VACCINE: Using Contextual Integrity For Data Leakage Detection

Yan Shvartzshnaider
New York University
yansh@nyu.edu

Zvonimir Pavlinovic
New York University
zvonimir@cs.nyu.edu

Ananth Balashankar
New York University
ananth@nyu.edu

Thomas Wies
New York University
wies@cs.nyu.edu

Lakshminarayanan Subramanian
New York University
lakshmi@nyu.edu

Helen Nissenbaum
Cornell Tech
helen.nissenbaum@cornell.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

ABSTRACT

Modern enterprises rely on Data Leakage Prevention (DLP) systems to enforce privacy policies that prevent unintentional flow of sensitive information to unauthorized entities. However, these systems operate based on rule sets that are limited to syntactic analysis and therefore completely ignore the semantic relationships between participants involved in the information exchanges. For similar reasons, these systems cannot enforce complex privacy policies that require temporal reasoning about events that have previously occurred.

To address these limitations, we advocate a new design methodology for DLP systems centered on the notion of Contextual Integrity (CI). We use the CI framework to abstract real-world communication exchanges into formally defined information flows where privacy policies describe sequences of admissible flows. CI allows us to decouple (1) the syntactic extraction of flows from information exchanges, and (2) the enforcement of privacy policies on these flows. We applied this approach to built VACCINE, a DLP auditing system for emails. VACCINE uses state-of-the-art techniques in natural language processing to extract flows from email text. It also provides a declarative language for describing privacy policies. These policies are automatically compiled to operational rules that the system uses for detecting data leakages. We evaluated VACCINE on the Enron email corpus and show that it improves over the state of the art both in terms of the expressivity of the policies that DLP systems can enforce as well as its precision in detecting data leakages.

CCS CONCEPTS

• **Security and privacy** *Usability in security and privacy*;

KEYWORDS

Contextual Integrity; Data Leakage Detection; DLP; Privacy

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313655

ACM Reference Format:
Yan Shvartzshnaider, Zvonimir Pavlinovic, Ananth Balashankar, Thomas Wies, Lakshminarayanan Subramanian, Helen Nissenbaum, and Prateek Mittal. 2019. VACCINE: Using Contextual Integrity For Data Leakage Detection. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313655

1 INTRODUCTION

Modern enterprises rely on Data Leakage Prevention (DLP) systems [12, 20, 27, 28, 39] to prevent the flow of sensitive information such as credit card and other business data to unauthorized entities. Rather than combating malicious behavior by end users, the goal of these systems is to prevent unintentional data leakage. This goal is accomplished by monitoring whether information flows (e.g. extracted from emails) adhere to a set of rules [35] that encode the organizational data handling policies, industry-specific regulations, and general privacy laws, and block or flag those flows that violate the rules. This monitoring process may be performed online as data is in motion, or offline to audit data that is in rest.

We will argue later that a key issue in the design of existing DLP systems is that they conflate the problem of extracting information flows from information exchanges and the problem of checking whether these flows are admissible. As a consequence, the tremendous progress made in natural language processing and machine learning techniques had little impact on modern DLP systems. Moreover, the rules that can be expressed in these systems fail to meet the demands imposed by the increasingly complex privacy policies and regulations that they must enforce. Specifically, they cannot account for contextual information such as consent, which involves reasoning about histories of information flows. The recent General Data Protection Regulation (GDPR) by the European Union is just one example of a privacy policy where such contextual notions are prevalent.

To address these limitations, we advocate a new design methodology for DLP systems centered on the notion of Contextual Integrity (CI) [30]. The key idea behind this methodology is to view real-world communication exchanges as sequences of information flows

as defined by the CI framework. A CI flow represents an atomic information exchange unit identified by the sender, recipient, attribute (i.e. data) being exchanged, and the subject associated with the attribute. CI flows carry the essential information being communicated in information exchanges while abstracting from the low-level details of these exchanges. Privacy policies are modeled as contextual norms that describe flow sequences admissible in a particular privacy context. DLP systems based on this view allow one to take advantage of advances in natural language processing techniques for extracting CI flows on one hand, and advances in formal methods for specifying and automatically enforcing privacy policies on the other hand.

The main technical contribution of this paper is VACCINE, a DLP auditing system for emails that follows this new CI-based design methodology. VACCINE provides users with a simple but expressive language for the declarative specification of complex privacy policies. In particular, it supports policies that require a DLP system to reason about the history of past information flows, which is not supported by other existing DLP systems in this general form. The policies are automatically compiled to sound-by-construction rules, implemented as SQL queries. These queries are used to check the admissibility of the extracted information flows.

VACCINE further leverages advanced techniques in natural language processing [23], such as syntactic dependency parsing [13], name entity recognizers [26], and question answering models [34], to efficiently extract contextual flows from complex email corpora. It combines rule-based and model-based techniques into a single combined model learned end-to-end in a generator-discriminator setting using distant supervision techniques.

We have evaluated VACCINE by applying it to the Enron email corpus [9, 36]. Compared to flow extraction used in existing DLP systems, we achieve improvements in F1 accuracy in detecting attributes by 4% and subject pairs by 32%. We have also used VACCINE to describe and enforce Enron privacy policies extracted from corporate privacy handbooks on the flows extracted from the email corpus. Our experiments demonstrate that VACCINE is effective at detecting violations of complex privacy policies in a real-world corporate setting.

2 DESIGN METHODOLOGY

In this section, we discuss the limits of commercial DLP systems and outline our new design methodology. For a detailed comparison with related work, see Section 5.

To check whether a particular information exchange such as sending an email is admissible, a DLP system has to solve two subproblems, as illustrated in Figure 1. First, it has to extract the (implicit) information flows from the information exchange. For instance, suppose Bob sends an email to Alice containing the phrase: *"You can find Mary's address below. Her credit card number is..."*. This email, thus, contains the implicit flow that Bob sends Mary's credit card data to Alice. Once this flow has been extracted, the second subproblem is then to check whether it is admissible according to the specified rules. For example, if a rule can disallow *any* flow of credit card data via email whatsoever, then Bob's email would be blocked or flagged as a data leakage. However, a more refined rule could include an additional relevant contextual information that, e.g., only disallowed flows with *Alice's* credit card data from Bob to Mary. In this case, the information flow would not be blocked.

Hence, the administrator of a DLP system has to provide two types of inputs to the system: (1) a specification of the information flows to be extracted from information exchanges, and (2) a specification of the rules that determine which of these extracted flows are admissible.

The issue with the design of current DLP systems is that, from the perspective of the administrator, these two types of information are conflated. The rules are specified directly in terms of the underlying information exchange format using combinations of regular expressions, templates, keywords, or text patterns. This severely limits the capabilities of these systems. In particular, the focus on regular expressions means that these systems cannot easily deploy more sophisticated flow extraction techniques based on natural language processing. For instance, the fact that the credit card data sent in Bob's email belongs to Mary cannot be captured by a regular expression because the flow extractor has to establish the identity of "Mary" and the possessive adjective "her" in the two consecutive sentences of the email, which involves a more complex analysis of the grammatical structure of these sentences.

It is therefore not surprising that current DLP systems only consider very basic notions of information flows and rules. While they can reason about senders and receivers of information flows like Bob and Alice, which can be easily extracted from the header of the email, there is no notion of a subject whose information is being transmitted, like Mary in our example. This, in turn, limits the expressivity of the rules that can be checked by these systems. For example, a more sophisticated DLP policy could demand that the flow from Bob to Alice is admissible as long as it is Bob's credit card number that Bob is sharing. Without the notion of a subject of the transferred information, such policies cannot be expressed.

