A Hybrid Framework for Data Loss Prevention and Detection

Elisa Costante
SecurityMatters
Email: elisa.costante@secmatters.com

Davide Fauri, Sandro Etalle, Jerry den Hartog, Nicola Zannone
Eindhoven University of Technology
Email: {d.fauri, s.etalle, j.d.hartog, n.zannone}@tue.nl

***Abstract*—Data loss, i.e. the unauthorized/unwanted disclosure of data, is a major threat for modern organizations. Data Loss Protection (DLP) solutions in use nowadays, either employ patterns of known attacks (signature-based) or try to find deviations from normal behavior (anomaly-based). While signature-based solutions provide accurate identification of known attacks and, thus, are suitable for the *prevention* of these attacks, they cannot cope with unknown attacks, nor with attackers who follow unusual paths (like those known only to insiders) to carry out their attack. On the other hand, anomaly-based solutions can find unknown attacks but typically have a high false positive rate, limiting their applicability to the *detection* of suspicious activities. In this paper, we propose a hybrid DLP framework that combines signature-based and anomaly-based solutions, enabling *both* detection and prevention. The framework uses an anomaly-based engine that automatically learns a model of normal user behavior, allowing it to flag when insiders carry out anomalous transactions. Typically, anomaly-based solutions stop at this stage. Our framework goes further in that it exploits an operator's feedback on alerts to automatically build and update signatures of attacks that are used to timely block undesired transactions before they can cause any damage.**

I. Introduction

Data loss, i.e. the unauthorized disclosure of sensitive information from a corporate network or a database [1], is a major threat for organizations. Organizations can lose their competitive advantage if confidential information is stolen. Moreover, data breaches can affect customers' perception towards a company's image by decreasing its reputation, especially if sensitive personal information is leaked. Unsurprisingly, data leakages are typically propagated by Insider Threats [2].

To minimize the risk of data breaches, organizations often employ Data Loss Protection (DLP) solutions as a defense mechanism [3]. DLP solutions monitor the access and exchange of confidential data to identify unauthorized disclosure or improper usage [4]. To distinguish allowed from malicious transactions, DLP systems maintain a model of either allowed (whitelisting) or malicious (blacklisting) behavior. This model can either be specified based on an expert's knowledge or learned from past transactions.

A blacklist with signatures describing well-known attacks hardly produces any false positive, allowing it to be used for *prevention* by blocking attacks before they are executed. However, such an approach cannot detect unknown attacks. In particular, it is often easy for insiders to avoid blacklisting-based detection. The insider has (privileged) access to systems, and can usually carry out actions that qualify as data leakage without breaking the system's rules and/or using leakage paths that are specific to the target system, and which cannot be considered in a general-purpose signature.

Anomaly-based solutions, which learn a model of normal behavior and flag any deviation from the model as a suspicious activity, can find unknown attacks but may have a high false positive rate. As such, anomaly-based systems are typically used only for *detection*; they raise an alert upon detecting a suspicious activity but do not block the activity. Alerts typically have to be manually analyzed to determine whether they are false positive or they correspond to an actual attack. This, however, has high operational costs and a lengthy response time to security incidents.

Despite the number of technological steps that have been taken to reduce data losses, cases of data breaches are not decreasing. A recent study conducted by the Open Security Foundation shows that over 502 million records, including credit card numbers, access credentials and other personal information, were leaked in the first half of 2014 [5].

According to MacDonald [6], the main problem lies in the fact that enterprises have long overspent on threat prevention and underspent on detection and response. He points out that it is impossible to have a signature available before an attack, but it is possible to have it after. Thus, it is necessary to have the ability to define new signatures as soon as new attacks are identified.

In this paper, we address the problem of identifying and reacting to insider threats by monitoring user activities and detecting anomalous behavior. In addition, once an anomaly has been flagged as suspicious we create on-the fly rules that are able to block any transactions that matches the suspicious pattern. Summarizing, we use an anomaly-based engine to detect anomalous transactions and, once the security operator flags an alert as malicious, our framework automatically creates attack signatures from the alert to prevent the execution of similar activities in the future.

To block new attacks, we design and integrate a prevention system with a white-box anomaly detection system in the style of [7]. The key characteristic of a white-box approach is that it provides an operator with the root causes of alerts. This allows the operator to interpret an alert and determine whether the alert is a false positive (i.e., a legitimate transaction marked as a suspicious activity) or a true positive (i.e., an actual attack).

© 2016, Elisa Costante. Under license to IEEE.
DOI 10.1109/SPW.2016.24

IEEE computer society

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

This feedback is used to improve the model for detection (false positive) or for prevention (true positive). In the latter case, the root causes of alerts are used to create and maintain blocking (or warning) rules that are used to prevent (or signal) the execution of the flagged activities in the future. In this work, we focus our attention on database activities. Nonetheless, our framework can be easily adapted to detect and prevent data loss by analyzing other types of data (e.g., network packets).

Our framework offers several benefits compared to existing DLP solutions. First, the anomaly-based component of the framework tailors itself to the user's behavior, making it possible to detect unknown and insider attacks. In addition, it overcomes the limitations of existing anomaly-based solutions by reducing the response time to alerts and consequently the spread of data leakages and the damages they cause. Finally, our framework reduces operational costs for handling suspicious activities in that alerts for similar malicious activities do not have to be reexamined. We have validated the framework using both synthetic and real-life datasets. In particular, we tested our approach using a log of database transactions executed within a service provider in the Netherlands.

The paper is structured as follows. The next section presents an overview of DLP solutions and discusses related work. Section III introduces a motivating example and presents an overview of our framework. Section IV presents an approach to white-box anomaly-based detection, and Section V defines how the rule base used for prevention is constructed and maintained. Section VI presents experiment results. Finally, Section VII concludes the paper and provides directions for future work.

II. Overview of DLP solutions

In this section we categorize existing DLP solutions and identify their main benefits and shortcomings, which determine the aim for our hybrid approach.

DLP solutions usually aim either at *detection*, i.e. raising an alert when suspicious activities are observed, or at *prevention*, i.e. blocking malicious activities. In either cases a model distinguishing normal from suspicious activities is needed. We identify two main dimensions to characterize the model underlying a DLP solution: *i) filtering* approach, which describes whether permitted uses or misuses are captured, and *ii) model construction*, which describes how the model is constructed. Notice that there are other dimensions which can be used to classify DLP solutions (e.g., network-based vs. host-based). However, these dimensions are less relevant for the purpose of this work, and thus we will not consider them. We refer to [8], [3] for more detailed taxonomies.

Filtering can be based on *blacklisting* or *whitelisting*. Blacklisting is used to represent transactions that are not allowed (e.g., well-known threats or undesired behavior). Every transaction matching an element in a black-list is blocked (prevention) or generates an alert (detection). In contrast, whitelisting is used to specify the allowed behavior; hence, only transactions that do match the model are considered legitimate.

