Received 17 August 2021; revised 2 June 2022; accepted 8 August 2022.
Date of publication 17 August 2022; date of current version 6 March 2023.

Digital Object Identifier 10.1109/TETC.2022.3198080

A Web Back-End Database Leakage Incident Reconstruction Framework Over Unlabeled Logs

YANHUA LIU, ZHIHUANG LIU, XIMENG LIU, (Senior Member, IEEE), AND WENZHONG GUO

The authors are with the College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China
CORRESPONDING AUTHOR: ZHIHUANG LIU (herecristliu@gmail.com).

This work was supported in part by the National Natural Science Foundation of China under Grants 62072109 and U1804263 and in part by the Natural Science Foundation of Fujian Province under Grants 2021J01625, 2021J01616, and 2018J07005.
This article has supplementary downloadable material available at https://doi.org/10.1109/TETC.2022.3198080, provided by the authors.

ABSTRACT In this article, we propose a web back-end database leakage incident reconstruction framework (WeB-DLIR) over unlabeled logs, designed to improve the intelligence and automation of reconstructing web back-end database leakage incidents triggered by web-based attacks in unannotated logging environments. Using WeB-DLIR, analysts can reduce the manual workload of tracing and responding to data leakage incidents. Specifically, we first design web front-end and back-end anomaly identification methods based on neural network models with a pruning strategy and fine-grained grouping clustering analysis, respectively, for completely identifying web-related abnormal events in unlabeled logs. To remove redundant abnormal events and reduce subsequent inspection work for false alarm cases, we then propose an anomaly detection result decision fusion method (DFADR). Moreover, to visualize the attack chain reflected by abnormal events, based on the decision fusion results, we propose an attack graph modeling method that can reflect the basic process of data leakage from multiple perspectives. Finally, based on the modeling results, the topology of the data leakage scenario reconstruction can be completed by further auditing the relevant logs. Experimental results using real-world datasets show that the proposed WeB-DLIR is efficient and feasible for practical applications.

INDEX TERMS Web-related data leakage, anomaly detection, attack modeling, incident reconstruction, unlabeled logs

I. INTRODUCTION

With the rapid development of Internet technology, the web and its related technologies are becoming more and more popular and widely used. Web applications with openness and ease of use are gradually replacing many traditional Internet services, more services can be handled on web applications, and a large number of web applications are widely arranged in organizations such as government agencies and enterprises. Therefore web applications involve more sensitive information, which is mainly kept in the web back-end database servers, storing core assets such as customer and confidential business data [1]. These core data attract a large number of cyber-attacks based on web platforms that are designed to access important information in the servers, especially security vulnerabilities that are frequently detected and exploited in modern web applications [2]. The resulting information leakage is causing increasing losses to people's working life, and web back-end database leakage incidents have attracted more attention and research.

Existing research about web back-end database leakage mainly concentrates on data leakage prevention (DLP) [3], [4], data leakage detection (DLD) [1], [5], and data leakage forensics (DLF) [6], [7]. Among these, DLP aims to predict and prevent data breaches before an incident occurs. DLD, on the other hand, detects potential or ongoing data leakages by analyzing multiple sources of log data [5], spotting anomalies in database transactions [1], etc. Unlike the previous two, DLF occurs more often after a data leakage attack and mainly involves collecting, pre-processing, and analyzing log data [6], tracking and displaying sensitive data propagation paths through data flow graph visualization [7], and finally identifying the data leakage incident.

The aforementioned works focus on capturing network records, detecting and preventing data breaches. To the best of our knowledge, there is currently no effective method that focuses on web back-end database leakage scenario reconstruction to present an intuitive view of security incidents. In fact, it is essential to investigate methods for reconstructing attack scenarios, which have been extensively studied in other application scenarios of cybersecurity [8], [9]. Attack scenario modeling can summarize the causal links of an attack, enabling analysts to understand how the attacker violates security regulations and to determine the impact of the attack for the prevention of subsequent attacks [8].