A related limitation of existing systems is that their rules can only refer to a single information exchange (the email from Bob to Alice in our example). This means that they can only take into account very limited contextual information. For instance, a reasonable privacy policy could demand that the flow from Bob to Alice is only admissible if Mary has given explicit consent to sharing her information. Checking such policies requires reasoning about the history of information flows that happened in the past, which cannot be done in existing systems. However, such policies are common in privacy laws such as GDPR [1], which these systems are meant to enforce.

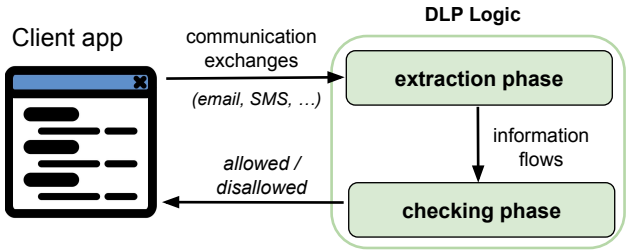

Figure 1: Illustration of an the enforcement mechanism that comprises two main phases: extraction and checking

2.1 Solving the Conflation Problem

Naturally, to solve the conflation problem, we need to separate the extraction and checking phases to make way for a modular DLP system architecture that is parameterized by the solutions to the two phases. A key technical challenge in achieving such separation is in defining a suitable information flow abstraction between the two phases. This abstraction will need to support seamless integration of customized solutions for each of the phases and needs to encompass relevant aspects of the communication exchanges and be amendable to different types of policies. Furthermore, the separation of extraction and checking phases in policy enforcement enables a natural division of labor in the process of implementing rules. For instance, on the one side, NLP experts can focus on devising techniques for extracting information flows from communication exchanges. On the other side, legal scholars and lawyers can focus on the task of checking that information flows obey the policies of interest, not worrying about the low-level details involving actual communication exchanges.

In this regard, the account of privacy [30] as contextual integrity offers a compelling answer to the question of how to tackle the aforementioned issues. It offers a) a suitable abstraction to decouple and interface between the extraction and checking mechanisms, and b) a common language for encoding legal policy and regulations such as FEPRA, HIPAA, and DLP rules into privacy rules. Moreover, CI overcomes the conflation problem by enabling the use of existing techniques tailored for the flow extraction and flow checking phases.

Next, we give a brief introduction to CI and also discuss how the theory and framework can be used to model the data leakage problem.

2.2 Contextual Integrity Overview

The theory of contextual integrity (CI) postulates that informational privacy is not all about secrecy (blocking information) [31] or control [44] but about the appropriateness of information flow within a particular context. Appropriateness of flow means flow that is compliant with contextual norms governing informational flows. To express an informational norm one must specify key parameters: *actors* (senders, recipients and subjects), *attributes* (the type of information at hand) and *transmission principles* (the constraints imposed on a particular information flow). Taken together, these components constitute context-relative informational norms. For instance, in the health context, the patient, acting in his capacity as both the sender and subject of an information flow, could be telling his doctor, the recipient, about his health issues, the attribute. The information flow would be constrained by the transmission principle of confidentiality, which restricts the onward flow of this information to other parties. Exchange the doctor for a friend, and the transmission principle might be reciprocity instead, since friends tend to expect to hear about each other's problems. A patient, by contrast, does not expect to hear about the health issues of his doctor. This is because the context of health and the context of friendship have different overarching goals: the doctor is there to promote the patient's health; friends are there to support each other. An informational norm is breached when an action or practice disrupts the actors, attributes, or transmission principles within a given information flow. Contextual integrity "is preserved when informational norms are respected and violated when informational norms are breached" [30].

It is worth emphasizing one of the most fundamental aspects of CI, namely that, in order to determine whether or not information flows respect or violate privacy expectations within a given context, one must address all three parameters: actors, information types, and transmission principles. Omitting any one of them may lead to an inconclusive or ambiguous description. Accordingly, any formal rendering of information flows, for the purpose of assessing their appropriateness, needs to include independent variables for these parameters. Defining privacy [30] in terms of CI is now a well-established approach in the privacy and computer science research communities. It is used to describe contextual informational norms, to detect infractions of these norms, and in approaches to accountability and enforcement [5, 17].

2.3 Data Leakage Detection using CI

In its core, the theory of CI offers a way to reason about the appropriateness of an information exchange. An *information flow* is the atomic unit of an information (*attribute*) exchange between *actors*. Information flow is formally defined by a tuple:

```
[sender, recipient, subject, attribute]
```

Actors (sender, recipient, subject) take on different roles/objectives that define the capacity in which they function in a given context such as doctors, teachers, or network middleware. Appropriateness of information flow, within a particular context, is defined using contextual informational *norms*. A norm consists of:

```
[Sender, Recipient, Subject, Attribute,
                    Transmission principle]
```

Here `Sender`, `Recipient`, and `Subject` refer to roles and `Attribute` to the type of information exchange under a specified `Transmission principle` condition. For instance, in the educational setting, where information exchange between actors is regulated by the FERPA policies, *a teacher* (sender role) would be allowed to share *grades* (attribute) of a *student* (subject role) to her *parents* (recipient role) if the student has given *explicit written permission* (transmission principle) [42]. If the student has not given such a permission, flows from the teacher to a student's parents involving her grades must not be allowed, i.e., such information exchanges are not appropriate. Thus, it is important to realize that contextual norms specify the appropriateness of a flow subject to conditions that may depend on the history of previous information flows.

We can use the above CI model to define a data leakage. In CI terms, a data leakage occurs within a particular context when an information exchange (flow sequence) is not deemed appropriate by any of the specified contextual norms. We use the above as a system abstraction for building real-world DLP systems that prevent data leakage by only allowing sequences that are consistent with the norms. More formally, we denote the flow sequences that the system deems admissible with T_s and the contextual informational norms of a privacy context as a set of flow sequences, T_n. A DLP system is designed to *correctly* enforce a privacy context by guaranteeing that $T_s \subseteq T_n$. i.e., the system exhibits flow sequences allowed by the given norms.

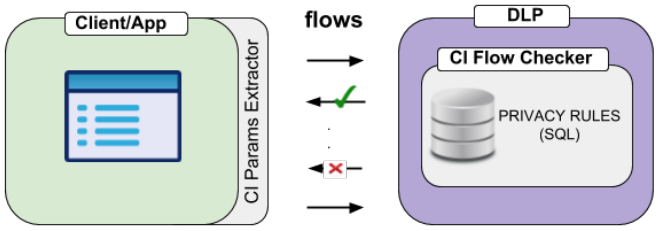

Figure 2: *VACCINE:* Information exchanges are processed by the CI Parameters Extractor that forms corresponding CI flows that are then fed into CI Flow Checker to check for violations with respect to existing privacy norms

3 VACCINE

We next describe our Verifiable and ACtionable Contextual Integrity Norms Engine (VACCINE), a DLP system for email communication based on CI. As illustrated by Figure 2, VACCINE maps information exchanges at the application level to the corresponding CI flows and checks them against the specified CI norms. The information flow abstraction is used to communicate the required contextual information to the checking mechanism.