Two main types of approaches have been proposed to build the model: *specification-* and *learning-based*. In a specification-based approach, the model is defined based on an expert's knowledge and background. This approach leads to very accurate models (low false positive rate) but exhibits some drawbacks as well. For instance, specification-based blacklisting systems (also called *signature-based*) can detect known attacks for which a signature is provided but cannot detect unknown attacks (e.g., those caused by exploiting zero-day vulnerabilities). This is a major limitation for nowadays IT systems, in which most damages caused by malicious activities can be related to previously unseen attacks (consider, for instance, the Sony case in the domain of data leakage [9] or the Stuxnet case in the domain of cyber attacks to industrial networks [10]). On the other hand, a specification-based whitelisting model can be used to detect unknown attacks, but maintaining the white-list up-to-date might be too costly, especially in highly dynamic environments. The effectiveness of specification-based solutions strongly depends on the quality of the model. The definition of such a model, however, is time-consuming and error prone, and requires a deep knowledge of the application domain.

Learning-based approaches significantly reduce this effort; they automatically learn the model using machine learning or statistical modeling techniques. Clearly, these approaches might create models that are less accurate than those manually specified; hence, they are incline to a high *false positive rate*, i.e. a large number of alerts are generated for legitimate actions. Since each alert has to be analyzed by a human operator, anomaly-based solutions have a high operational cost. However, their potential of detecting unknown attacks (when combined with whitelisting) together with the possibility of automatically creating the model, makes them an attractive solution.

In the remainder of the section, we review existing specification-based and learning-based solutions. As the focus of this work is on the analysis of database activities, we mainly survey solutions that can be applied for the detection and prevention of data breaches in database systems.

a) Specification-based solutions: Specification-based solutions relying on either blacklisting or whitelisting, are widely used nowadays for DLP. Signature-based (i.e., specification-based blacklisting) solutions use a model comprising of rules that define patterns of malicious activities or fingerprints of sensitive data that are not allowed to leave the organization network, and block any transaction matching the model.

The fingerprinting of confidential documents or database records (e.g., credit card number) is commonly used by organizations to prevent data loss. A fingerprint is the hash value of a set of data [3]. Every transaction is inspected and its content is fingerprinted. The resulting hash value is compared with fingerprints of confidential information: if the value matches, the transaction is blocked. Since each fingerprint is computed from a sequence of words (or letters), a change in one character of the data results in a different hash value. This means that fingerprinting can be bypassed by slightly rephrasing the content (e.g., by replacing some characters). To overcome this problem, Shapira et al. [11] propose to fingerprint the core confidential contents, ignoring non-relevant (non-confidential) parts of a document. This way, escaping the detection requires

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

more extended changes to the document. There are also methods that create a model of sensitive values using keywords, regular expressions, text classification [12], and information retrieval [13], [14], to detect the presence of sensitive data leaving the organization perimeter. Among signature-based solutions for DLP, we can also find several network Intrusion Detection Systems (IDS). For instance, Snort [15], the most widely used signature-based IDS, listens to the network traffic and blocks transactions that match its internal rules. Snort can be adapted to detect unauthorized disclosure of sensitive data over the network, e.g. by creating string matching rules.

Though typically not called DLP, access control [16] is a form of specification-based whitelisting. Access control mechanisms rely on policies defining the actions users can perform on an object. There are access control mechanisms, e.g. based on XACML [17], which can act both as blacklisting and whitelisting. These mechanisms allow the specification of both positive and negative authorization policies and can be configured to map non-applicable and indeterminate decisions to either permit or deny decisions depending on the context of use. Policy authoring, however, has been proven to be time consuming and error prone [18], [19]. Moreover, access control mechanisms are too rigid and are not suitable for complex social systems like healthcare.

b) Learning-based solutions: Learning-based solutions are usually based on whitelisting [20], [21], [22], [23], [24], [25], [26], [27], [28]. These systems, called anomaly-based, automatically learn a model of normal behavior by observing past activities and flag deviations from the model as anomalies.

Different approaches and sets of features (to characterize transactions) have been proposed to learn the model underlying anomaly-based solutions. For instance, Fonseca et al. [22] build normal behavior profiles based on the assumption that SQL commands and the order in which queries are executed are relevant. During detection, if an attacker executes valid queries but in an incorrect order, an alert is raised. In [24] normal profiles are built using a Naïve Bayes classifier. The system learns to predict the *userid* or *role* of users based on the SQL command, tables and columns in the query. When a new query arrives, an alarm is raised if the userid (or role) predicted by the classifier does not match the actual value. Wu et al. [28] use a similar approach with an extended feature space. Mathew and Petropoulos [25] propose to profile normal behavior based on the data users retrieve. A mixed approach combining result-centric and context-centric features is presented in [23] in which a mining algorithm is used to define association rules between context and result set. This way, the same request may be legitimate if performed within one context but abnormal within another context. In [27] normal profiles are represented in terms of statistical distribution: if a new query does not match the original probability distribution, it is considered anomalous. The white-box anomaly-based detection system in [7] aims to detect and rank alerts within DBMS by assessing their anomaly level, i.e. to what extent database activities are anomalous, based on a rich set of syntax-, context- and result-centric features.

A drawback of existing solutions is that they provide little or no support for alert handling [29]. In particular, when an alert is raised, it is often accompanied only with information about the anomaly level of the event that raised the alert, e.g. *deviation degree* and *anomaly score*. In addition, these solutions do not provide support for "enforcing" an alert, e.g. by automatically creating response actions that can be taken when the same (or similar) alerts come along.

To the best of our knowledge, no blacklisting learning-based solutions have been proposed for database systems. Although such solutions are conceivable, there are intrinsic problems in their definition. First, the availability of attack datasets is typically scarce, making it difficult to learn a model. More importantly, database attacks, even of the same type, can vary significantly from each other, making it difficult to identity distinguishing features for their characterization.

c) Hybrid Solutions: A few IDS combine signature-based and anomaly-based solutions. However, they either assemble these two techniques with no feedback loops and, thus, are not able to define new signatures [30], or infer signatures from predefined data-mining schemes [31]. In both cases, domain knowledge is not exploited for detection and alert handling.

III. Approach

This section illustrates the challenges in data loss detection and prevention through a running example within the healthcare domain and presents a framework to address these challenges.

A. Motivating Example

This section introduces a case study in healthcare, which is used to illustrate the challenges concerning data loss prevention and detection. A local hospital provides treatment for a variety of diseases, ranging from common flu to HIV. Patient information is stored in a central database at the hospital in the form of electronic health records. Our scenario focuses on two different employees, Rob, a doctor, and Jos, a system administrator. Due to his duties, Rob often accesses patient information concerning diagnoses and may work at any hour of the day. Jos, on the other hand, typically works from 8am to 5pm and accesses patients' contact information but not their diagnoses and prescribed treatments. Fig. 1 shows the profile of normal behavior for these two users, while Fig. 2 shows the usage of IP addresses ranges. Intuitively, the profiles describe how likely users perform actions with the given characteristics (called *transaction features*). In Fig. 1 and subsequent examples, we use an illustrative set of features to characterize transactions. We refer to [7] for an exhaustive list of transaction features for the characterization of database activities. Within this setting, consider the following insider threat scenario.

