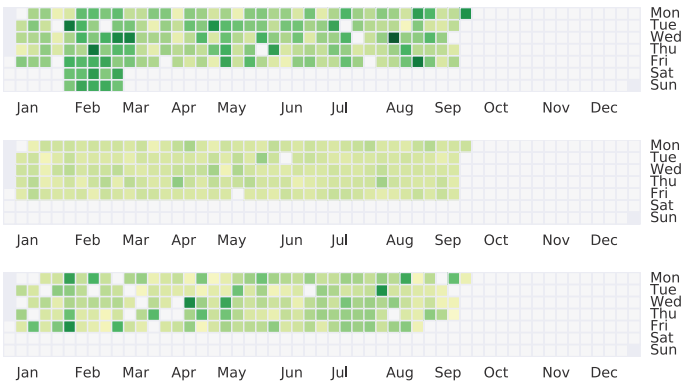

Figure 3.1: Top: *A malicious user that has a very high baseline misclassification rate.* **Middle**: *A regular user with a low misclassification rate. Inactive during weekends.* **Bottom**: *Incompetant user with a high misclassification rate.*

parties. In order to assign a score, domain experts are consulted to manually assess the sensitivity on a per record or document level for a subset of the corpora. These metrics allow us to measure and analyze the consequence of potential data loss on a per user level, and can be further applied to enhance existing anomaly detection algorithms or access control mechanisms.

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

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

5 Conclusion

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

particular we have focused on incorporating the user properties associated with the act of mislabeling data objects.

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[17] B. Vielmetti. Chinese engineer accused of stealing trade secrets from GE, 2014. URL `https://goo.gl/ff9yif`.

Paper VI

Data Leakage Prevention for Secure Cross-Domain Information Exchange

Data Leakage Prevention for Secure Cross-Domain Information Exchange

Kyrre Wahl Kongsgård, Nils Agne Nordbotten, Federico Mancini and Paal E. Engelstad

Abstract

Cross-domain information exchange is an increasingly important capability for conducting efficient and secure operations, both within coalitions and within single nations. A data guard is a common cross-domain sharing solution that inspects and validates that the security labels of exported data objects are such that they can be released according to policy. While we see that guard solutions can be implemented with high assurance, we find that obtaining an equivalent level of assurance in the correctness of the security labels easily becomes a hard problem in practical scenarios. Thus, a weakness of the guard-based solution is that there is often limited assurance in the correctness of the security labels. To mitigate this, guards make use of content checkers such as dirty word lists as a means for detecting mislabeled data.

To improve the overall security of such cross-domain solutions we investigate more advanced content checkers based on the use of machine learning. Instead of relying on manually specified dirty word lists, we can build data-driven methods that automatically infer the words associated with classified content. However, care must be taken when constructing and deploying these methods as naive implementations are vulnerable to manipulation attacks. In order to provide a better context for performing classification, we monitor the incoming information flow and use the audit trail to construct controlled environments. The usefulness of said deployment scheme is demonstrated using a real collection of classified and unclassified documents.

1 Introduction

The need for efficient information exchange within national armed forces, coalitions, and between military and civilian entities has received significant attention in recent years. This need is in strong contrast with the traditional approach to securing classified military systems, where isolation of security domains and

information systems has been the default approach. Thus, concepts such as NATO's Information Exchange Gateways (IEGs), and similar initiatives within the nations, have been established to enable cross-domain information exchange in a secure manner.

These cross-domain solutions are required to perform various security controls, (e.g., information flow control, antivirus, and access control) to ensure that the interconnection does not compromise confidentiality, integrity, or availability. In addition, non-security specific requirements such as what type of information needs to be exchanged (e.g., friendly force identification, chat, or documents), and protocol specific details, may also impact security and what type of security controls are required. A key challenge, particularly when interconnecting domains at different classification levels, is to ensure sufficient assurance in the information flow control so that classified data is not leaked.