Specifying norms. Norms can originate from a combination of external sources such as privacy handbooks and context specifications. Other external sources may also be used as a base for CI norms in a specific domain, such as legal and policy documents, professional codes of conduct, findings of empirical social scientists, privacy advocates, ethicists, stipulations of sponsoring organizations, etc. To express contextual norms, VACCINE provides a logic for specifying what flows are admissible based on past events. For example, a norm might state "Allow information flows from Alice to Bob with Mary's credit card information, *provided Mary previously gave Alice permission to do so*".

Enforcing norms. For this task, VACCINE operationalizes norms into a set of actionable privacy rules by automatically compiling norms into SQL queries which then check if a given flow is admissible taking into account the history of previously observed flows. To enforce the norms, VACCINE first extracts the respective flows from the email text with the help of the CI Parameter Extractor and then uses the CI Flow Checker to see whether the extracted flows are admissible by running the respective SQL queries.

VACCINE's approach offers intrinsic benefits to the design of DLP systems. First, it achieves *modularity in design* as a result of decoupling the extraction and checking phases in policy enforcement. Second, it increases the *expressiveness of privacy rules* due to its support for CI based norm enforcement. Finally, it supports *obfuscation of sensitive information* through the use of the CI framework that only captures the type of information being conveyed, not the actual data.

In the following, we describe the two key components of the system in more detail. We also briefly describe the temporal logic that we use for expressing the CI norms that are enforced by the CI Flow Checker.

3.1 CI Flow Extractor

The CI flows comprise the CI parameters extracted by the CI parameter extractor. As depicted in Figure 2, the parameter extractor detects the relevant CI parameters associated with each information exchange such as actors (sender, recipient, subject) and the type of information (attribute) and map them onto CI flows. The extractor retrieves the sender and the intended recipients from the underlying protocol's metadata. This is the straightforward bit compared to the task of identifying attributes and the corresponding subjects, which requires a more sophisticated approach. For example, if the policy states that "John's financial information" should not be transferred without his permission, the CI extractor needs to identify all patterns of text that correspond to financial information like, *bank accounts, SSN*, etc. Similarly, for each attribute type mentioned in the privacy policy text, the CI extractor needs to identify the CI subject, in this case *John*. The resulting CI flow will comprise of the extracted parameters:

```
[sender,
        financial information (attribute),
                John (subject),
                                recipient]
```

To tackle the attribute-subject task, we combine a suite of natural language models using recent advances in distant supervision techniques to identify the mentioned attributes and link them to the appropriate subjects. We assume that the relevant CI parameters are retrieved from the policy either manually or with the assistance of an automatic CI parameter detection algorithm [37].

Extracting attributes. The CI extractor needs to identify the instances of the abstract information types included in the rules (policy). For example, if the rule applies to financial information, the extractor needs to detect all possible related instances in the text such as SSN, bank information, credit card number and so forth. This many-to-one mapping makes it difficult to use a single model to detect all instances of a given attribute. To tackle this issue, we adopt the Snorkel [32] framework to generate a large amount of training data using a small number of annotators with weak supervision to increase accuracy. This approach helps to bootstrap our model to annotate CI parameters in emails using a small number of human curated annotators. As Figure 3 shows, we use the labels outputs of multiple models to train a generative annotator choice model. Namely, we import regular expressions corresponding to each attribute[1]. Secondly, we include name entity recognizers which use a dictionary mapping to known types of nouns to identify numeric and non-numeric attributes [23]. Lastly, we use the syntactic structure to identify the nouns associated with subjects. The outputs from each of the models is used to train a simple logistic classifier on these labels. The resulting unsupervised annotator choice model performs better than the majority vote of all annotators by modeling the agreement between various annotators. The high precision, low coverage probabilistic labels provided by the generative model is trained to discriminate by a fully connected feedforward neural

[1]There can be more than one pattern for attribute like "phone number". See http://regexlib.com/Search.aspx?k=phone

network against gold labels annotated by humans on a small set of information flows.

Extracting subjects. For this task, similarly to attribute detection, we use the output of a three model mixture to train a generative annotator choice model for subjects. The annotator models set comprise: 1) a heuristic model that matches the entity closest in syntactical proximity to the attribute using a name entity recognizer (NER) 2) a model to detect matching subject-object using syntactic dependency parse trees, 3) a state-of-the-art conversational question answering model (BiDAF) [34] which encodes the email text and a question whose answer is the desired subject of an attribute into an embedding to perform semantic dimensionality reduction. For example, if the attribute detected is a phone number, the question posed to the QA model is "Who should we call?".

3.2 CI Logic

As explained in Section 2.2, a system execution in VACCINE can be abstractly seen as a timed sequence of CI flows. That is, a set of contextual informational norms describes those flow sequences that are allowed in a given privacy context. We next introduce a logic, `TFlow`, for expressing privacy norms that can be automatically converted into actionable rules used to check whether given information flows obey the specified norms.

Syntax. `TFlow` allows the declarative specification of CI flow sequences where the admissibility of a flow in a sequence can depend on whether other flows have or have not been observed. More precisely, `TFlow` is a variant of metric linear temporal logic [3, 15, 22, 40], which has been used to formalize CI-based privacy frameworks in other settings. We here adapt this logic to the specific setting of DLP and identify several important restrictions that enable the efficient checking of these norms in a DLP system. One important restriction is that we allow only temporal operators that can reason about flows that have happened in the past, which makes the logic suitable for both offline and online monitoring. The syntax of the logic is as follows:

$$\begin{aligned}\phi \in \texttt{TFlow} ::= {} & \phi_1 \textbf{ and } \phi_2 \mid \phi_1 \textbf{ or } \phi_2 \mid \textbf{not } \phi \mid \textbf{exists } x.\ \phi \\ & \mid \phi_1 \textbf{ since } \phi_2 \mid \textbf{flow}(x_1, x_2, x_3, x_4) \textbf{ where } P(\vec{x})\end{aligned}$$

Atomic formulas $\textbf{flow}(x_1, x_2, x_3, x_4)$ **where** $P(\vec{x})$ are used to describe flows from a sender x_1 to a receiver x_2 concerning attribute x_3 of subject x_4 such that the formula $P(\vec{x})$ holds. Here P is a Boolean combination of basic predicates, such as equality between variables $x \in Var$ and constants $c \in C$. In the case that $P(\vec{x})$ is simply *true*, we omit the **where** clause for convenience. Constants include numbers, strings, and other common data structures encoding agents and attributes of the target CI context. Temporal formulas of the form ϕ_1 **since** ϕ_2 are used to express that at some point in the past the formula ϕ_2 held after which all later time steps satisfied the formula ϕ_1. The semantics of existential quantification (**exists** $x.\phi$), logical conjunction (**and**), disjunction (**or**), and negation (**not**) are as expected.

Contextual norms are typically expressed as `TFlow` formulas of the form

$$\textbf{flow}(snd, rcp, att, sub) \textbf{ and } \phi(snd, rcp, att, sub)$$

where *snd*, *rcp*, *att*, and *sub* refer to the parameters (agents and attribute) of the flow being checked for admissibility, and ϕ is a formula specifying the transmission principle of the norm.