Example 1. *A pharmaceutical firm is interested in the testing of a new drug to treat HIV infection. Eve, who is involved in the commercialization of the drug, knows that* Jos *works as a system administrator for a healthcare provider with thousands of patients. Eve persuades* Jos *to provide her the list of all patients at the facility suffering from HIV. To accomplish this task,* Jos *queries the database to retrieve the email address of those patients. However, he cannot obtain the complete list*

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

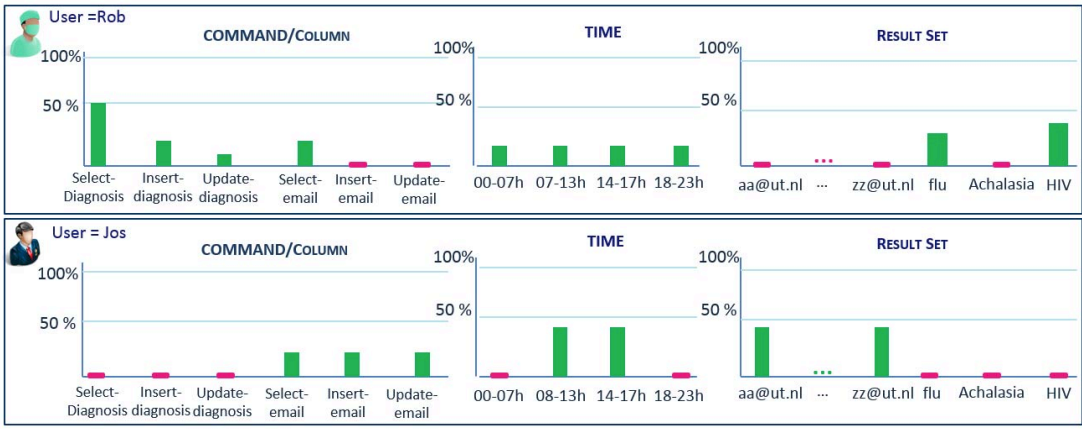


Figure 1: Example of (partial) profiles for Rob and Jos.

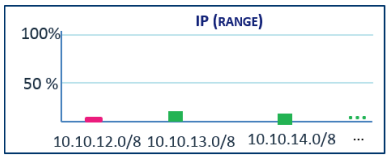


Figure 2: Example of profile for IP (range).

Alert ID	User	Command/Column	Time	Result set	IP
001	jos	Select/email	07:09	{aa@ut.nl,. . . ,zz@ut.nl}	10.10.12.5
002	jos	Select/diagnosis	07:14	{flu}	10.10.12.22
003	jos	Select/diagnosis	07:14	{HIV}	10.10.12.22
004	jos	Select/drugs	08:01	{Atripla}	10.10.12.22
005	rob	Select/diagnosis	23:54	{Achalasia}	10.10.12.23

Table I: Example of alerts. The cells in red identify the root causes of the corresponding alert.

using a single SQL query. The access to the health record of patients in certain therapy groups is subject to restrictions. For each of these patients Jos *has to check her record individually and then retrieve the email address if the patient is affected by HIV. To not arouse suspicion,* Jos *accesses the database from a workstation in a less visible corner of the office, and carries out the operation in the early morning.*

Jos, as a system administrator, has permission to access the records in the hospital database although for IT maintenance purposes only. Thus, specification-based prevention approaches (e.g., access control or blacklisting) cannot prevent this attack. At most, a blacklist could specify that Jos can only access a limited number of records per day. This would slow Jos down but not prevent him from getting the data spread over a number of days. On the other hand, an anomaly detection approach could detect that Jos misbehaved. As shown in Table I, Jos performs actions that do not match his profile on several aspects. However, although most anomaly-based approaches are able to signal anomalous transactions to the security operator, they offer very little useful information to analyze the alerts.

White-box anomaly detection [7], in addition to raising alerts, also indicates their root causes, i.e. the features of the action that are unusual given Jos's profile. For instance, alert 002 has the following root causes: (i) Jos executed a select operation on column diagnosis; (ii) Jos was active before his usual working hours; and (iii) Jos retrieved data that he typically does not access. These root causes provide the security operator with crucial information to analyze the alert and determine whether a response is needed. Yet, how to form and execute this response is not yet addressed by existing anomaly-based solutions. Moreover, anomaly-based approaches can be characterized by a high false positive rate. The main problem is that any behavior not previously observed is marked as an anomaly. For instance, alert 005 indicates that Rob retrieved the diagnosis for a patient affected by Achalasia. Achalasia is a very rare disease and Rob never treated a patient affected by this disease in the past. However, doctors can retrieve the diagnosis of the patients they treat regardless of what the diagnosis is; thus, this alert can be considered a false positive.

B. Framework

As illustrated in the previous section, existing DLP solutions are not fully able to cope with the problem of data breaches. In this section, we present a hybrid framework that combines signature-based and anomaly-based approaches to overcome their limitations. In particular, it provides capabilities to immediately respond to an alert by automatically creating rules that are used to block similar queries. An overview of the framework is presented in Fig. 3. It consists of five main phases: *(i)* learning; *(ii)* prevention, *(iii)* detection, *(iv)* alert analysis and *(v)* rule management.

During the learning phase, transactions are analyzed by a learning engine to create profiles of normal behavior. Here, we consider profiles created using the white-box anomaly-based solution presented in [7]. This solution specifies profiles of normal behavior in terms of feature histograms as the ones in Fig. 1. Specifically, for every feature, a histogram is learned from a given set of transactions by analyzing the frequency of feature values. However, the framework is general enough to be extended to any detection tool able to generate white-box alerts indicating the root causes of anomalous transactions.