Solutions for collaboration and information sharing across security domains may generally be categorized as transfer solutions or access solutions. A transfer solution enables the transfer of information from one domain to another, while an access solution provides a user access to services and/or information within another domain without logically transferring the information from that domain. In the latter case the access solution itself may be viewed as an extension of the domain to be accessed, imposing the domain separation requirements on the access solution (e.g., a thin client connected by a secure tunnel). Transfer solutions may be further categorized based on their ability to provide one-way or two-way transfer. E.g., one-way data diodes are frequently used when information needs to be moved from a lower classified domain to a higher classified domain, while two-way information exchange may be enabled using a security filter or guard. We here use the term guard to refer to solutions basing their release decisions (at least partly) on security labels, while it may otherwise perform similar checks as a security filter (e.g., ensuring that data objects are according to some predefined format).

Assuming that security labels are correct, a guard may provide stronger security than a security filter alone, as a security filter typically may be bypassed by anyone knowing the allowed message format. This may to some extent be mitigated by having the security filter authenticate senders, but the use of security labels nevertheless provide an additional layer of security and also better applies to content whose sensitivity typically can not be determined by its format or type, such as documents, emails, or chat messages.

Before a user or service can initiate a request to export a data object, it must first be assigned a security label. This label is cryptographically bound to the data object. While the integrity of the data object and security label as

such is cryptographically protected during transfer and storage, it is much more difficult to ensure that the correct security label is attached in the first place. For instance, if a RESTRICTED document is labelled as UNCLASSIFIED, it may result in it being released to an unclassified environment (i.e., leaked). Such mislabelling may be due to human or technical errors, or be due to users or malware trying to bypass security controls.

While the use of high assurance operating systems and applications may significantly reduce the risk of technical errors and malware, the use of commodity general purpose operating systems and applications are often mandated due to practical and economical reasons. This lack of assurance in end-user systems may in some cases be mitigated by labelling data objects based on origin, where a potentially high assurance intermediary mechanism (e.g., gateway) labels all data from a given origin (e.g., computer or network) with a given classification (e.g., RESTRICTED). However, this approach would not allow documents from the same origin to have different security classifications. Thus, while applicable in some scenarios, this approach is often too inflexible to be practical. In the more general cases, the security label needs to be determined based on the content, rather than the origin, of the data object.

To mitigate the risk of incorrect security labels, another layer of protection in the form of a *content-checker* may be applied. For text-based data objects a "dirty word list" is often used, which scans the object for the presence of keywords that are often associated with classified content, e.g., security classifications, certain technical terms, locations, and project acronyms. The effectiveness of these content checkers are fully dependent on the quality of the rather static dirty word list in use. Given more recent advances in use of machine learning, data-driven content checkers based on machine learning have the potential to improve security of guard based cross-domain solutions.

This paper highlights our experiences in developing secure, scalable and robust cross-domain solutions (using data guards) and methods for increasing the assurance in the correctness of the user or application assigned security labels. Furthermore, it provides an in-depth view into the security challenges faced when using machine learning to create data-driven content-checkers.

The remainder of the paper is organized as follows. Section 2 discusses the design and architecture of two guard prototype implementations for handling the exchange of SOAP and XMPP messages, showing how such solutions can be realized with high assurance. In Section 3 we discuss how machine learning methods can be utilized for DLP in a cross-domain setting to mitigate the limited assurance in the correctness of security labels, the security concerns that must be addressed when doing so, as well as engineering aspects such as