In addition, the above web-based data leakage research uses a variety of richly annotated security event datasets as a way to detect and prevent data breaches. Unfortunately, the current cyberspace environment is constantly changing and new attacks are occurring, making it difficult for security agencies to obtain enough samples of attacks in a short period [10]. Professional analysts usually put in a lot of work to collect labeled log data, while getting unlabeled logs is much easier [11]. However, data leakage scenario reconstruction with only unlabeled logs faces some challenges. One of the main challenges is to reduce anomalous false alarms unrelated to the data leakage scenario while ensuring the accuracy of identifying unlabeled data anomalies [12], which is critical to reducing the subsequent auditing workload of analysts. Moreover, how to correlate abnormal events and infer what activities the attacker performed [13], which will affect the quality of data leakage scenario reconstruction. Hence, in this paper, we seek to address the following challenge: *How do we utilize unlabeled multi-source logs to accurately reconstruct and visually present web back-end database leakage incidents?*

To address the above-mentioned issues, we propose a **W**eb **B**ack-end **D**atabase **L**eakage **I**ncident **R**econstruction framework (WeB-DLIR) over unlabeled logs. The main contributions of this paper are summarized below:

1) Our proposed WeB-DLIR is designed to provide powerful support for web back-end database leakage forensics in a real network environment to prevent subsequent web-based attacks. WeB-DLIR can identify abnormal events in unlabeled multi-source logs and correlate them with fusion, and then reconstruct data leakage incidents through attack modeling.
2) We propose an abnormal event detection method based on unlabeled logs at the web front and back ends. We design a deep neural network with pruning strategy for anomaly detection of payloads at the web front-end and a fine-grained grouping clustering analysis strategy for anomaly detection of database access characteristics at the web back-end.
3) We perform decision fusion on the anomaly detection results (DFADR) to retain abnormal behaviors with correlated events. Then we propose an automated attack graph-based modeling approach that is able to model the causal linkage between different correlated events, thus formalizing the attack scenario of data leakage into a problem of finding the maximum connected subgraph in visual analysis.
4) We evaluate WeB-DLIR using web-related unlabeled logs during practical enterprise data leakage periods, and the evaluation results show that our proposal helps to improve the accuracy and intelligence of anomaly detection, correlation analysis, attack modeling, and reconstruction of data leakage incidents.

The remainder of this paper is organized as follows. In Section II, we discuss the preliminaries and describe the motivation of the methods used by WeB-DLIR. In Section III, we formalize the attack model and problem description. In Section IV, we give an overview of WeB-DLIR and elaborate on the technical details of each module in WeB-DLIR. Section V evaluates the performance of WeB-DLIR. Finally, we conclude our work in Section VI.

II. BACKGROUND AND RELATED WORK

In this section, we review anomaly detection, correlation fusion, and attack modeling techniques involved in the data leakage reconstruction process. Furthermore, we clarify the motivation of the methods used or proposed in WeB-DLIR.

A. WEB-BASED ANOMALY DETECTION

Existing web-based anomaly detection methods can be categorized under web front-end-oriented and web back-end-oriented anomaly detection, where the web front-end is mainly analyzed in terms of web traffic and web payload, and the web back-end is mainly analyzed in terms of web database access. Many researchers perform anomaly detection based on web traffic [14], but anomaly identification in the traffic characteristics dimension generally only identifies the abnormal hosts associated with the preliminary stage of network attack implementation.

Therefore, researchers have focused on information that reflects malicious behaviors such as obtaining access and control privileges, that is, web front-end payloads. Identifying abnormal payloads can be considered as a special case of text classification. There are already many methods that can be applied to abnormal payload detection, for example, Chen et al. [15] investigated the method of extracting payload features and verified the accuracy and stability of the malicious payload classification model using the XGBoost algorithm. However, most of them take a lot of time to do manual feature engineering and even require expert guidance to identify important payload features. In recent years, CNN has been widely and effectively used in tasks such as sentence classification [16], so Tian et al. [17] explored the effectiveness of CNN in malicious web shell detection. However, since CNN is weak in learning sequence information [18], the method of pre-adding a BLSTM has emerged. BLSTM is good at extracting sequence features and helps CNN to obtain the most important contextual information in a sentence, which is not possible with a unidirectional LSTM. The BLSTM-CNN strategy is widely used in the field of NLP thus proving to be an effective idea, such as Xiao et al. [19] and Zhao et al. [20]. Inspired by this, our WeB-DLIR uses a BLSTM-CNN model for web front-end payload anomaly detection in the hope of achieving better identification results while reducing the workload of analysts.

On the other hand, anomaly identification of web back-end database access characteristics is also crucial. After gaining control of a critical host or server, an attacker reads sensitive data from the back-end database as a legitimate identity, which is an anomaly closely related to data leakage. In this process, it is less likely to contain malicious payload characteristics and thus more stealthy. Therefore, based on actual database access data, the use of unsupervised learning methods for analyzing the user behavior in accessing databases is the most common and effective way for database anomaly detection [1]. For example, Kul et al. [21] performed vector-based clustering analysis to identify abnormal query features for analysts to further check database security issues. In particular, the Kmeans method in unsupervised learning is widely adopted in anomaly detection, for example, Kamra et al. [22] used the Kmeans clustering method and outlier detection techniques to enhance the identification ability of abnormal access to the database. However, since the determination of the number of clusters and the initial clustering centroids in Kmeans has a large impact on its performance, we perform fine-grained grouping clustering of database access features so that the value of K can be easily determined in a small range.