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

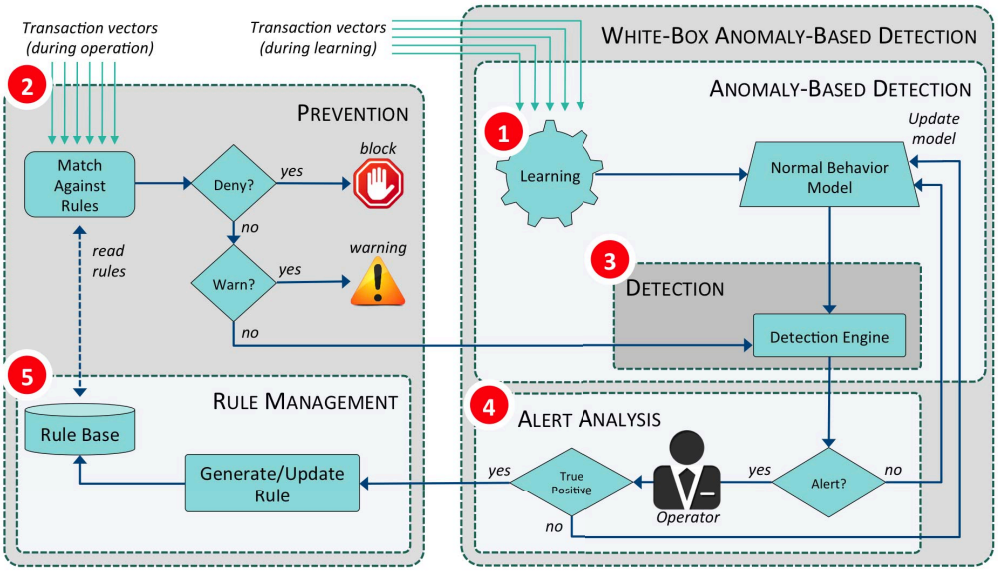


Figure 3: Framework for Data Loss Prevention and Detection

Every new transaction is analyzed by the prevention module. This module matches the transaction against a *rule base*, which comprises deny (or blocking) rules and warning rules. If the rule base contains a rule matching the transaction, the transaction is blocked or a warning is raised according to the type of rule that is fired; otherwise, the transaction is passed through to the detection engine. The detection engine aims to detect unknown attacks. In particular, this engine verifies whether the transaction matches the previously learned profiles of normal behavior (Section IV). In case of a match, the transaction is used to update the current profiles; otherwise, an alert is raised.

Alerts are analyzed by an operator, who leveraging his domain knowledge can flag them as true/false positives. If an alert is marked as a false positive (i.e., it corresponds to a legitimate activity), it is used to update the model of normal behavior. Otherwise, if the alert corresponds to a malicious activity, the operator can decide to *enforce* it, namely to automatically create some rules (devised from the alert) to be added to the rule base. In particular, the operator can decide to enforce the alert by creating a blocking rule or a warning rule. This way, when a similar transaction arrives, it can be blocked or signaled before its execution without further intervention from the operator, hence providing prevention capabilities.

Our framework offers several benefits compared to existing DLP solutions. First, the anomaly-based component of the framework allows the detection of unknown attacks. In addition, it reduces the response time to alerts and consequently the spread of data leakages and the damages they cause. For instance, after the alert 002 generated by Jos's activity, an operator could promptly act by enforcing the alert and blocking every new query for diagnosis information coming from Jos. This reduces operational costs for handling suspicious activities in that alerts for similar activities do not have to be reexamined.

IV. WHITE-BOX ANOMALY-BASED DETECTION

For the detection and analysis of anomalous activities, we adopt the white-box anomaly-based detection system proposed in [7]. This system uses an anomaly detection engine that automatically learns the normal behavior by observing past database activities (learning phase) and flags any deviation from such a behavior as an anomaly (detection phase). This DLP system offers a number of advantages compared to other DLP systems. First, while being able to detect unknown attacks because of its anomaly-based design, it results in a lower false positive ratio compared to other anomaly-based approaches. Moreover, the white-box approach enables the extrapolation of the root causes of alerts. This information is at the basis of the creation and update of a rule base for the prevention of malicious activities (Section V). In the remainder of this section, we present the main concepts underlying this detection system in terms of database transactions. However, the system is general and can be applied to other application domains (e.g., [32] applies these ideas to network intrusion detection).

Definition 1 (Transaction)**.** *A* transaction *is a triple* $\langle Q,RS,CI\rangle$ *where* Q *is an SQL query,* RS *is the corresponding result set and* CI *denotes the context in which* Q *is executed.* $\mathcal{T}$ *denotes the set of all possible transactions.*

Transactions are characterized through features.

Definition 2 (Feature)**.** *A* feature $f : \mathcal{T} \longrightarrow V_f$ *is a function associating to each transaction a value from its codomain* V_f*. A* feature value v_f *is the result of applying function* f *to a transaction* $t \in \mathcal{T}$*, i.e.* $v_f = f(t)$*. A* feature space $\langle f_1, ..., f_n\rangle$ *is an ordered sequence of features.*

Features are defined with respect to query, result set and context information. For instance, syntax-centric features are

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

used to characterize aspects related to the specification of the queries (e.g., SQL *command* and *table* list); context-centric features include the *user* who submitted the query and the *time* when the transaction was executed.

Definition 3 (Transaction vector)**.** *Given a feature space $\langle f_1, ..., f_n\rangle$ and a transaction t, a* transaction vector *$\langle v_{f_1}, ..., v_{f_n}\rangle$ is a sequence of feature values v_{f_i} where $v_{f_i} = f_i(t)$. $\mathcal{X}$ denotes the set of possible transaction vectors.*

Intuitively, a transaction vector is the feature representation of a transaction. In some cases, anomalies can only be detected by considering a set of features together. To this end, we use the notion of compound feature proposed in [33].

Definition 4 (Compound Feature)**.** *Given a feature space $\langle f_1, ..., f_n\rangle$, a* compound feature *$c$ is an element of $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{P}(\{f_1, ..., f_n\})$.*

Given a feature space $\langle f_1, f_2, f_3, f_4\rangle$, we use notation $\{f_1, f_2\&f_4, f_3\}$ to represent the set of compound features $\{\{f_1\}, \{f_2, f_4\}, \{f_3\}\}$ and we use $\{v_{f_1}, v_{f_2}\&v_{f_4}, v_{f_3}\}$ to represent the values the compound features take.

Definition 5 (Detection Engine, Root Cause and Anomaly)**.** *Given a feature space $\langle f_1, ..., f_n\rangle$, a set of compound features $\mathcal{C} \subseteq \mathcal{P}(\{f_1, ..., f_n\})$ and a transaction vector $x \in \mathcal{X}$, a detection engine is a function $m : \mathcal{X} \times \mathcal{C} \to \{anomalous, normal\}$. In addition, we say that:*

- *$x \in \mathcal{X}$ is an* anomaly *iff $\exists\ c \in \mathcal{C}$ s.t. $m(x, c) = anomalous$;*
- *$c \in \mathcal{C}$ is a* root cause *iff $m(x, c) = anomalous$.*

Whenever a detection engine receives a transaction, it will raise an *alert* if the corresponding transaction vector is flagged as an anomaly. The root causes allow an operator to interpret the anomaly (alert analysis phase) and, thus, decide how future occurrences of (similar) transactions should be handled, i.e. either improving the detection model if it is a false positive or creating a prevention rule if the alert is a true positive.

Example 2. *The detection engine presented along with our running example (Section III-A) uses feature space $\langle$*user, command, columns, time, resultSet, IP*$\rangle$. Anomalies are determined with respect to the set of compound features $\mathcal{C} =$* {user&command&columns, user&time, user&resultSet, IP}*. Consider a transaction $t = \langle Q, RS, CI\rangle$ raising alert 003 in Table I. The result set is $RS = \{HIV\}$, and context information is $CI = \{user = jos,\ time = 07{:}14, IP = 10.10.12.22\}$; the corresponding transaction vector is $x = \langle jos, select, \{diagnosis\}, 07{:}14, \{HIV\}, 10.10.12.22\rangle$. In addition, suppose that the detection engine m gives the following results:*

- *$m(x,$* user&command&columns*$) = anomalous$*
- *$m(x,$* user&resultSet*$) = anomalous$*
- *$m(x,$* user&time*$) = anomalous$*
- *$m(x, c) = normal$ for every other $c \in \mathcal{C}$*

Thus, transaction vector x is flagged as anomalous, and compound features user&command&columns*,* user&resultSet *and* user&time *are the root causes of the alert.*

V. Prevention and Rule Management

In this section, we discuss the prevention capabilities of our framework. In particular, we first present a formalism to represent the rule base and to match new transactions against it. Then, we show how the rule base is updated when the operator flags an alert as a true positive.