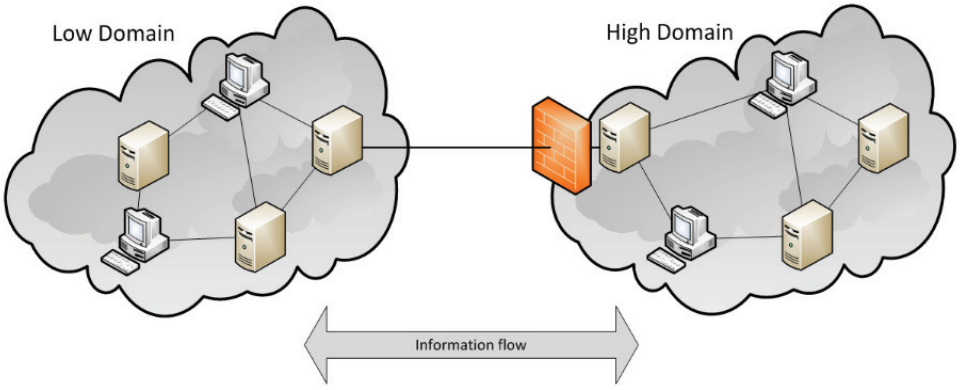

Figure 1.1: The data guard enables two-way information flow between a "High" and "Low" domain. Each object passing through the guard will has its security label validated, its content checked according to policy/configuration (e.g., content may be scanned for malware and the presence of "dirty words"), and its sender and destination fields may be verified and subject to access control. Having passed these checks, the object is then released on the condition that its security label is such that it is considered, as specified by the governing security policy, to be releasable.

which features to include and how to train and deploy said methods in order to maximize both the security and performance of the end-system. Experiments regarding the detection of data leaks are then conducted and the results discussed. Section 4 surveys related work while Section 5 summarizes the paper and highlights our contributions.

2 Prototype High Assurance Guard

In cooperation with Thales Norway AS we have developed two prototype guard implementations, the first for use in service-oriented architectures (SOA) and the other to support cross-domain chat. The first guard [4] supports SOAP messages, as used in Web services, while the chat guard supports XMPP messages. Both guards are based on the core of a military messaging guard being developed by Thales Norway and target a Common Criteria EAL 5 certification. The guards are in alignment with the HAAG protection profile proposal [11] and uses the proposed STANAG 4774 for XML confidentiality label and STANAG 4778 for binding label and data.

Fundamental to the guard design is the use of a high assurance separation kernel. While many different guard implementations exist, most of these are

based on medium assurance operating systems effectively preventing evaluation at higher assurance levels. The separation kernel ensures that different partitions (e.g., virtual machines or processes) cannot influence each other except by using well-defined interaction mechanisms. This allows security critical functions to be separated and protected from non-critical functionality and helps ensure least privilege and non-bypassability. Together the strong separation, high assurance, and ability to control communication between components (i.e., partitions) makes for a good environment to build high assurance systems.

Functionally the guard is separated into several different components, each implemented as one or more partitions. Central to the design is the core component which ensures that each object passed to the guard is processed correctly. This includes subjecting the object to label and signature checks, content checking and other access controls configured. Content checking is done through a separate component which provides a generic plug-in interface for content checkers. Depending on the scenario, different content checkers (e.g., malware scanning and/or format checking such as XML schema validation) can be included as needed. This architecture allows new content checkers to be added without risk of compromising other guard components.

Protocol adapters provide the interface towards the interconnected domains. Different protocol adapters are used to handle the specifics of a given protocol, e.g., XMPP or SOAP/HTTP. The main task of a protocol adapter is to extract protocol dependent attributes and transform these to protocol-independent attributes used by the core component. Additional components are used for handling configuration and PKI. This architecture makes it easier to add new protocols without changing the security critical code of the core, and thus simplifies the certification process.

The guards' primary release control mechanism is label checking. A Mandatory Access Control (MAC) policy specifies the label ranges allowed to pass and how to handle unlabelled or incorrectly labelled objects. Other controls are also available for configuration, inlcuding allowed source and destination addresses, integrity control, and content-checkers. When multiple such controls are in effect, a message may be blocked from release if failing any one of these checks.

To support different applications and information exchange requirements, guards will have to handle messages and protocols of varying complexity. SOAP is for instance a fairly simple protocol, while XMPP is more complex. The functional needs must always be balanced against the need for security protection. Examples of this includes presence status, which in XMPP chat provides information about who is logged on and available. This information is very useful

for the users, but can also be highly sensitive since it can reveal who is on duty and allow mapping of work schedules. Whether or not to allow this information to flow between security domains depend on the scenario and the level of risk acceptance. Support for this is given through configuration of the guard. Messages may also have different types of attachments, that may pose their own security risks and may require separate content checkers. Again, what is to be allowed needs to be determined be weighing the operational gain against the additional security risk.