Examples. As a first example, let us consider a simple generic norm $norm_g$ stating that everybody can send their own data to themselves. This norm can be expressed in `TFlow` as

$$\begin{aligned}norm_g(snd, rcp, att, sub) \stackrel{\text{def}}{\equiv} {} & \textbf{flow}(snd, rcp, att, sub) \textbf{ where} \\ & snd = sub \textbf{ and } snd = rcp.\end{aligned}$$

As another example, suppose we want to express the $norm_p$ stating that an agent *snd* can share an attribute of an agent *sub* if *sub* has previously given *snd* an explicit permission to do so, and *sub* has not revoked the permission since then. This norm can be formulated in `TFlow` as

$$\begin{aligned}norm_p(snd, rcp, att, sub) \stackrel{\text{def}}{\equiv} {} & \textbf{flow}(snd, rcp, att, sub) \textbf{ and} \\ & ((\textbf{not flow}(sub, snd, \langle -, permission, att\rangle, sub)) \textbf{ since} \\ & \textbf{flow}(sub, snd, \langle +, permission, att\rangle, sub)).\end{aligned}$$

Here, we use the term $\langle +, permission, att\rangle$ to indicate that permission of sharing *att* has been granted, and similarly $\langle -, permission, att\rangle$ to indicate that this permission has been revoked.

We point out that existing DLP systems cannot handle the second norm because it involves reasoning about flows that occurred in the past. It is also instructive to point out that the above formulation of the norms does not depend or involve lower-level details of communication exchanges, thereby achieving the level of abstraction needed for separating the checking phase from the extraction phase.

Formal semantics. The formal semantics of `TFlow` formulas can be defined in terms of timed sequences of flows. In this work, we take an equivalent approach where a flow sequence is modeled as a table of flows where each row in the table contains the arguments of one flow in the sequence together with its timestamp indicating when the associated flow occurred. Hence, the flow sequence can be reconstructed from the flow table using this timestamp information.

The satisfiability of a `TFlow` formula ϕ is defined by a satisfiability relation $T, \Gamma, i \models \phi$ which states that ϕ holds at time point i in the flow trace given by flow table T and subject to the variable assignment Γ, which maps the free variables of ϕ to values. For instance, here are the formal definitions of the satisfaction relation for the two most important constructs in the logic:

$$\begin{aligned}T, \Gamma, i \models {} & \textbf{flow}(x, y, z, w) \textbf{ where } P(\vec{x}) \qquad \text{iff } f \in T(i) \text{ and} \\ & \Gamma[x \mapsto f(1), y \mapsto f(2), z \mapsto f(3), w \mapsto f(4)](\vec{x}) \in I(P)\end{aligned}$$

$$\begin{aligned}T, \Gamma, i \models {} & \phi_1 \textbf{ since } \phi_2 \qquad \text{iff } T, \Gamma, j \models \phi_2 \text{ or} \\ & i > 0 \text{ and } T, \Gamma, i \models \phi_1 \text{ and } T, \Gamma, i-1 \models \phi_1 \textbf{ since } \phi_2\end{aligned}$$

In the above, we use $T(i)$ to denote the flows in T with the timestamp i. Similar notation is used for accessing constituent elements of a flow f. Lastly, we assume that the function I provides the interpretation for basic predicates expressing constraints on variables and constants. We omit the full details of the semantics as it is standard [15, 22]. We note that the above semantics directly supports extensions of `TFlow` to metric temporal logics where formulas can explicitly refer to time points.

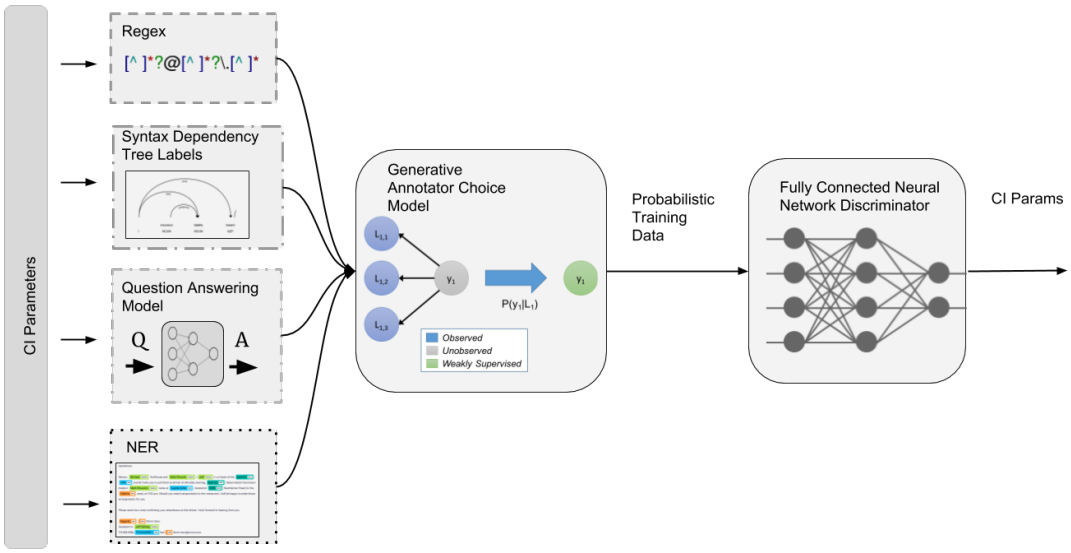

Figure 3: CI Parameters Extractor. We combine outputs from multiple models to train a generative annotator choice model, which we subsequently use to train a fully-connected neural network discriminator for extracting relevant CI parameters.

3.3 CI Flow Checker

We have implemented our flow checker on top of an SQL database. The database maintains a table `FH` of all the flows that have been accepted so far. Checking whether a new flow f obeys the norms expressed in `TFlow` amounts to checking if `FH` extended with f satisfies any of the norms. Each norm $\phi(\vec{x})$ is automatically compiled to an SQL query that produces a table with one column per free variable in $\vec{x}$ as well as a timestamp column. This table contains one row per timestamp i and valuation Γ for $\vec{x}$ such that $\mathtt{FH}, \Gamma, i \models \phi$ holds. Thus, if the query for any of the norms contains an entry for the current timestamp i, then the new flow is accepted and permanently added to `FH`. Otherwise, it is rejected.

The algorithm for generating the SQL queries is defined recursively on the structure of formulas and closely follows from the semantics of `TFlow`. However, a naive encoding into SQL yields queries that do not scale well, in particular if we must deal with nested **since** formulas and unconstrained negation. We therefore discuss several restrictions and optimizations to a naive encoding that considerably improves the performance.

Syntactic restrictions. In order to support the automatic compilation of `TFlow` formulas into efficient SQL queries, we impose some restrictions on `TFlow`. First, the free variables in the subformulas of disjunctions have to be identical. Further, all variable must occur as a parameter in some **flow** predicate. Finally, we require that every negative formula **not** ϕ_1 that occurs in a positive context is guarded by a positive formula ϕ_2. A formula is positive if it contains no unguarded negative formulas and a formula occurs in a positive context if it occurs under an even number of **not** operators. A negative formula **not** ϕ_1 is guarded by ϕ_2 if these formulas occur either as (**not** ϕ_1) **and** ϕ_2 or as (**not** ϕ_1) **since** ϕ_2 where all the free variables of ϕ_1 also occur free in ϕ_2.