A. Rule Modeling and Matching

The rule base allows the timely response to anomalous transactions. It consists of signatures of malicious activities represented as sequences of (anomalous) features' values. The data structure storing the rule base should allow matching and update operations to be easy and computationally efficient. Moreover, it should be understandable and easy to maintain and explore. To this end, we represent it in form of a tree.

Definition 6 (Rule Tree)**.** *Given a feature space $\langle f_1, ..., f_n\rangle$ and a set of actions $\mathcal{A}$, a* rule tree *Δ^i rooted in feature f_i (with $i \in \{1, \ldots, n\}$) is defined as follows:*

$$\begin{cases} \text{A tree } \Delta^i : V_{f_i} \mapsto T_{i+1} \text{ is a partial function to subtrees} \\ \text{A leaf } \Delta^{n+1} \text{ is an element of } \mathcal{A}, \text{ i.e. } T_{n+1} = \mathcal{A} \end{cases}$$

where V_{f_i} is the codomain of f_i and T_i all trees rooted in f_i.

A rule tree Δ^i rooted in feature f_i can be seen as a set of edges labeled with feature values that connect f_i to its immediate subtrees. Thus, hereafter, we use notation $\{(v_1, \Delta_1^{i+1}), \ldots, (v_w, \Delta_w^{i+1})\}$ to denote the partial function that assigns Δ_j^{i+1} to $v_j \in V_{f_i}$ ($j = 1, \ldots, w$), i.e. $\Delta_j^{i+1} = \Delta^i(v_j)$ is the immediate subtree of Δ^i for v_j. In a rule tree there is often a 'default value', i.e. all but a few values will have the same subtree. We introduce symbol $\otimes$ to denote *any other value*. To capture this in the format introduced above, we use $\{(v_1, \Delta_1^{i+1}), \ldots, (v_k, \Delta_k^{i+1}), (\otimes, \Delta_{k+1}^{i+1})\}$ to denote the (total) function that assigns Δ_j^{i+1} to v_j ($j = 1, \ldots, k$) and Δ_{k+1}^{i+1} to all other values ($v_{k+1} \in V_{f_i} \setminus \{v_1, \ldots v_k\}$). It is worth noting that given a feature space comprising n features, a tree rooted in f_1 has *always* depth equal to $n + 1$.

When a new transaction arrives, it is matched against the rule base to verify whether it is has to be blocked or it should be passed on to the detection engine. For the matching, the tree is traversed; if a matching path from the root until a leaf is found, then the action represented by the leaf is fired (i.e., DENY or WARN). Otherwise, the transaction is allowed to go further.

Definition 7 (Transaction Matching)**.** *Given a feature space $\langle f_1, ..., f_n\rangle$, the partial function* match *takes a transaction vector and matches it with a rule tree to return the resulting action in $\mathcal{A}$ if any:*

$$\mathsf{match}(\langle v_{f_i}, \ldots, v_{f_n}\rangle, \Delta^i) = \mathsf{match}(\langle v_{f_{i+1}}, \ldots, v_{f_n}\rangle, \Delta^i(v_{f_i}))$$
$$\mathsf{match}(\langle\rangle, \Delta^{n+1}) = \Delta^{n+1}$$

Recall that Δ^{n+1} is an action in $\mathcal{A}$. We say a transaction t with transaction vector x matches a rule tree Δ if match(x, Δ) *is defined; we call the action it returns the* fired action*.*

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

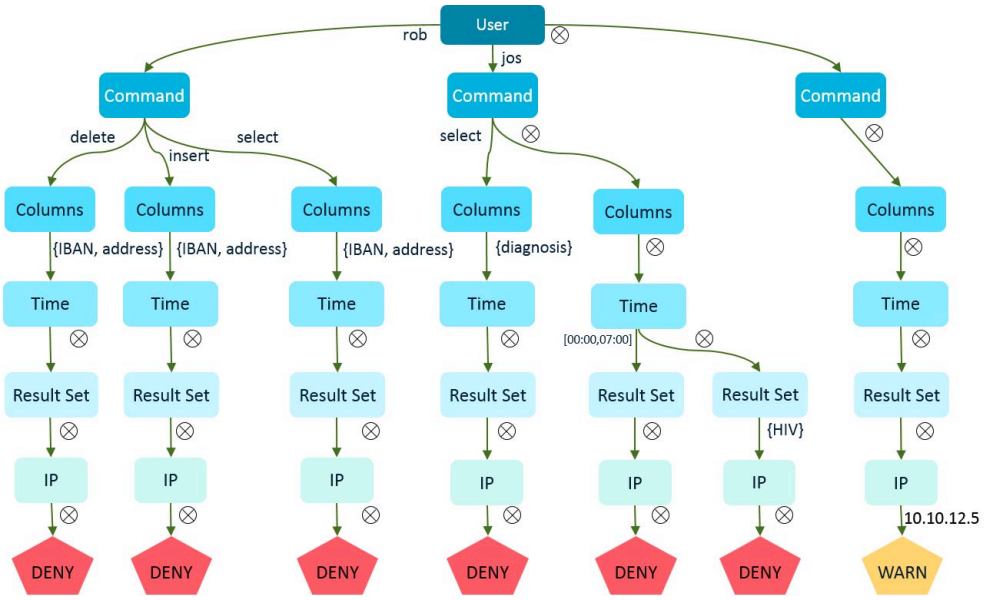


Figure 4: An example of rule tree

Note that the matching of a transaction with a rule base returns only one action. This is guaranteed by the tree structure used to represent the rule base.

Example 3. *Fig. 4 shows a sample rule tree for our running example. The rule tree states that* Rob *(a doctor) cannot execute transactions with command* delete, insert *or* select *on columns* IBAN *and* address*. The rule base also prevents* Jos *from* selecting *column* diagnosis *and from retrieving values including* HIV *in the result set. Clearly,* Jos *can do any other transaction not matching the tree. In addition, the rule tree raises a warning for transactions coming from IP* 10.10.12.5*, indicating, for instance, that activities from this IP are kept under observation.*

B. Rule Creation & Rule Base Update

Given an alert, generated by the detection engine, the operator has to decide whether the alert is a true or a false positive. It is worth noting that a transaction can be flagged as anomalous because of different reasons, i.e. more than one compound feature can be anomalous for the transaction. Essentially, the root causes of an alert provide different and independent explanations of why the transaction is anomalous. The operator should decide for which root causes a rule should be created along with the action to be enforced.