The prototype guards are designed with high assurance certification in mind and the risk of information leakage due to compromise or malfunctioning of the guard is thus minimized. However, the trustworthiness of the primary release control mechanism, label control, is limited by the trustworthiness of the security labels themselves.

As of now, we do not have tools that can autonomously estimate the sensitivity of a text with a degree of confidence high enough to reduce the risk of information leakage to an acceptable level. Simple dirty word lists are quite limited. It is then natural to investigate the possibility of developing more effective automated classification techniques by using recent advancements in the field of machine learning. The second part of the article will be devoted to the latest research on this topic.

3 Machine Learning-Based Content Checking

Machine learning lends itself naturally to the problem of classifying unstructured textual information. However, much of the available research focuses on classification of text into a set of predefined topics, which is not directly applicable to our problem. The sensitivity of a text does not depend only on what it talks about, but also on the context in which it was produced and what kind of damage the information can do if leaked. This type of assessment is difficult even for a person, let alone for an algorithm.

As long as existing data has a known classification, it is possible to verify that it is not labeled incorrectly by employing direct comparison techniques like hashing. Estimating the sensitivity for completely new [3, 5, 12] or heavily processed (rewritten, summarized etc.) [7] information on the other hand is more challenging and is better handled using machine learning. By presenting the algorithm with examples of known classified and unclassified documents it will attempt to infer which features that are associated with each of the target classes (classification levels).

In order to further improve the performance, we worked on two ideas. The first consists in training the algorithm with an even more specific context to increase the accuracy rate. The other explores the possibility to improve the probability of detecting users (or other entities) with an abnormal amount of likely misclassifications over time, rather than aiming at detecting misclassification of single documents. In a guard setting, automatic classification may also be used to prioritize which documents are to be subject to manual review, in which case a somewhat lower classification accuracy may be acceptable.

In the remainder of this section, we will provide an overview of the challenges faced and the state of the art in developing and engineering machine learning-based content-checkers for the cross-domain information exchange setting.

3.1 Features

Before any learning can take place, the documents must be transformed into feature vectors. Feature engineering refers to the process of capturing an important characteristic of a document as a numerical value (feature). It is the part of the machine learning process that requires the most in-depth domain expertise, and is, together with the size/quality of the training data set and the choice of model class, what has the greatest impact on the performance of the resulting model[19].

Features for textual content are primarily derived from variations of word counts/frequencies, but one also uses more general features such as the average sentence length, the number of capitalized words and statistics regarding punctuation. Advanced features such as the part-of-speech (PoS) and named-entity recognition (NER) tags of words in a sentence are also beneficial for certain classes of tasks. A list of the features that we have used for the machine learning-based content-checker include:

- ***N-gram***: An n-gram is a contiguous sequence of n words. Term-frequency inverse document-frequency weights (TF-IDF) modifies these frequencies such as to better reflect the importance of a particular n-gram for the document. In the bag-of-words (BoW) model a document is represented as a multiset of its n-grams. While the BoW model discounts word order (expect for what is captured within the n-gram) and any grammar, it

[19] In the field of representation learning it has been demonstrated that one can, for some tasks, with a sufficiently large data set and the right network architecture automate the feature engineering phase. This is one of the reasons behind the resurgence and popularity of research into deep learning methods.

retains the semantic aspects and has been shown, despite its simplicity, to be very useful for text classification and in information retrieval systems [9]. N-gram frequencies can also be computed on the character level.

- ***Lemmatisation***: Lemmatisation is the process of grouping together the inflected forms of words, e.g., "flies" is mapped to "fly" and "better" is mapped to "good". This pre-processing step could be beneficial for sparser documents and for detecting paraphrased content and the use of synonyms.

Features can be extracted in parallel with their outputs concatenated into a single high-dimensional vector representation.

3.2 Controlled Environments

In a cross-domain scenario each data access request in the "High" domain can be logged on a per-user/session level. The audit trail of access requests, or the incoming information flow, can be used to derive what we have named controlled-environments. A controlled environment refers to any environment where we have control on all imported documents and their respective security classification. The set of imported documents, e.g., those accessed by the user during a session, is defined as *input*, and any new document(s) generated within the controlled environemnt is defined as *output*. By using the set of input documents as basis for the classifier, thereby reducing the noise in the classification process as the input documents are more relevant to the output, we can more accurately estimate the classification of output documents [7]. Our proposed solution inspects the information flow to the controlled environment as shown in Figure 3.1b, and estimates the classification of output documents based on the information about the input documents.