For instance, in the norm $norm_p$ given in § 3.2, the negative formula (**not** **flow**$(sub, snd, \langle -, permission, att\rangle, sub)$) is guarded by the positive formula **flow**$(sub, snd, \langle +, permission, att\rangle, sub)$ below the **since** operator. This restriction of the logic to a *guarded fragment* is particularly important because it ensures that negative subformulas can only be used to prune the result tables obtained from positive subformulas. This restriction enables us to cache the evaluation of subformulas in dedicated tables which can then be updated incrementally using simple queries as new flows are added to the history table `FH`. We describe this process in more detail below. Without this restriction, the evaluation of temporal formulas involving the **since** operator cannot be done efficiently (i.e. without reevaluating the entire history of flows each time the norm is checked). For otherwise, to enable caching we would have to know in advance the entire universe of values for agents and attributes that may appear in a future flow. We found that these syntactic restriction did not affect the expressivity of the logic in practice.

Compiling norms to efficient SQL. Our optimized compilation process associates a unique SQL table T_ϕ with every subformula of a norm that needs to be checked. These tables cache the satisfaction relation for ϕ with respect to the current history table `FH`. Thus, when we check a norm ϕ, we can simply query the associated cache table T_ϕ directly. These cache tables are updated whenever a new flow is added to the history table and time advances to a new timestamp. The updates are computed recursively according to the structure of the subformulas.

For instance, suppose ϕ is of the form (ϕ_1 **since** ϕ_2) and we extend the history table `FH` with a new incoming flow f at time i. Then we first update the caches for the subformulas ϕ_1 and ϕ_2 recursively to account for the update. To update the cache for ϕ itself, note that according to the semantics of **since**, `FH` extended with f satisfies ϕ at timestamp i iff either ϕ_2 is satisfied at i or ϕ_1 is satisfied at i and ϕ

already holds at the previous timestamp i'. In effect, this means that the table for ϕ can be updated as $T_\phi \triangleq T_\phi \cup T_{\phi_2}(i) \cup (\pi_t(T_\phi(i')) \circ T_{\phi_1}(i))$. Here, $\circ$ stands for inner join (on the common columns), and π_t projects out the timestamp column in a given table. The cache table for an atomic formula $\phi \triangleq \textbf{flow}(x, y, z, w)$ **where** $P(\vec{x})$ is updated with the results of the following generic SQL query:

```
SELECT snd AS x, rcp AS y, att AS z, sub AS w
FROM FH WHERE P(x⃗)
```

Here, we assume the case where $\vec{x} = (x, y, z, w)$. If P has free variables other than x, y, z, w, these variables are additionally added as columns in the `SELECT` clause. Those columns are then populated with all possible values in `FH` using appropriate subqueries. The syntactic restrictions on free variables in norm formulas ensure that this encoding is both sound and complete.

4 EVALUATION

In this Section, we present results of evaluating the key components of VACCINE in a corporate setting. We first evaluated the CI parameter extractors that form the basis of the combined CI extractor individually on the emails from the Jeb Bush Presidential campaign. We then present an end-to-end case study of data leakage protection using VACCINE on the corpus of emails associated with the Enron corporation [9, 36]. We detail the accuracy of VACCINE parameter extraction techniques on the Enron email dataset, the extraction of contextual norms (expressed in `TFlow`) for the Enron corporate privacy context, performance numbers on evaluating the norms using the VACCINE SQL-based flow checker, and the benefits of using VACCINE for real-world data leakage protection.

4.1 Extraction Microbenchmark

We compared the efficiency of the CI flow extraction against manually annotated emails from the Jeb Bush Presidential campaign [2]. We manually annotated 436 attributes and 72 subject parameters. Table 1 summarizes the performance of the components used in the Snorkel based CI parameter extraction. Some of the failures of the model can be attributed to genuinely ambiguous natural language in the email. A typical example is the following sentence: *"Can you tell me if you have plans to amend the drivers license laws, so that as I, Florida homeowner from the UK, can have a drivers license for 6 years as I used to have"*. Here, the manually annotated subject was *"I"*, but the model's output was *"Florida"*. Similarly, for attribute detection, many of the failures are due to the jargons used within an organization which can be confused for affiliations. This makes the case for continuous improvement of the model through active learning once a system like VACCINE is deployed to keep up with the evolving language patterns.

4.2 Enron Case Study

We have evaluated VACCINE's DLP system performance for offline auditing of email communication. For this experiment, we used the Enron corporation email corpus [9, 36]. Specifically, we aimed to answer the following questions:

(1) How effective is the VACCINE framework in preventing potentially unauthorized flows in a real-world emulated context?

Model Type	Parameter Type	F1 Accuracy (%)
Regex	Attribute	71.56
NER	Subject	56.94
QA	Subject	61.11
QA+Query Rewrite	Subject	63.88

Table 1: Benchmark of CI parameter extraction from Jeb Bush's emails

(2) How efficiently can CI flows be checked against the privacy rules automatically generated from `TFlow` norm descriptions?

We chose the Enron corporation for our enterprise context since both its code of ethics [43] as well its email dataset are publicly available [9, 36]. From Enron's code of ethics we extracted CI parameters for an enterprise context. We identified 145 agents with 12 company roles, such as director, manager, and regular employee. We assigned roles to employees, identified by their email addresses, using publicly available data sources [18, 41]. We also identified 13 attributes, such as social security number, visa number, address, etc. Our focus was on emails containing the exchange of sensitive information such as passwords, bank account information, phone numbers, etc. We thus identified 73 emails involving exchange of such information and extracted 87 corresponding CI flows. We used these flows to evaluate the effectiveness and efficiency of VACCINE. All of the experiments were carried out on an `Intel(R) Xeon(R)` machine with eight 3.6GHz cores.

4.2.1 Norm Extraction. We identified 40 Enron contextual norms that focus on access and disclosure of personal identifiable information (PII) in a corporate setting. The norm creation process was conducted over the available documentation of Enron's code of ethics and its organizational structure. Due to the lack of a publicly available (Enron) enterprise privacy handbook that specifies precise norms, we created norms to enforce a simulated *ethical wall* on email communication between Enron employees. An ethical wall [29] refers to establishing an environment that will prevent information exchange between specific departments within the organization to avoid any conflicts of interest. Our norm extraction process focused on each type of PII and how agents in the various enterprise roles can exchange the corresponding attributes. A typical example of a norm is: *Enron employees can send their social security number to administrative managers*. This norm in our policy language `TFlow` is expressed as

$$norm_{SSN}(s, r, a, b) \stackrel{\text{def}}{\equiv} \textbf{flow}(s, r, a, b) \textbf{ where } s = b \wedge a = \text{“}SSN\text{”} \textbf{ and } in\_role(s, \text{“}employee\text{”}) \textbf{ and } in\_role(r, \text{“}manager\text{”}).$$

Here, the *in_role* predicate is defined using the **since** construct in a similar fashion to the predicate of having a permission in Section 3.2. In contrast, employees are allowed to send their less sensitive information, such as phone number and addresses to anyone without restrictions. This norm is captured in `TFlow` as

$$norm_{PII}(s, r, a, b) \stackrel{\text{def}}{\equiv} \textbf{flow}(s, r, a, b) \textbf{ where } s = b \wedge (a = \text{“}phone\_number\text{”} \vee a = \text{“}address\text{”}).$$

Also, the distribution of information belonging to other Enron employees can be done if the subject employee gave a permission for the disclosure of such information. This norm specified in `TFlow` is essentially the norm $norm_g$ introduced in Section 3.2.