Definition 8 (Rule Creation)**.** *Given a feature space $\langle f_1, \ldots, f_n \rangle$, a set of compound features $\mathcal{C} \subseteq \mathcal{P}(\{f_1, ..., f_n\})$, an anomalous transaction vector $x = \langle v_{f_1}, \ldots, v_{f_n} \rangle$ and an action $a \in \mathcal{A}$, the rule tree for a root cause $c \in \mathcal{C}$ is Δ^1 given by ($i \in \{1 \ldots n\}$):*

$$\Delta^i = \begin{cases} \{(v_{f_i}, \Delta^{i+1})\} & \text{if } f_i \in c \\ \{(\otimes, \Delta^{i+1})\} & \text{otherwise} \end{cases}$$
$$\Delta^{n+1} = a$$

Next, we exemplify rule creation using our running example.

Example 4. *Let assume that* Jos*, noticing he cannot access column* diagnosis *any longer, decides to select column* drugs *because he knows that, if a patient takes* Atripla*, then he almost certainty has* HIV*. This action from* Jos *produces the following transaction vector $x = \langle jos, select, \{drugs\}, \{Atripla\}, 08{:}01, 10.10.12.22 \rangle$. Since, according to* Jos*'s profile, he never accessed column* drugs *before, the detection engine m will give the following results:*

- $m(x, \text{user\&command\&columns}) = anomalous$
- $m(x, \text{user\&resultSet}) = anomalous$
- $m(x, c) = normal$ *for every other* $c \in \mathcal{C}$

Both compound features user&command&columns *and* user&resultSet *are anomalous. Thus, two rules can be created to prevent the execution of similar transactions, one for each root cause. These rules are shown in Fig. 5.*

Next, we define how, given a rule, the rule base is updated. We provide a general update procedure that does not make assumptions about how the rule is created. This, however, means that the rule may define an action for a transaction that already has an action defined in the rule base. To deal with this we assume, from here on, that the actions have a precedence. For example, DENY overrides WARN, meaning that the former will be used rather than both.

Definition 9 (Rule Tree Update)**.** *Given a feature space $\langle f_1, \ldots, f_n \rangle$ and an ordered set of actions $\mathcal{A}$, we define the merge of two rule trees Δ_1^i and Δ_2^i both rooted in feature f_i (using functional notation) as follows: For all $v \in V_{f_i}$,*

$$(\Delta_1^i + \Delta_2^i)(v) = \begin{cases} \Delta_1^i(v) & \text{if } \Delta_2(v) \text{ is undefined} \\ \Delta_2^i(v) & \text{if } \Delta_1(v) \text{ is undefined} \\ \Delta_1^i(v) + \Delta_2^i(v) & \text{if both are defined} \end{cases}$$
$$\Delta_1^{n+1} + \Delta_2^{n+1} = \max\{\Delta_1^{n+1}, \Delta_2^{n+1}\}$$

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

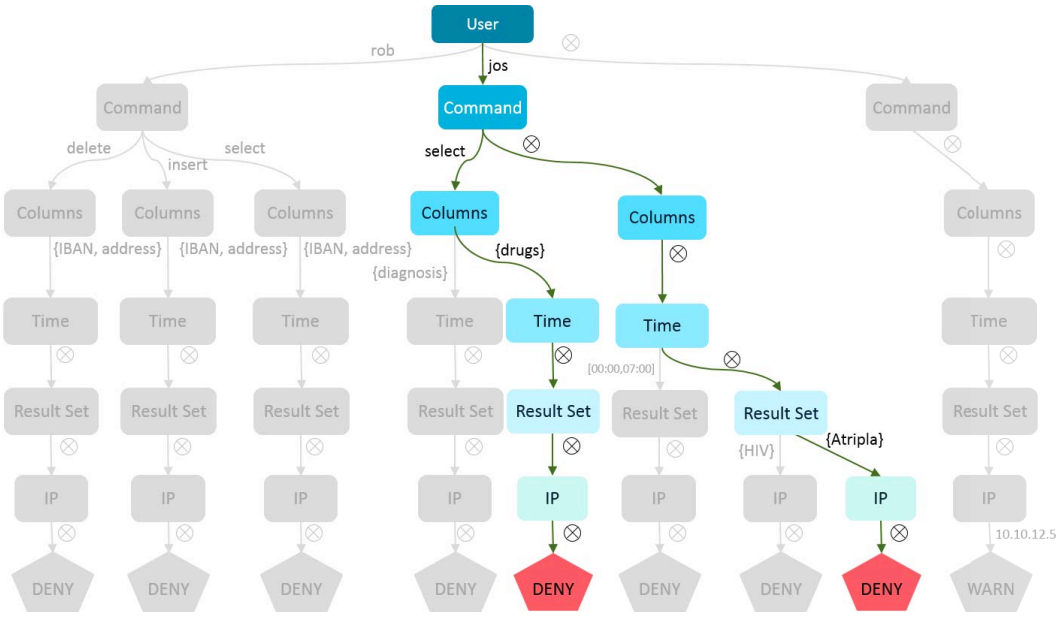


Figure 6: An example of added rules to the tree

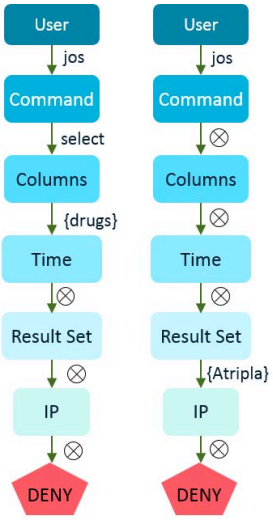


Figure 5: An example of single trees as derived from alerts.

To merge two actions we use their precedence as explained above. To merge two trees we take the subtrees in either tree where subtrees occurring in both are (recursively) merged.

Example 5. *Consider the rules created in Example 4 (Fig. 5) to respond to anomalous transaction* $x = \langle jos, select, \{drugs\}, \{Atripla\}, 08{:}01, 10.10.12.22\rangle$. *To block future executions of similar transactions, these rules are used to update the rule-tree in Fig. 4. The resulting tree is shown in Fig. 6. The first rule creates a new branch at the* column *level to block the selection of column* drugs *by* Jos. *The second rule create a new branch at the* ResultSet *level to block queries made by* Jos *that returns* Atripla *in the result set.*

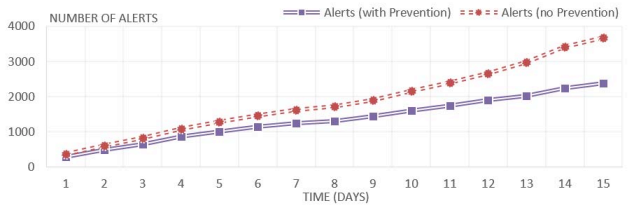
VI. Validation

To validate our approach, we have extended the RapidMiner plugin described in [7] with a prevention module based on the proposed framework. The plugin implements modules to learn normal behavior profiles and to detect anomalous transactions. The prevention module automatically creates and updates the rule base based on the alerts generated by the detection module and matches newly arriving transactions against it.