Experiments We want to analyze the performance for message-like (i.e. short) and using both a controlled environment setting as well as a traditional global classifier (one that uses the complete set of documents for training).

As a data set we use a subset of de-classified documents from the Digital National Security Archive. From this repository we extracted the three sub-collections:

- *Afghanistan: The Making of U.S. Policy, 1973-1990*;

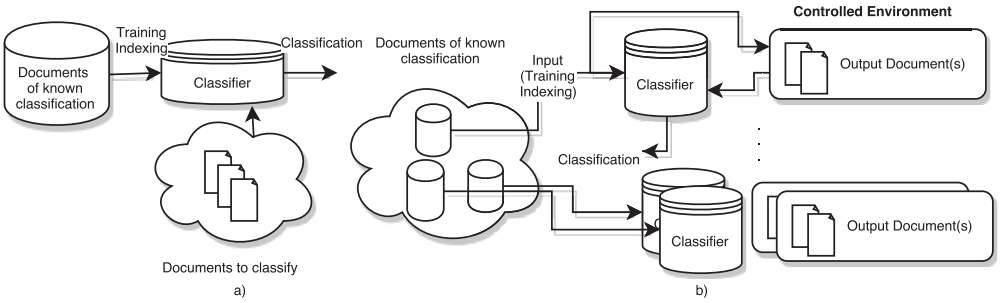

Figure 3.1: a) Usually, a classifier used in DLP is trained on all available documents **b**) With a controlled environment, only the documents of known classification accessed from the environment are used to train the classifier, which in turn is used to classify documents generated within the environment. Multiple controlled environments can exist simultaneously, each characterized by its own input and output.

- *China and the United States: From Hostility to Engagement 1960-1998* and
- *The Philippines: U.S. Policy during the Marcos Years, 1965-1986.*

These were chosen because they contained a mix of both classified and unclassified documents from unrelated domains and from partially overlapping time periods.

We train the classifiers (l_2-regularized logistic regression) on documents, from the DNSA data set, that were imported into a controlled environment and then evaluate the performance on the corresponding abstracts (these are removed from the input documents). This procedure simulates how (potentially classified) information is transformed into new documents. We have also studied other transformation models, e.g., the mixing of documents and the use of synonym (phrases), but we omit them from further discussions as the "abstract" transformation is the most realistic and challenging one. In order to assess how well this methodology performs on short (e.g., message-like) documents, we use DNSA documents as the input/training data and evaluate it on sentence(s) sampled from the corresponding abstract. For comparison we also train global classifiers that use all the available data as a training set. The leakage of known unmodified documents can be detected with very high accuracy (0.99) using both methods, and is not discussed further as this can be handled using existing methods (e.g., hashing).

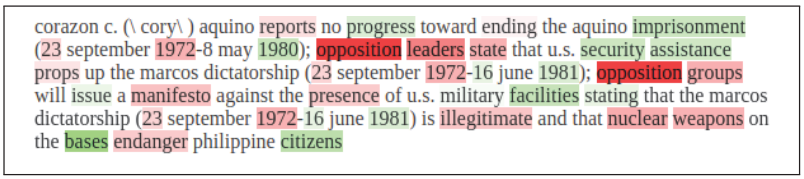

Figure 3.2: Words highlighted in red are those associated with the classified content class, while green words are those associated with the unclassified content class. The darker the color the stronger the signal/connection between the feature and the class is. The intra-class ranking of features are affected by normalization/scaling factors and assumes the set of features to be independent. This analysis is meant to offer a cursory insight into the underlying decision mechanisms and not a rigorous feature selection discussion.

Figure 3.2 shows a visualization of how the classifier analyzes a document to determine its sensitivity level. The words highlighted in red indicates terms that are associated with the more sensitive class. For the particular example shown, it is clear that the model has learned that words such as "opposition", "nuclear", and "endanger" are often linked with sensitive information, while the terms "progress", "citizen", and "imprisonment" on the other hand are more likely to signify a non-classified document.