To identify the key abnormal events of data leakage, we perform anomaly detection on both the front and back ends of the web.

B. ABNORMAL EVENT CORRELATION FUSION

By fusing anomaly detection results or network security incident correlation, it can achieve the goals of accurately identifying anomalies, reducing redundant alarms and false alarms, and improving the accuracy of security incident identification [23]. Zhang et al. [24] proposed a correlation analysis method for network security incidents based on attribute similarity, which can effectively improve the efficiency of security incident analysis for network administrators. Liu et al. [25] proposed a rough set-based network security incident correlation scheme to build a network security incident database and knowledge base, which solved the problem of simplification and correlation of large-scale security incidents. Ambre et al. [26] correlated network security incidents based on the idea of the probability calculation method, which reduced the false alarm rate for internal environment anomaly detection.

However, simply staying in the study of more accurate anomaly detection methods or more effective security incident extraction research may not meet the needs of reconstructing web-based data leakage scenarios, and our WeB-DLIR considers more on the supporting role of correlation analysis for subsequent analysis stages such as attack modeling.

C. SECURITY EVENT ATTACK MODELING

Attack modeling techniques aim to help analysts understand the sequence of network attack incidents or potential threat behaviors to enable the blocking of potential attacks or timely fixing of vulnerabilities [27]. The more popular attack modeling techniques are graph-based attack graph and attack tree approaches [28], on which a large number of studies have been actively explored. For example, Hossain et al. [8] proposed a method to reconstruct attack scenarios in real-time in an enterprise environment based on the attack graph approach, which can obtain a concise view of the attack scenario. Niu et al. [29] proposed a complex attack model TCAN based on dynamic attack graphs to visualize each attack step. Maciel et al. [30] model DDoS attacks based on the attack tree approach to evaluate the impact of DDoS attacks against computer systems on the system for threat risk analysis.

Attack graph and attack tree methods focus on describing the possible paths of the attack process and can be applied to analyze and predict attacker behavior, identify network vulnerabilities, etc. In WeB-DLIR, we consider attack modeling techniques that can simultaneously reflect the time series, abnormal hosts, abnormal event causality, and the attack step process to achieve an intuitive and perceptive presentation of security incidents such as web-based data leakage.

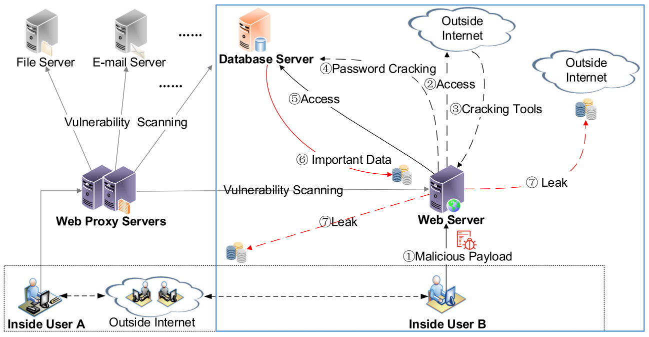


FIGURE 1. **Attack model of WeB-DLIR. In the diagram, the dashed lines represent steps that may, but not always, occur.**

III. PROBLEM STATEMENT

A. ATTACK MODEL

As shown in Figure 1, we assume that the data leakage scenario occurs inside an organization such as an enterprise. The starting point of the attack is often some legitimate hosts within the organization, such as Inside User A or Inside User B, who may be communicating with hosts on an external network or under the control of an external attacker. The attack steps are divided into three main steps: first conduct vulnerability scans on the organization's internal assets, such as file servers and database servers, to obtain vulnerability information; then the vulnerable servers are attacked with malicious payloads to gain control; finally, the server is used to read important internal database server assets and cause database leakage propagation. This paper focuses on the main steps that cause database leakage, which is the part of the blue area on the right side of Figure 1. Our WeB-DLIR consists of four main entities, namely Inside User (**IU**), Web Server (**WS**), Database Server (**DS**), and Outside Internet (**OI**).

TABLE 1. Summary of notations.