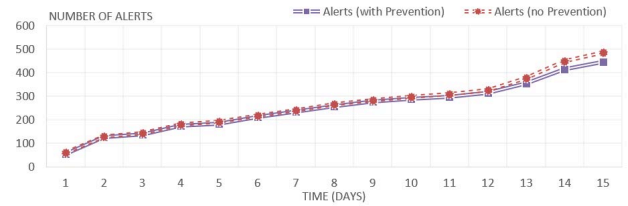
A simple analytical analysis can already prove the correctness of our approach in blocking all similar repeated attacks, i.e. those with the same root causes, as soon as the first true positive is identified. Thus, in the experiments we focus on evaluating how *effective* our approach is, in blocking new 'unwanted' behavior and thus reducing the effort for the operator. We henceforth assume that every alert raised by the detection module is a true positive: upon receiving an alert, the preventive module creates a DENY rule for every root cause, and updates the rule base accordingly. Note that we could also randomly select certain alerts as false positives, and use them to improve the detection model: this, however, does not influence the overall reduction in the number of raised alerts as subsequent occurrences of similar transactions would just be detected as normal traffic.

To assess the impact of our solution we measure the number of alerts raised over time, with and without the prevention module in place. This gives a measure of the transactions that were blocked, thus providing an indication of the reduced effort for the operator and, indirectly, of how much data loss was prevented. A precise quantification of this latter value, however, would require assigning a value to the data; while there is some work addressing this, e.g. [29], it is outside the scope of this paper. Likewise, we do not analyze the

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

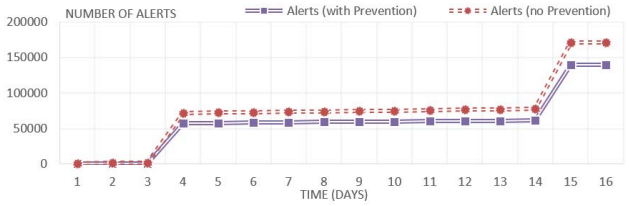


(a) Small training set

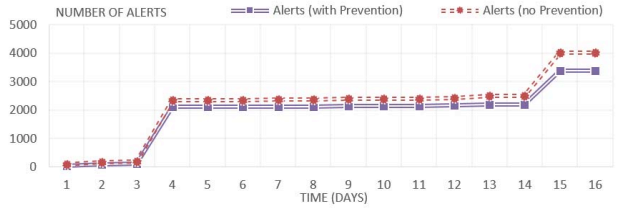


(b) Large training set

Figure 7: Number of alerts with/without prevention for synthetic data



(a) Small training set



(b) Large training set

Figure 8: Number of alerts with/without prevention for real data

effectiveness of the detection engine in terms of detection rate and false positive rate as this has already been evaluated in a previous work (see [7]): we note that a reduction in the absolute amount of alerts implies a proportional reduction in the absolute amount of false positives. We performed a number of experiments using both synthetic and real-life logs.

A. Synthetic data

In the first set of experiments, we used a synthetic dataset obtained from the healthcare management system GnuHealth (http://health.gnu.org). We simulated the normal behavior of users at a hospital (validated by domain experts), including an admin and different users, mainly doctors and nurses taking care of patients with different diseases. The dataset contains 30,490 queries spanned over a time frame of 15 days.

To evaluate how much the increase in effectiveness depends on the accuracy of the initial detection module, we varied the size of the dataset used to train the detection engine. In particular, we learned normal behavior profiles using 3% and 30% of the dataset (915 and 9,146 transactions resp.); the remaining 70% of the dataset (21,344 transactions) was analyzed by the obtained detection engines, for validation purposes.

The results of the experiments are presented in Fig. 7. The figure shows the number of alerts raised by the detection system over time. In particular, the dashed line indicates the number of alerts when only the detection module is operative, whereas the continuous line indicates the number of alerts when also the prevention module is in place. Since these transactions represent undesired behavior, the graph gives an idea of the amount of malicious operations that were blocked. Note that the two lines do not start at the same number of alerts: this is because the rule base is not updated daily but after each transaction, i.e. a new rule is added to the rule base as soon as the detection engine raises an anomaly.

As shown in Fig. 7, the use of the prevention module leads to a reduction in the number of alerts. This is in line with the fact that, over time, the detection model and the rule base gain in accuracy. Over the 15 days of our analysis, we observed a reduction of approximately 35% when the small dataset was used for the training of the detection engine (Fig. 7a) and 10% when the large dataset was used (Fig. 7b).

B. Real-life data

The real dataset is taken from the log of an (Oracle) operational database of a large IT company. The database is accessed through a web application internal to the organization, which has about 100 users. The dataset was created by enabling the DBMS auditing facility, and it contains 12,040,910 transactions. To execute the experiments, we used 3,612 and 361,227 transactions (approximately 0.03% and 3% of the whole dataset resp.) to learn the normal behavior, while we used 8,428,637 transactions (70% of the dataset) for validation.

The results of the experiments are presented in Fig. 8. Clearly, the difference in the number of alerts in the two cases increases with the passing of time. In particular, we can observe a reduction of approximately 18% when the small dataset was used for the training of the detection engine (Fig. 8a), and 15% when the large dataset was used (Fig. 8b). Note that two days are characterized by a large number of alerts compared to the others (recognizable in Fig. 8 for the steepness of the line). This is mainly because a significantly higher number of transactions were executed in those days.

C. Discussion

The effectiveness of our prevention system depends on how often unwanted transactions, similar to those already found, are attempted. To get an estimation of the "frequency" of unwanted behavior, we learned an intentionally incomplete model of normal behavior and used the remaining transactions to represent the unwanted behavior. Fig. 7 gives the best and worst case scenario for our approach, where repetition of unwanted behavior is respectively prevalent (35%) or 'rare' (10%), while Fig. 8 shows the typical expected behavior in a real system.

The experiments show that our approach offers prompt responses to seal leakage (by updating the rule base), reducing

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.

response time and, with it, the amount of data leaked. This provides great benefit especially considering that data leakages typically stay unsealed for days or weeks [9].

One of the main problems in detection systems is that an operator has to deal with a large number of alerts, many of which can be false positive. An advantage of our approach is that, in the long term, it reduces the amount of alerts and, at the same time, it improves the accuracy of the whole DLP system. In particular, the overall number of alerts is reduced because the prevention module blocks (already analyzed) undesired transactions without passing them on to the operator.

Finally, although our implementation is not optimized with respect to performance, the time to match a transaction against a rule base containing over a thousand rules is around 0.4 ms, which makes our approach suitable for real time applications.