A comparison between a controlled environment and global deployment scenario is presented in Figure 3.3. It shows the accuracy of the model as a function of the number of sentences from the abstract that is sent as a message. Comparing the two graphs depicted, we see that we are able to achieve a significant boost in performance when using the per-user trained (i.e., controlled environment) model instead of the traditional global classifier. While this is a surprising result, and one that seemingly contradicts the conventional wisdom that more data always provides higher accuracy, it reflects that determining sensitive content is very context dependent and that we are exploiting the assumption that any classified content can be traced back to information contained in the imported documents. As such, a controlled environment would likely result in severe performance degradation if we wanted to use it to detect classified information that is completely unrelated to the input documents.

3.3 Internal Threat Scores

As the content-checkers are plagued with non-trivial false positive rates, we have also investigated the idea of constructing a meta-score called *Internal/Insider*

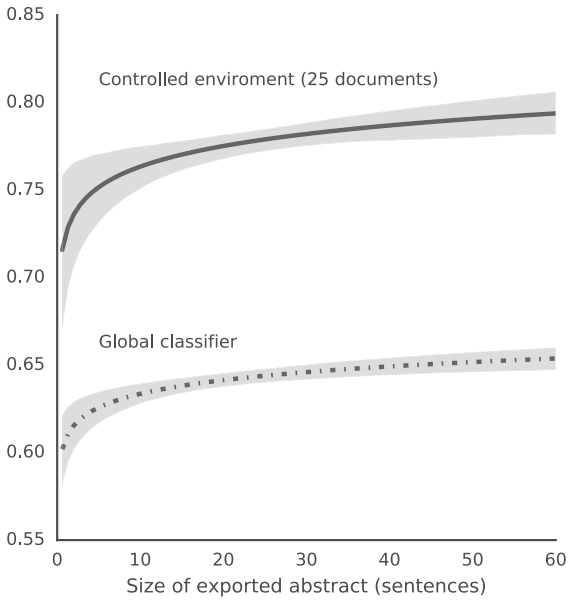

Figure 3.3: Content-checker performance. Accuracy as a function of the number of sentences in the exported abstract sample, for both a controlled environment of size 25 and a global classifier using data derived from the DNSA dataset.

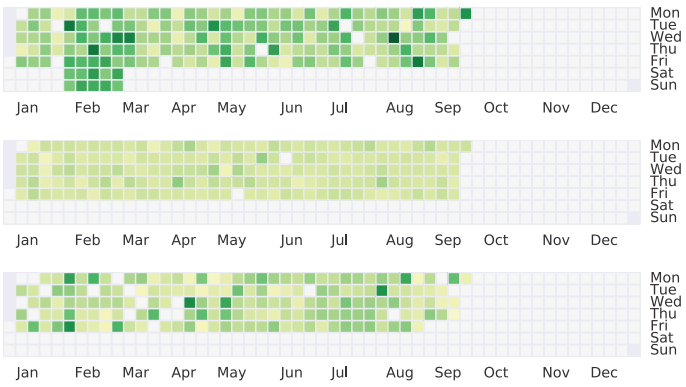

Figure 3.4: Top: *A malicious user that has a very high baseline misclassification rate.* **Middle**: *A regular user with a low misclassification rate. Inactive during weekends.* **Bottom**: *Incompetant user with a high misclassification rate. A darker shade of green signifies a higher ITS value.*

Threat Score (ITS) that uses the aggregated confidence scores to detect long-term discrepancies between the user-assigned sensitivity level and the sensitivity level predicted by the machine learning model [8]. It works by modeling how the users (or other entities) assigns labels as a generative process and then infers (using a Bayesian network model) the latent variables that describe how often documents are misclassified in general for each user. These misclassification rates are what we use to compute the ITS. On a more technical level the model captures to which extent the deviations of the confidence score distribution for one user and the confidence score distribution for known classified documents can be attributed to incorrectly assigned labels by the user. By operating on a per-user (as opposed to per-document) level the number of false alarms is reduced. Figure 3.4 displays a visualization of the ITS.