Notation	Description
D_M	web-related unlabeled multi-source logs
D_H / D_S	unlabelled web front-end / back-end logs
sip / dip / tih	source IP / destination IP / time in hours
url	Uniform Resource Locator
$num(unum)$	communication amount of sip and dip within tih (with the same url)
$sqlnum$	number of SQL operation statements
$dbtype$	database access type
D_{NH} / D_{NS}	subset of abnormal records in D_H / D_S
D_{fb}	anomaly records with correlated events

- IU is a legitimate user inside the organization, but after suffering from the control of hackers, IU performs malicious load attacks on the web server based on the vulnerability information obtained during scan probing, such as XSS attacks, SQL injection attacks, etc. (step ①).
- WS is a web server inside the organization, but several vulnerabilities can be maliciously exploited. IU attacks the WS, accesses resources such as password cracking tools from OI (step ②③), performs password cracking on DS (step ④), and gains access to important data (step ⑤); in the other case, if the WS already has access privileges to the DS, step ⑤ can be performed directly.
- The DS is a database server that stores important data inside the organization. After the WS obtains sensitive data from the DS (step ⑥), it may further pass sensitive information to the IU and OI or other hosts (step ⑦).
- OI is a network external to the organization that serves as a source for attackers to obtain scanning and cracking tools, and may also be used to store sensitive information passed from DS.

B. PROBLEM FORMULATION

Based on the above model, we now describe the definition of a web back-end database leakage incident reconstruction scheme (WeB-DLIR) over unlabeled logs. Important notations are summarized in Table 1.

Definition 1 (WeB-DLIR). Consider an web-related unlabeled multi-source log dataset $D_M{=}\{D_{m1}, D_{m2}, \ldots, D_{mn}\}$. *We first extract the subset* $\{D_{mi}\}_{i\in\{1,\ldots,n\}}$ *containing URL (e.g., web server communication logs) and the subset* $\{D_{mj}\}_{j\in\{1,\ldots,n\}}$ *containing database services (e.g., Mysql login and operation logs), which are integrated respectively and denoted respectively as: web front-end dataset* $D_H{=}\{dh_1, dh_2, \ldots, dh_{n1}\}$, *where each* dh *is represented as* $\langle tih, sip, dip, url, unum\rangle$, $n1$ *is the number of* D_H *dataset records; web back-end dataset* $D_S{=}\{ds_1, ds_2 \ldots, ds_{n2}\}$, *where each* ds *is represented as* $\langle tih, sip, dip, num, sqlnum, dbtype, \{\xi_1, \xi_2, \ldots, \xi_k\}\rangle$, $n2$ *is the number of* D_S *dataset records,* ξ_i *is the other communication statistics for sip and dip within tih. Our design goals are, in order: 1) identify abnormal hosts from* D_H *and* D_S*; 2) correlation analysis to derive and model data leakage attack scenarios; 3) accurately describe and visualize data leakage incidents.*

To accomplish the WeB-DLIR task, we define two types of detection tasks (Definition 2 and Definition 3) and a modeling task (Definition 4).

Definition 2 (Abnormal behavior detection). Detect anomalies for D_H *and* D_S*, respectively, and obtain* $D_H\_L_0$ *and* $D_S\_L_0$ *with anomaly probability labels. Abnormal behavior is defined as records that contain malicious communication payloads or deviate from normal communication patterns.*

Definition 3 (Decision fusion). This task aims to correlate and analyze $D_H\_L_0$ *and* $D_S\_L_0$ *to detect potential time and host clusters of data leakage incidents, i.e., to obtain a refined aggregation of correlated abnormal records that are closely related to data leakage,* D_{fb}*, represented as* $\{\langle tih, sip, dip, num, tp\rangle\}$*, where* tp *belongs to* $\{tp_f, tp_b\}$*, denoting abnormal web front-end behavior or abnormal web back-end behavior.*

Definition 4 (Attack modeling and scenario reconstruction). A data leakage incident includes probing, control privilege acquisition, and access to sensitive data. To model an attack event from D_{fb}*, visualize the linkage and causal logic of the anomaly time and action, and then present the main processes of a data leakage incident. Furthermore, more detailed and intuitive scenario reconstruction can be achieved under the assumption of obtaining the host identity within the organization, which can be easily satisfied in real-world organizational settings.*

IV. PROPOSED WEB-DLIR SCHEMES

A. OVERVIEW OF WEB-DLIR

The general idea of WeB-DLIR is based on the investigation steps of cyber forensics [31] and the basic stages of the problem statement described in Section III. To achieve high-accuracy anomaly detection and high-quality reconstruction of data leakage scenarios, the following four interrelated phases are involved in WeB-DLIR (shown in Figure 2).