VII. Conclusion

In this paper, we have presented a hybrid approach for data loss prevention and detection. Existing approaches focus either on prevention, e.g. by applying signature-based techniques that are unable to detect zero-day attacks, or on detection, e.g. by applying anomaly-based techniques that suffer a high false positive rate and thus have high operational costs. In this paper, we overcome these limitations by combining a white-box anomaly-based detection technique able to raise alerts for any previously unseen transaction, with a rule-based prevention technique that blocks transactions that an operator has previously flagged as malicious. After an operator determines whether an anomaly raised by the detection engine is an actual attack, our approach automatically creates rules (devised from the alert) and updates the rule base accordingly. Experiments show that our approach achieves promising results, providing the capability of reducing the spread of data leakage and the effort required by the operator. Finally, although we define a transaction in terms of SQL queries, the use of a different feature set to characterize transactions (e.g., to reflect request/response in a web application or packets in a control network) allows the easy adaption of our solution to different domains, like web applications, firewalls or network-based intrusion detection systems.

Acknowledgments: This work has been partially supported by the European Commission through project FP7-SEC-607093-PREEMPTIVE, by the Dutch program COMMIT through the THeCS project and by NWO through the IDEA ICS project.

References

[1] P. Gordon, "Data Leakage - Threats and Mitigation," SANS Institute, Tech. Rep., 2007.

[2] C. L. Huth, D. W. Chadwick, W. R. Claycomb, and I. You, "Guest editorial: A brief overview of data leakage and insider threats," *Information Systems Frontiers*, vol. 15, no. 1, p. 1, 2013.

[3] A. Shabtai, Y. Elovici, and L. Rokach, *A survey of data leakage detection and prevention solutions*, ser. SpringerBriefs in Computer Science. Springer, 2012.

[4] T. Wuchner and A. Pretschner, "Data loss prevention based on data-driven usage control," in *Proc. of ISSRE*. IEEE, 2012, pp. 151–160.

[5] Open Security Foundation, "Data breach trends during the first half of 2014," Report, 2014.

[6] N. MacDonald, "Architecting a new approach for continous advanced threat protection," in *Gartner Security & Risk Manag. Summit*, 2014.

[7] E. Costante, J. den Hartog, M. Petkovic, S. Etalle, and M. Pechenizkiy, "Hunting the unknown - white-box database leakage detection," in *Data and Applications Security and Privacy XXVIII*, ser. LNCS 8566. Springer, 2014, pp. 243–259.

[8] R. Koch, "Towards next-generation intrusion detection," in *Proceedings of International Conference on Cyber Conflict*. IEEE, 2011, pp. 1–18.

[9] "2014: A Year of Mega Breaches," Ponemon Institute, Res. Rep., 2015.

[10] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet," Symantec, Symantec Security Response, 2011.

[11] Y. Shapira, B. Shapira, and A. Shabtai, "Content-based data leakage detection using extended fingerprinting," *CoRR*, vol. abs/1302.2028, 2013.

[12] M. Hart, P. Manadhata, and R. Johnson, "Text classification for data loss prevention," in *PET*, ser. LNCS 6794. Springer, 2011, pp. 18–37.

[13] E. Gessiou, Q. H. Vu, and S. Ioannidis, "IRILD: an Information Retrieval based method for Information Leak Detection," in *Proc. of Eur. Conf. on Computer Network Defense*. IEEE, 2011, pp. 33–40.

[14] J. M. Gómez-Hidalgo, J. M. Martın Abreu, J. Nieves, I. Santos, F. Brezo, and P. G. Bringas, "Data leak prevention through named entity recognition," in *Proceedings of International Conference on Social Computing*. IEEE, 2010, pp. 1129–1134.

[15] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proc. of LISA*. USENIX Association, 1999, pp. 229–238.

[16] P. Samarati and S. D. C. d. Vimercati, "Access Control: Policies, Models, and Mechanisms," in *FOSAD*. Springer, 2001, pp. 137–196.

[17] OASIS XACML Technical Committee, "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, 2013.

[18] G. Hughes and T. Bultan, "Automated Verification of Access Control Policies Using a SAT Solver," *STTT*, vol. 10, no. 6, pp. 503–520, 2008.

[19] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone, "Analysis of XACML Policies with SMT," in *Principles of Security and Trust*, ser. LNCS 9036. Springer, 2015, pp. 115–134.

[20] C. Bockermann, M. Apel, and M. Meier, "Learning SQL for database intrusion detection using context-sensitive modelling," in *DIMVA*, ser. LNCS 5587. Springer, 2009, pp. 196–205.

[21] C. Chung, M. Gertz, and K. Levitt, "DEMIDS: A Misuse Detection System for Database Systems," in *Integrity and Internal Control in Information Systems*, ser. IFIP AICT 37, 2000, pp. 159–178.

[22] J. Fonseca, M. Vieira, and H. Madeira, "Integrated intrusion detection in databases," in *Dependable Computing*, ser. LNCS 4746. Springer, 2007, pp. 198–211.

[23] M. Gafny, A. Shabtai, L. Rokach, and Y. Elovici, "Poster: applying unsupervised context-based analysis for detecting unauthorized data disclosure," in *Proc. of CCS*. ACM, 2011, pp. 765–768.

[24] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *VLDB Journal*, vol. 17, pp. 1063–1077, 2008.

[25] S. Mathew and M. Petropoulos, "A data-centric approach to insider attack detection in database systems," in *Recent Advances in Intrusion Detection*, ser. LNCS 6307. Springer, 2010, pp. 382–401.

[26] A. Roichman and E. Gudes, "DIWeDa - Detecting Intrusions in Web Databases," in *Data and Applications Security XXII*, ser. LNCS 5094. Springer, 2008, pp. 313–329.

[27] R. Santos, J. Bernardino, M. Vieira, and D. Rasteiro, "Securing Data Warehouses from Web-Based Intrusions," in *Web Information Systems Engineering*, ser. LNCS 7651. Springer, 2012, pp. 681–688.

[28] G. Wu, S. Osborn, and X. Jin, "Database intrusion detection using role profiling with role hierarchy," in *Secure Data Management*, ser. LNCS 5776. Springer, 2009, pp. 33–48.

[29] S. Vavilis, A. Egner, M. Petkovic, and N. Zannone, "An anomaly analysis framework for database systems," *Computers & Security*, vol. 53, pp. 156–173, 2015.

[30] O. Depren, M. Topallar, E. Anarim, and M. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. App.*, vol. 29, no. 4, pp. 713–722, 2005.

[31] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid intrusion detection with weighted signature generation over anomalous internet episodes," *Dependable and Secure Computing*, vol. 4, no. 1, pp. 41–55, 2007.

[32] O. Yüksel, S. Etalle, and J. den Hartog, "Reading between the fields: Practical, effective intrusion detection for industrial control systems," in *Proc. of SAC*. ACM, 2016.

[33] E. Costante, J. den Hartog, M. Petkovic, S. Etalle, and M. Pechenizkiy, "A behaviour-based approach to database leakage detection," *Journal of Information Security and Applications*, 2015, to appear.

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on October 31,2023 at 17:04:12 UTC from IEEE Xplore. Restrictions apply.