Another concern that must be analyzed and addressed is the threat of the content-checker itself becoming a target for an attacker.

3.4 Secure Machine Learning

A core underlying principal behind most machine learning algorithms and tasks is that the training and evaluation datasets are generated from the same unknown distribution, i.e., it assumes a stationary environment. Under this assumption, minimizing the empirical risk (informally - the error) on the smaller

training data set, which have often been painstakingly hand labeled, is equivalent with minimizing the risk on the larger evaluation data set. However, this assumption is violated for security tasks such as intrusion detection and DLP systems, where one must take into account the possibility of attackers actively seeking to by-pass detection by manipulating the classifier itself. A machine learning algorithm is said to be *secure* if it performs adequately when deployed in adversarial conditions.

Security assessments of machine learning systems is conducted with respect to the three axis [2]:

- ***Influence***: A user can influence the learning system by conducting either: a causative or an exploratory attack. Causative (interchangeably: poisoning) refers to manipulating (parts of) the training data with the intention of exerting control of the learning process. Exploratory refers to inducing and exploiting a misclassification, e.g., by rewriting a classified document such as not to trigger the content-checker.

- ***Security Violation***: Security violations takes on one of two forms: integrity, e.g., sensitive content being incorrectly classified and let through the guard, and availability, e.g., non-sensitive data being misclassified en masse, which may effectively render the system useless.

- ***Specificity***: The scope of the attack. It can be either a targeted or an indiscriminate attack.

An attack of particular concern in a content-checker context, is the possibility of data leaking from the model itself. If a user knows the functional form of the classifier (e.g., whether we are using a logistic regression, SVM, or some other model), and if the user can probe it for numerical outputs, i.e., if he has access to the probability/confidence scores for each data point, then through repeated experiments he can recover the actual parameter values of the underlying model or (parts of) data points in the training set [6]. When the training set contains classified information one must be particular wary of the possibility of the model leaking data.

We can analyze the security risks for the inferred model with respect to the attack categories/classes, estimate the feasibility of said methods, as measured in terms of the cost (risk and resources) incurred by the attacker, and propose potential mitigation steps:

- ***Exploratory*** (*Insider attacker*): A malicious user can, in theory, always bypass the detection mechanism by rewriting a document such as not to

trigger the alarm. While this procedure can be automated for images [10], it remains a manual process for unstructured text. Taken to its extreme, we arrive at a scenario in which the insider employs methods of steganography to covertly embed classified information within other innocuous content. There does not exist a generic solution that solves this, and any solution must be combined with host-based systems to detect the presence of stenographic software.

- ***Causative*** (*Insider Attacker*, *Controlled Environment*): By carefully choosing what to import, the training set can potentially be shaped in such a way that the algorithm later misclassifies documents containing classified content that the user wants to exfiltrate. Defenses against these attacks include algorithms that effectively sanitize the data by modifying the learning process to dynamically discount those data points in the training set that have a significant negative impact on the performance. A competing class of defense mechanisms recasts the problem as one of anomaly detection, e.g., does the model parameters of one user deviate dramatically from the model parameters of other users. Similarly, performing a causative attack to exfiltrate larger amounts of data would likely result in detectable anomalies in the set of imported documents for a controlled environment setting.

  Instance-based algorithms (e.g. K-NN) are not as susceptible to causative attacks because:

  - there is no training phase involved and
  - we can use a decision strategy in which any imported document with a similarity score greater than a threshold value will result in assigning the strictest label, e.g., "Classified", to the document.

- ***Model Data Leakage***: When the model is invoked by the trusted guard there is no known way of performing such an attack as the user does not have access to the confidence scores. Furthermore, with a controlled-environment the user is already authorized to access the documents in the training set, which renders such an attack meaningless.

4 Related work

The first work studying the use of machine learning to predict the security classification level of textual documents was done by Brown et al. [3]. Similar

studies later reproduced and expanded upon these results [1, 5, 12], while the concept of controlled environments was introduced by us in [7].