4.2.2 Identifying Contextual Information. To evaluate the extraction of CI parameters, we manually annotated our corpus of selected emails (73) with attributes such as phone number and conference call codes and CI subjects corresponding to them. As a baseline accuracy for extracting relevant attributes without taking any contextual information into account, we used regular expression based models (regex) for detecting attributes. These models are similar to the ones currently available in tools like Google DLP and Microsoft Exchange.

Results Table 2 shows that by aggregating multiple regular expressions using a distant supervised framework called Snorkel [32], gives an improvement of 4% points in F1 accuracy. For subjects detection, it is close to 1% F1 accuracy. Combining results from the QA model using the "Who should we call?" question with other NER based models increases the accuracy by almost 4%. Narrowing context by posing a question that includes an attribute, e.g., "Whose phone number is 123-456-7890?", leads to increased accuracy by a further 9%.

Model Type	Parameter Type	F1 Accuracy (%)
Regex	Attribute	90.20
Snorkel	Attribute	**94.80**
NER	Subject	48.35
QA	Subject	67.12
Snorkel	Subject	71.83
QA+Query Rewrite	Subject	**80.82**

Table 2: Accuracy of CI parameter extraction from Enron emails

4.2.3 Leakage Detection Performance. The purpose of our next experiment was to measure the general performance of checking flows against the extracted norms. First, we automatically compiled the extracted Enron norms to SQL queries that we then evaluated against approximately 6700 Enron flows, consisting of our manually extracted flows and initial flows used to set up the organizational structure of the company. For instance, the fact that agent a is an employee is modeled as the flow

$$(enron\_inc, enron\_inc, \langle +, in\_role, employee\rangle, a).$$

Intuitively, this flow encodes that a has been previously given a role of a regular employee in the Enron corporation. We also use the initial flows to incorporate reasonable assumptions about disclosure privileges for PII information, say, that within each department (i.e., role), employees can disclose each other's phone numbers.

Our implementation of the SQL flow checking engine follows the high-level algorithm introduced in Section 3.3 except that we cache only since formulas in auxiliary tables. We only maintain the rows with the latest timestamp in cache tables as they are the only once needed for proper updating in the future. We use Sqlite3 as our relational database. Lastly, we compressed 40 Enron norms into 12 succinct ones as many of them shared the same structure except that they differ slightly in the specific **where** conditions. For instance, consider a norm stating that managers can disclose employee phone numbers to CEOs and a norm stating that managers can disclose employee phone numbers to directors. We compress these two norms into a single one where managers can disclose employee phone numbers to either CEOs or directors.

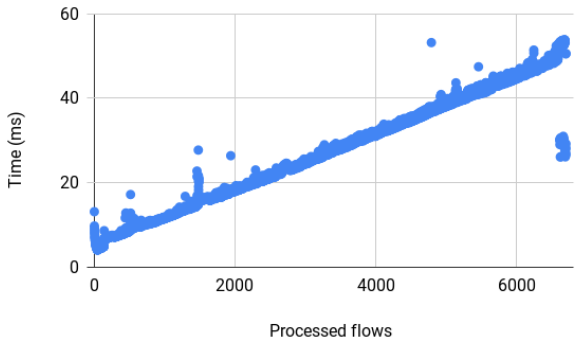

Figure 4: Flow checking performance (in miliseconds) through time

Results The performance times are reported in Table 3. We provide measurements for flow checking in general and its breakdown subject to accepted and blocked flows. We also provide information on time spent performing cache updates while checking flows. As it can be seen, average time spent analyzing a flow is reasonable. We point out that rejecting a flow takes more time than accepting it. This is because we implement a form of short-circuit evaluation of the norms where a flow is accepted as soon as we encounter a norm that is satisfied. The remaining norms then no longer need to be evaluated. We note that the initial flows for setting up the organizational structure are always accepted, accounting for the high number of accepted flows. We also measured how flow checking times vary depending on how the history flow table grows. The results are shown in Figure 4. Flow checking times grow linearly with the size of the table, i.e., the number of accepted flows in the past. The outliers in the graph mostly correspond to the checking of flows that occur at the end of the experiment where we check the flows extracted from the Enron emails containing sensitive information. On the other hand, initial flows are always accepted by exactly the same norm, so the performance of analyzing such flows is more stable. The reported numbers can be improved by employing more sophisticated algorithms used in work on runtime policy monitoring [15, 22].

	total	avg	med	min	max	sd
flows (6706)	184	0.027	0.027	0.004	0.054	0.012
accepted (6658)	181	0.027	0.026	0.004	0.053	0.012
blocked (48)	2.54	0.052	0.052	0.051	0.054	0.001
cache upd.	95	0.007	0.0003	0.0002	0.0001	0.008

Table 3: Performance times (in seconds) for SQL checking of flows against the Enron norms

4.2.4 Detecting Information Leakages. The flows rejected in the previous experiment constitute data leakages subject to the identified contextual Enron norms. We note that the flow checking experiment does not suffer from false positives or false negatives. This is

due to the fact that the flows used in the experiment were manually extracted and SQL queries implementing the norms are sound by construction (as described in Section 3.3).

In our next experiment, we have analyzed the nature of emails from which the rejected flows were extracted.

Results Arguably, the most interesting case of rejected flows involves visa information. VACCINE rejected a flow between two Enron employees where the sender was sharing visa information about a third party subject. More precisely, the sender was sharing the type of the visa in question, its expiration date, and the information that the subject was granted multiple entries to the United States. To the best of our understanding, the subject had not previously granted the sender with a permission to share such confidential information.

The vast majority (44) of rejected flows involves distribution of passcodes/passwords and phone numbers for company conference calls. Our implementation of VACCINE rejected such flows as we did not extract or create any contextual norms governing the distribution of conference call metadata. In effect, VACCINE conservatively treats such flows as data leakages. VACCINE also prevented a flow where a third party resumé was shared between two Enron employees. Our current norms only allow agents to share their own resumes. Lastly, our implementation also blocked a flow between a third party and one of the higher-ranked Enron employees where the flow subjects were Enron bank accounts. Although careful inspection of the corresponding email did not show any actual leakage of sensitive company information, our norms were deliberately designed to contain the communication exchanges involving confidential company data within Enron.

These experiments show that VACCINE provides an effective tool for data leakage prevention, modulo an effective flow extractor.

4.2.5 Complexity of Email Communication Exchanges. In our last experiment, we measured how many CI flows are typically generated in an email conversation. We note that a single email can generate multiple flows between different participants. This is because an email often quotes other emails that have been sent earlier in the same conversation. In such cases, the new email also generates *transitive flows* involving subjects and attributes from the quoted emails to the new recipients.

We grouped the emails in the Enron data set into conversations. We then defined an email conversation as a set of emails with the same email subject (modulo reply and forwarding prefixes, etc.). We extracted from each email a set of quadruples (one per recipient) consisting of the sender email address, a recipient email address, the time the email was sent, and the recipient type which can be `TO`, `CC`, and `BCC`, omitting the email content and other metadata. Here is an example of such a tuple:

```
(tracy.geaccone@enron.com,
 sarah.taylor@enron.com,
                2001-11-27 12:48:05,TO)
```

We identified every unique email address with a unique participant. For emails sent to mailing lists, we simply simulated broadcasting the email to all of the active conversation participants.