1) *Web front-end and back-end anomaly detection:* After pre-processing work such as format conversion, data cleaning, and feature extraction on the web-related unlabeled multi-source logs, data are divided into front-end payload and back-end database access traffic. Anomaly identification is performed based on neural network modeling with a pruning strategy and fine-grained grouping clustering analysis, respectively, and

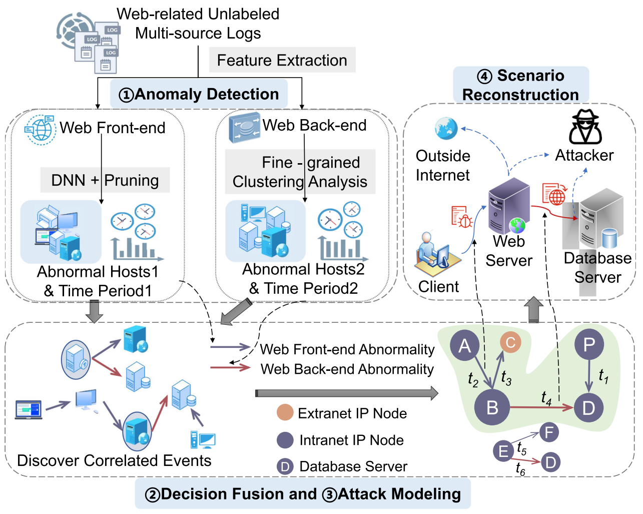

Web-related Unlabeled Multi-source Logs
Feature Extraction
①Anomaly Detection
④ Scenario Reconstruction
Web Front-end
Web Back-end
DNN + Pruning
Fine - grained Clustering Analysis
Abnormal Hosts1 & Time Period1
Abnormal Hosts2 & Time Period2
Outside Internet
Attacker
Web Server
Database Server
Client
Web Front-end Abnormality
Web Back-end Abnormality
Extranet IP Node
Intranet IP Node
Database Server
Discover Correlated Events
②Decision Fusion and ③Attack Modeling

FIGURE 2. **Overview of WeB-DLIR's architecture. The associations between the different phases are expressed by dashed lines.**

the refined methods aim to reduce false alarm and reduce the workload of subsequent analysis. We introduce this part in detail in Sections IV-B1 and IV-B2.

2) *Decision fusion of anomaly detection results:* We propose a method DFADR for decision fusion of anomaly detection results at the front and back ends and even at multiple levels. We extend each time period at the level closer to the database features by several hours before and after, and during the interval, find the related records of the host intersection of multiple levels, i.e., the related records of the identified correlated events. Thus it can target the main time periods, the participating abnormal hosts, and the corresponding abnormal types for attack modeling and data leakage scenario reconstruction. We introduce this part in Section IV-B3 in detail.
3) *Data leakage attack modeling:* We propose an automatic attack modeling method based on attack graphs. The correlated events obtained by decision fusion are visualized as directed graphs with weights, distinguishing anomaly types and reflecting causality. The attack scenarios of web data leakage incidents are obtained by finding the maximum connected subgraphs that satisfy some conditions. The connections of different anomaly stages and the basic process of data leakage incidents are presented in the visualization interface. We introduce this part in detail in Section IV-B3.
4) *Data leakage scenario reconstruction:* Based on the attack modeling results, the data leakage incident scenario is initially depicted. In addition, the relevant logs of the hosts that eventually constitute the complete attack scenario are audited in the corresponding time period. Combined with the knowledge of basic settings within the organization, it allows for a more detailed picture of the data leakage scenario, thus completing the reconstruction.

Remark. Our WeB-DLIR overcomes the challenges posed by unlabeled logs with: 1) change of focus perspective: the main subtasks we accomplish are detecting hosts related to abnormal behavior and detecting correlated events, i.e., only data related to detection content is extracted from unlabeled logs; 2) design of the methods: for web front-end payloads, it is possible to collect some payloads of positive and negative samples, making the web front-end anomaly detection in a supervised learning manner. Moreover, a pruning strategy is added to mitigate the false alarms caused by the discrepancy between the collected samples and the detected samples. For web back-end features, we use fine-grained grouping followed by unsupervised clustering. Our decision fusion and attack modeling methods are also designed to apply to unlabeled logs; 3) utilization of prior information: using basic settings within the organization or analysis of communication patterns from prior work can provide us with some guidance for unsupervised anomaly detection, decision fusion, and attack modeling. Note that these guidelines are not directly for the analyzed data, i.e., unlabeled logs.