5 Conclusion

In this paper we have discussed the use of high assurance data guards for controlling the information flow between different security domains.

We observe that one are currently able to develop guards, whose assurance level in practical scenarios surpasses the assurance provided by the security labels which the guard relies upon. Thus, while the guard only releases data objects with a security label releasable by policy, there is typically less confidence that those security labels are correct. Thus, more effective content checkers that detect such mislabeled data objects would be of significant value.

We have introduced the concept of applying machine learning techniques to construct automated, data-driven content checkers. Our treatment of the topic extends beyond the theoretical considerations by including what we have, through extensive experiments, observed to be the best practices for data-driven content checkers, including which features to use and how to deploy the classifiers. We have focused especially on assessing the impact different deployment settings have on the security and performance of the end system. As such, we have presented the concept of controlled environments, where the audit trail or incoming information flow is used to construct per-user/session classifiers, and which yielded a significant improvement in performance. The proposed methods have also been analyzed with respect to causative, exploratory and data leakage attacks, and we noted that while they still remain vulnerable to causative attacks carried out by sophisticated insiders, they completely alleviate the threat of the inferred model leaking sensitive data from the training set.

While previous work only looked at building classifiers for complete documents [7], we have extended and adapted these methods to work shorter messages, which is a much more difficult case.

The performance we currently achieve is not high enough to warrant a fully automated deployment scheme. However, with an appropriate decision threshold the classifiers can be used to determine which documents that must be manually assessed. A meta-score (ITS) operating on the per-user long-term classifications trends can also be used to further reduce and manage the number of false alarms [8]. As future work one can also investigate the feasibility and usefulness of other forms of classifiers, e.g,. language and genre detection for increasing the trust in exported documents. We have conducted preliminary

experiments using an authorship verification model, built using the stylometric information embedded in the past chat messages of users, to detect instances in which an outgoing messages was not authored by the user in question. Combining the results from such different types of classifiers may potentially help improve accuracy.

References

[1] K. Alzhrani, E. M. Rudd, C. E. Chow, and T. E. Boult. Automated us diplomatic cables security classification: Topic model pruning vs. classification based on clusters. *arXiv preprint arXiv:1703.02248*, 2017.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010. ISSN 08856125. doi: 10.1007/s10994-010-5188-5.

[3] J. D. Brown and D. Charlebois. Security classification using automated learning (scale): Optimizing statistical natural language processing techniques to assign security labels to unstructured text. Technical report, DTIC Document, 2010.

[4] R. Haakseth, N. Nordbotten, Ø. Jonsson, and B. Kristiansen. A high assurance guard for use in service-oriented architectures. In *International Conference on Military Communications and Information Systems*, 2015.

[5] H. Hammer, K. W. Kongsgård, A. Bai, A. Yazidi, N. A. Nordbotten, and P. E. Engelstad. Automatic security classification by machine learning for cross-domain information exchange. In *Proc. IEEE Military Communications Conference*, volume 31, 2015.

[6] Z. Ji, Z. C. Lipton, and C. Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

[7] K. W. Kongsgård, N. A. Nordbotten, F. Mancini, and P. E. Engelstad. Data loss prevention based on text classification in controlled environments. In *Information Systems Security*, pages 131–150. Springer, 2016.

[8] K. W. Kongsgård, N. A. Nordbotten, F. Mancini, and P. E. Engelstad. An internal/insider threat score for data loss prevention and detection. In *International Workshop on Security And Privacy Analytics*. ACM, 2017.

[9] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[10] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[11] K. Wrona and N. Menz. Protection profile for the nato high assurance abac guard (haag), version 1.3. NCIA Technical Report TR-2012-SPW0084-18-13-4, NATO Communications and Information Agency (NCIA), 2013.

[12] K. Wrona, S. Oudkerk, A. Armando, S. Ranise, R. Traverso, L. Ferrari, and R. McEvoy. Assisted content-based labelling and classification of documents. In *Military Communications and Information Systems (ICMCIS), 2016 International Conference on*, pages 1–7. IEEE, 2016.

Paper VII

Data Loss Prevention for Cross-Domain Instant Messaging