Out of the whole Enron email data set, we randomly chose 2,000 email conversations involving approximately 80,000 emails and 11,500 participants in total, as shown in Table 4 (first and second row). We used the email time and recipient type fields to recreate email threads for each conversation. There are two email threads per conversation on average, as shown in Table 4 (third row). Due to some missing information in the data set, we observed cases where participants sent emails in the middle of the conversation although they never received any emails in the conversation up to that point. In our analysis, such emails start a new conversation thread, which explains multiple threads per conversation. We used the extracted email threads to generate CI information flows. To model the worst case scenario in terms of the information flow complexity, whenever an email is sent by some sender in a conversation, we assume the sender creates new content and also transitively sends all of the emails in the conversation up to that point. Also, a CI flow has a subject and attribute. For the purposes of this experiment, the actual subject and attribute values are not important. For transitive flows, however, the subject is the sender of the corresponding email that is being quoted.

	Total	Min	Avg	Max
Emails	78,386	1	39	15,555
Participants	11,609	1	17	2,428
Threads	4,082	1	2	196
Flows	218,070	1	109	85,280

Table 4: Statistics for each of the 2000 Enron email conversations

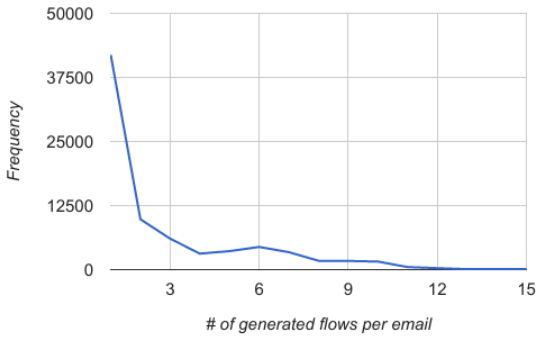

Figure 5: Email frequency distribution based on the number of generated flows per email

Results Given the above dataset, we measure how many flows an email yields in practice. As it can be already inferred from Table 4, a single email yields 2.78 flows on average. Moreover, Figure 5 shows the frequency distribution of emails depending on the number of flows they generate. It can be seen that the flow-per-email ratio is in general low, reaching 15 at maximum. This suggests that the load on a DLP (auditing) system created using VACCINE will be tolerable in practice. This is further evidenced by Figure 6 showing a linear increase in the number of generated flows subject to a total number of thread emails.

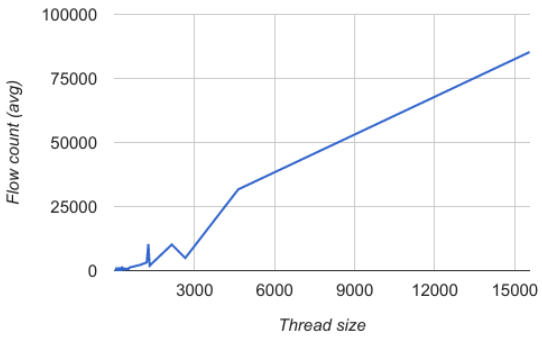

Figure 6: Average number of generated flows per size of the conversation thread

5 RELATED WORK

We already discussed how VACCINE compares to existing commercial DLP solutions in Section 2. The research community has explored sophisticated email DLP methods mostly relying on a combination of machine learning and related classification techniques. These approaches roughly fall under content- and behavior-driven methodologies [35]. Content-based approaches [10, 11, 21, 25] determine the sensitivity of a message based on textual content analysis, either through keyword, regex pattern matching, or machine learning techniques. Behavior-based approaches [24, 38, 45] analyze the organizational structure, common roles and duties, as well as past interaction to identify "normal" behavior patterns and use them to detect unlikely information flows between a pair of sender and recipient. Our work is largely orthogonal to these approaches. While prior solutions explored advancing techniques in identifying insensitive data and malicious behavior patterns, VACCINE instead focuses on providing a robust way to express, administer, and enforce complex DLP policies.

Early work by Armour et.al., looked at designing a privacy compliance engine for emails [4, 7] to prevent privacy violations in large enterprise settings. Similar to our system, their engine comprises an information extraction and violation detection module. The information extraction module relied on an enterprise domain ontology, part-of-speech and semantic tagging to find relevant information in the email text. While the work positions itself as a privacy compliance tool, it does not go beyond facilitating role-based access control to information. Namely, the implicit assumption here is that roles and ownership determine the right to access. In particular, it does not support complex policies involving e.g. notions of consent that require reasoning about the history of observed emails. Furthermore, VACCINE takes advantage of the advances in NLP techniques since this pioneering work. We are not aware of other work that took the ideas in [4, 7] and developed them beyond the preliminary proposal explored there.

Related Work using CI. Defining privacy [30] in terms of CI is now a well-established approach in the privacy and computer science research communities. It is used to describe contextual informational norms, to detect infractions of these norms, and in approaches to accountability and enforcement [5, 17]. Much of the prior work in this space has focused on sophisticated formal logics for describing CI norms that involve complex temporal properties of sequences of information flows such as those involved in HIPAA [6, 14, 16, 19]. We build on these earlier works and adapt them to the specific needs of the DLP setting. Our observation is that in this setting, a simpler logical framework is sufficient to express the relevant norms. With these restrictions, norms can be automatically compiled into SQL queries. The encoding into SQL enables the use of our system in applications where both robustness and efficiency of the DLP engine are important factors.

Compliance-based Approaches. Numerous approaches have been invented to map system constraints to privacy policies that are generated external to these systems. Two examples, are due to Sen, et al. [33] and Breaux and Anton [8]. Sen, et al. have developed LEGALEASE, a language with precise semantics that enables enterprises to express privacy policies against which practices can be checked for compliance. With similar goals, Breaux and Anton have proposed a methodology for extracting implementable software requirements from privacy and security regulatory requirements expressed in legal language that is sometimes vague and ambiguous. The intention in both systems is similar to our own in that it seeks to translate privacy requirements expressed in natural language (assuming this covers legal language) into formally expressed rules that allow for compliance checking. The threat scenarios, however, are quite different.

6 CONCLUSIONS

We presented a new design methodology for building DLP systems based on the theory of Contextual Integrity that offers more precise and modular detection of information flow violations compared to traditional DLP systems.

Using our methodology, we built VACCINE, a system for auditing email communication corpora. VACCINE supports a powerful declarative language for specifying expressive privacy policies that are automatically compiled to sound-by-construction actionable rules, implemented as SQL queries. Furthermore, VACCINE leverages advanced NLP techniques to efficiently extract relevant information omitted by the existing DLP systems.

In our evaluation, we used VACCINE to audit the Enron emails corpus and achieved significant improvement compared to the methods used by state-of-the-art systems.

7 ACKNOWLEDGEMENTS

We thank Paula Kift and Schrasing Tong for their help in the initial stage of this work. This work is supported by the following National Science Foundation (NSF) under grants CCF-1350574, CNS-1514422, CNS-1801501, CNS-1704527, SES-1537324, SES-1650589, the National Security Agency grant H98230-18-D-006 and the Cisco CG-653005 Research Award. We are also grateful to the NSF I-CORPS 1650769 grant that allowed us to discuss the data leakage problem with large number of companies and ultimately helped us with articulating the VACCINE problem.

REFERENCES

[1] 2018. General Data Protection Regulation. https://gdpr-info.eu/issues/consent/.