The submodules in WeB-DLIR, i.e., the series of methods proposed in Section IV-B below, are sequentially related and correspond sequentially to the design goals described in our problem formulation. Specifically, anomaly detection identifies abnormal events at the web front and back ends, which provide analytical data for subsequent stages; decision fusion reduces redundant abnormal events and identifies correlated events, followed by attack modeling to visualize and model the discovered correlated events. In addition, our framework does not include a detailed description of multi-source log feature extraction and auditing logs when reconstructing data leakage scenarios. In general, the pre-processing steps and manual auditing phases vary by specific logs.

B. ARCHITECTURE OF WEB-DLIR

In this section, we introduce the submodules of WeB-DLIR in detail.

B.1 WEB FRONT-END ANOMALY DETECTION BASED ON BLSTM-CNN AND PRUNING STRATEGY

Since attack techniques such as SQL injection and XSS launched from the web front-end are often realized through URLs [32], this paper mainly analyzes the payloads from URLs. The basic format of URLs is `http://host:<port>/<path>?<param>`, `<port>` is 80 by default, `/<path>?<param>` is optional. SQL injection and other malicious behaviors are often characterized in the `<path>` and `<param>` parts.

As we discuss in Section II-A, we use the BLSTM-CNN model for URL anomaly detection. Specifically, the CNN mainly extracts the local features of URL text and loses some information on URL sequence, which leads to the degradation of the recognition effect. If the BLSTM is added after the CNN, it cannot fully utilize the capability of the BLSTM. Conversely, by attaching BLSTM before CNN, BLSTM acts as an encoder to generate sequence features. Each symbol in the URL sequence contains information about the symbol's context after processing, and the CNN can use a richer representation than the original input to find significant information and thus can obtain better accuracy.

The description of the BLSTM-CNN-based payload anomaly detection method is shown in Algorithm 1. It uses the *Tokenizer* function, *Pad_Sequences* function, and *Predict* function of Keras [33], and BLSTM-CNN is also built upon Keras. First, the unlabeled URLs to be detected are extracted from the web front-end logs, and the normal and abnormal URL samples collected extensively are joined into a dictionary.[1] The dictionary and the URLs to be detected are split into sequences of character-level tokens and digitally processed by the *Tokenizer* function. Then the sequence length is aligned by the *Pad_Sequences* function to obtain the training set called *Train* and the set to be detected called *Test*. A BLSTM-CNN hybrid model is constructed for *Train* training, followed by a prediction of *Test*. Finally, the web front-end dataset is returned with anomaly probabilities and is sorted from highest to lowest anomaly probability. A more detailed procedure for building BLSTM-CNN and detecting abnormal URLs is given in the Appendix, *(available online)*.

Algorithm 1. Payload Anomaly Detection Based on BLSTM-CNN

```
Input: Web front-end dataset $D_H = \{\langle tih, sip, dip, url, unum\rangle\}$; Normal URL request sample set $G_q$; Abnormal URL request sample set $B_q$; Sequence maximum length $Maxlen$.
Output: Dataset labeled with anomaly probability $D_H\_L_0 = \{\langle tih, sip, dip, url, unum, prol_0\rangle\}$.
1: $D_{HU} \leftarrow url$ in $D_H$;
2: $Dt \leftarrow G_q \cup B_q$;
3: $Seq_{Dt} \leftarrow Tokenizer(Dt)$ and $Seq_{D_{HU}} \leftarrow Tokenizer(D_{HU})$;
4:    $Train \leftarrow Pad\_Sequences(Seq_{Dt}, Maxlen)$ and $Test \leftarrow Pad\_Sequences(Seq_{D_{HU}}, Maxlen)$;
5: Building a BLSTM-CNN hybrid model called $BC$;
6: $Model \leftarrow BC(Train)$;
7: $prol_0 \leftarrow Model.Predict(Test{=}\ 0)$;
8: $D_H\_L_0 \leftarrow D_H + prol_0$;
9: Sorting $D_H\_L_0$ by $Label$ from largest to smallest.
```

Algorithm 1 can detect communication records containing malicious URLs. However, we note that in practice, there are normal scanners that frequently and extensively scan and probe important assets in the organization, to ensure that these hosts are not vulnerable [34]. The "normal" communication records during the scanning process may be identified by intelligent detection methods as "malicious" URLs, which makes the false alarm rate of the detector too high and can cause a great disruption to the completion of WeB-DLIR tasks.