[2] 2018. Jeb Bush's Gubernatorial Email Archive. http://americanbridgepac.org/jeb-bushs-gubernatorial-email-archive/.

[3] Rajeev Alur and Thomas A. Henzinger. 1993. *Information and Computation* 104, 1 (1993), 35–77. https://doi.org/10.1006/inco.1993.1025

[4] Quintin Armour, William Elazmeh, Nour El-Kadri, Nathalie Japkowicz, and Stan Matwin. 2005. Privacy compliance enforcement in email. In *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 194–204.

[5] Adam Barth, Anupam Datta, et al. 2006. Privacy and contextual integrity: Framework and applications. In *Proceedings of the Symposium on Security and Privacy*. IEEE, 15–pp.

[6] David Basin, Felix Klaedtke, and Samuel Müller. 2010. Monitoring security policies with metric first-order temporal logic. In *Proceedings of the 15th ACM symposium on Access control models and technologies*. ACM, 23–34.

[7] Narjes Boufaden, William Elazmeh, Yimin Ma, Stan Matwin, Nour El-Kadri, and Nathalie Japkowicz. 2005. PEEP-An Information Extraction base approach for Privacy Protection in Email. In *Proceeding of the Second Conference on Email and Anti-Spam*.

[8] Travis D Breaux, Annie Antón, et al. 2008. Analyzing regulatory rules for privacy and security requirements. *Software Engineering, IEEE Transactions on* 34, 1 (2008), 5–20.

[9] Carnegie Mellon University. 2016. Enron Email Dataset. https://www.cs.cmu.edu/~./enron/.

[10] Vitor R Carvalho, Ramnath Balasubramanyan, and William W Cohen. 2009. Information leaks and suggestions: A case study using mozilla thunderbird. In *Sixth Conference on Email and Anti-Spam (CEAS)*.

[11] Vitor R Carvalho and William W Cohen. 2007. Preventing information leaks in email. In *International Conference on Data Mining*. SIAM, 68–77.

[12] Checkpoint. 2017. Data Loss Prevention Software Blade. https://www.checkpoint.com/products/dlp-software-blade/.

[13] Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 740–750. https://doi.org/10.3115/v1/D14-1082

[14] Omar Chowdhury, Andreas Gampe, et al. 2013. Privacy promises that can be kept: A policy analysis method with application to the HIPAA privacy rule. In *Proceedings of the 18th SACMAT*. ACM.

[15] Omar Chowdhury, Limin Jia, Deepak Garg, and Anupam Datta. 2014. Temporal Mode-Checking for Runtime Monitoring of Privacy Policies. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014*. 131–149. https://doi.org/10.1007/978-3-319-08867-9_9

[16] Omar Chowdhury, Limin Jia, Deepak Garg, and Anupam Datta. 2014. Temporal mode-checking for runtime monitoring of privacy policies. In *International Conference on Computer Aided Verification*. Springer, 131–149.

[17] Natalia Criado and Jose M Such. 2015. Implicit Contextual Integrity in Online Social Networks. *Information Sciences: an International Journal* 325 (2015), 48–69.

[18] Andrew Fiore and Jeff Heer. 2004. UC Berkeley Enron email analysis. http://bailando.sims.berkeley.edu/enron_email.html.

[19] Deepak Garg, Limin Jia, and Anupam Datta. 2011. Policy auditing over incomplete logs: theory, implementation and applications. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 151–162.

[20] Google. 2016. Scan your email traffic using data loss prevention. https://support.google.com/a/answer/6280516?hl=en.

[21] Tanya Goyal, Sanket Mehta, and Balaji Vasan Srinivasan. 2017. Preventing Inadvertent Information Disclosures via Automatic Security Policies. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 173–185.

[22] Hsi-Ming Ho, Joël Ouaknine, and James Worrell. 2014. Online Monitoring of Metric Temporal Logic. In *Proceedings of the Runtime Verification - 5th International Conference*. 178–192. https://doi.org/10.1007/978-3-319-11164-3_15

[23] Matthew Honnibal and Ines Montani. 2018. SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io.

[24] Chetan Kalyan and Krithika Chandrasekaran. 2007. Information leak detection in financial e-mails using mail pattern analysis under partial information. In *Proceedings of the 7th WSEAS International Conference on Applied Informatics and Communications (AIC)*. Citeseer, 104–109.

[25] Gilad Katz, Yuval Elovici, and Bracha Shapira. 2014. CoBAn: A context based model for data leakage prevention. *Information sciences* 262 (2014), 137–158.

[26] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. *CoRR* abs/1603.01360 (2016). arXiv:1603.01360 http://arxiv.org/abs/1603.01360

[27] McAfee. 2017. Total Protection for Data Loss Prevention (DLP). https://www.mcafee.com/us/products/total-protection-for-data-loss-prevention.aspx.

[28] Microsoft. 2017. Data loss prevention in Exchange 2016. https://support.google.com/a/answer/6280516?hl=en.

[29] Microsoft. 2017. title = Understanding Ethical Walls. url = https://technet.microsoft.com/en-us/library/aa996850(v=exchg.141).aspx.

[30] Helen Nissenbaum. 2010. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford Law Books.

[31] Richard A Posner. 1977. Right of privacy, the. *Ga. L. Rev.* 12 (1977).

[32] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *CoRR* abs/1711.10160 (2017). arXiv:1711.10160 http://arxiv.org/abs/1711.10160

[33] Satyaki Sen, Saikat Guha, et al. 2014. Bootstrapping privacy compliance in big data systems. In *Proceedings of the Symposium on Security and Privacy*. IEEE.

[34] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. *CoRR* abs/1611.01603 (2016).

[35] Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2012. *Data Leakage Detection/Prevention Solutions*. Springer. 17–37 pages.

[36] Jitesh Shetty and Jafar Adibi. 2004. The Enron email dataset database schema and brief statistical report. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.9477.

[37] Yan Shvartzshanider, Ananth Balashankar, Thomas Wies, and Lakshminarayanan Subramanian. 2018. RECIPE: Applying Open Domain Question Answering to Privacy Policies. In *Proceedings of the Workshop on Machine Reading for Question Answering*. 71–77.

[38] Salvatore J Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern, and Ke Wang. 2006. Behavior-based modeling and its application to email analysis. *Transactions on Internet Technology (TOIT)* 6, 2 (2006), 187–221.

[39] Symantec. 2017. Data Loss Prevention. https://www.symantec.com/products/information-protection/data-loss-prevention.

[40] Prasanna Thati and Grigore Rosu. 2005. Monitoring Algorithms for Metric Temporal Logic Specifications. *Electronic Notes in Theoretical Computer Science* 113 (2005), 145–162. https://doi.org/10.1016/j.entcs.2004.01.029

[41] The University of Edinburgh. 2017. http://www.inf.ed.ac.uk/teaching/courses/tts/assessed/roles.txt.

[42] US Department of Education. 2016. Family Educational Rights and Privacy Act (FERPA). https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html.

[43] Curtis C Verschoor. 2002. It isn't enough to just have a code of ethics.(Ethics). *Strategic Finance* 84, 6 (2002), 22–24.

[44] F Westin Alan. 1967. *Privacy and freedom*. Atheneum. New York.

[45] Polina Zilberman, Shlomi Dolev, Gilad Katz, Yuval Elovici, and Asaf Shabtai. 2011. Analyzing group communication for preventing data leakage via email. In *International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 37–41.