[1]http://www.secrepo.com/; https://github.com/foospidy/payloads

To address the misjudgment of normal scanners and servers when identifying malicious URLs, we propose to add a pruning strategy, concluded in Algorithm 2, which can exclude normal scanners and normal scan records, making the web front-end anomaly detection more accurate. Specifically, we first group (*Gpby*) the *sips* in the top 50% of $D_H\_L_0$ after the judgment of Algorithm 1, and delete (*Drop*) those *sips* with the lowest number of *tih* communications or *dip* communications, which will not become candidate scanners. This is because the scanner has the characteristics of acting as *sips*, communicating with multiple *dips* in multiple time periods. Then use *Sum* to calculate the sum of communication *unum*, and *Uniq* to calculate the number of different communication *tih* and *dip* for each group. The statistics can be summed by assigning weights to them according to the actual organizational situation, and the *sips* with high scores are most likely to be scanners within the organization. Next, the communication records associated with these *sips* are deleted from $D_H\_L_0$ based on the number of scanners to be removed. That is, the scanner hosts and the corresponding communication hosts are not considered abnormal.

Algorithm 2. Pruning Strategy After Anomaly Detection

```
Input: Dataset labeled with anomaly probability $D_H\_L_0 = \{\langle tih, sip, dip, url, unum, prol_0\rangle\}$; Number of scanners to be pruned $DelNum$.
Output: Dataset $D_H\_L_0$ after pruning.
1: $D_{Prun} \leftarrow D_H\_L_0[prol_0 > 0.5]$;
2: $G_S \leftarrow Gpby(D_{Prun}, sip)$;
3: Initializ $D_{st}$ to keep the statistic of $G_S$;
4: for $i \leftarrow 1$ to $Len(G_S)$ do
5:     $sip_i \leftarrow G_{S_i}[sip]$, $unums_i \leftarrow Sum(G_{S_i}[unum])$, $nh_i \leftarrow Uniq(G_{S_i}[tih])$, $np_i \leftarrow Uniq(G_{S_i}[dip])$;
6:     $D_{st} \leftarrow D_{st}.append(\langle sip_i, nh_i, np_i, unums_i\rangle)$;
7: end
8: Normalize $nh, np$ and $unums$ in $D_{st}$ to 0-1;
9:     $D_{st} \leftarrow Drop(D_{st}, D_{st}[nh] = 0)$, $D_{st} \leftarrow Drop(D_{st}, D_{st}[np] = 0)$;
10: $D_{st}[score] \leftarrow D_{st}[nh] + D_{st}[np] + D_{st}[unums]$;
11: Sorting $D_{st}$ by $score$ from largest to smallest;
12: $Del_{ips} \leftarrow D_{st}[0 : DelNum][sip]$;
13: $D_H\_L_0 \leftarrow Drop(D_H\_L_0, D_H\_L_0[sip]$ in $Del_{ips})$.
```

B.2 WEB BACK-END ANOMALY DETECTION BASED ON FINE-GRAINED GROUPING CLUSTERING

As we discuss in Section II-A, we propose a fine-grained grouping clustering (FGC) strategy to detect anomalies in the database access characteristics of the web back-end. The essential motivation for our fine-grained grouping is that the database access characteristics under different groupings have large differences, making it difficult to share the same clustering metric.

Definition 5 (Distance between samples). Given a database access characteristic dataset $D_S = \{ds_1, ds_2, \ldots, ds_n\}$ with p feature attributes $\{b_1, \ldots, b_p\}$. For data points ds_i

and ds_j, define the euclidean distance metric between the two points as

$$\|ds_i - ds_j\| = \sqrt{\sum_{z=1}^{p} (ds_{iz} - ds_{jz})^2}, \tag{1}$$

where $i, j = 1, 2 \ldots, n$. As shown in Algorithm 3, the steps of the fine-grained grouping clustering for D_S are as follows.

1) Data feature processing: Divide *hour* and *day* from *tih* (*Ghour* and *Gday*); calculate the *Sum* of *num*, *sqlnum* and different day numbers *nd* for each group of $\langle sip, dip \rangle$ in D_S; use IP conversion function (*IP_Int*) to convert *sip* and *dip* into integer values; divide D_S into different subsets $\{D_{S_i}\}$ according to *dbtype* type.
2) Fine-grained grouping: For each D_{S_i}, groups are created according to ①$\langle hour, sip, dip \rangle$, ②$\langle sip \rangle$, and ③$\langle dip \rangle$, respectively, and features to be clustered are extracted. Specifically, ① is mainly concerned with the communication behavior of each pair of $\langle hour, sip, dip \rangle$, with p_1 clustering features and K_1 number of clusters. K_1 is recommended to be 2 because there are generally two types of communication behaviors: weekdays and rest days; ② and ③ are concerned with the change of the communication object of each IP, with p_2 clustering features and K_2 number of clusters. K_2 is recommended to be 1 because the normal communication objects within the organization are more fixed.
3) Clustering: Kmeans (*Km*) is utilized to cluster the feature items grouped according to ①②③ respectively. The values of each feature term are normalized to the 0-x interval before clustering. The distance between each record and its clustering center, called the outlier score (*outs*), is then calculated according to Equation (1) (referred to as a function *Gouts*). The maximum value of *outs* for records with p clustering features is less than $\sqrt{px^2}$.
4) Abnormal score calculation: For ①, the anomaly score is equal to *outs*, while the anomaly score of ②③ is *outs* divided by the different communication days (*nd*) of each $\langle sip, dip \rangle$ group to reduce the anomaly score of fixed communication objects, i.e., normal hosts. Then assign ① and ②③ weights as ω_1, $\omega_a \times \omega_2$ respectively, and calculate the anomaly score (*abs*) of each record of D_{S_i} according to the Equation (2). Finally, integrate each D_{S_i} to get the anomaly score of each record in D_S.

The anomaly score for records in each D_{S_i} is calculated as

$$\begin{aligned} D_{S_i}[abs] =& \omega_1 \times G_1[outs] + \omega_a \omega_2 \times (G_2[outs]/G_2[nd]) \\ &+ \omega_a \omega_2 \times (G_3[outs]/G_3[nd]), \end{aligned} \tag{2}$$

where ω_a is calculated as

$$\omega_a = \sqrt{|p_1| x^2} / (\sqrt{|p_2| x^2 / nd_{min}}), \tag{3}$$

where nd_{min} is the minimum number of communication days among all $\langle sip, dip \rangle$ in D_S. Since we divide by nd when calculating the G_2 and G_3 anomaly scores, we construct ω_a this way in order to keep the anomaly metric range of G_1 and G_2, G_3 as close as possible before weighting.

Note that after the fine-grained grouping, the number of clusters is small, thus reducing the time to find and determine the number of clusters and cluster centers. The administrator can easily give K_1, K_2 and ω_1, ω_2 depending on the underlying settings within the organization, so our fine-grained grouping method is K-free in this case, which means FGC doesn't need to bother with choosing K.

Algorithm 3. Anomaly Detection of Database Access Characteristics Based on Fine-Grained Grouping Clustering

```
Input: Web back-end dataset $D_S$; Number of clusters and
       weights for each group $K_1, K_2, \omega_1, \omega_2$.
Output: Dataset labeled with anomaly probability $D_S\_L_0$.
 1: $D_S[hour] \leftarrow Ghour(D_S[tih]), D_S[day] \leftarrow Gday(D_S[tih])$;
 2: $G_{sd} \leftarrow Gpby(D_S, \langle sip, dip \rangle)$;
 3: for $i \leftarrow 1$ to $Len(G_{sd})$ do
 4:   $D_S[\langle sip, dip \rangle = G_{sd_i}][nums, sqlnums] \leftarrow Sum(G_{sd_i}[num, sqlnum])$;
 5:   $D_S[\langle sip, dip \rangle = G_{sd_i}][nd] \leftarrow Uniq(G_{sd_i}[day])$;
 6: end
 7: $D_S[sip], D_S[dip] \leftarrow IP\_Int(D_S[sip], D_S[dip])$;
 8: $\{D_{S_i}\} \leftarrow Gpby(D_S, dbtype)$;
 9: Initialize $D_S\_L_0$ to keep $D_{S_i}$ with calculated anomaly
    scores;
10: for $i \leftarrow 1$ to $Len(\{D_{S_i}\})$ do
11:   $G_1 \leftarrow Gpby(D_{S_i}, \langle hour, sip, dip \rangle))$;
12:   $G_2 \leftarrow Gpby(D_{S_i}, sip)), G_3 \leftarrow Gpby(D_{S_i}, dip))$;
13:   for each $G_{1_z}, G_{2_j}, G_{3_k}$ in $G_1, G_2, G_3$ do
14:     if $Len(G_{1_z}) < K_1$ then $K_1 = 1$, $G_{1_z}[outs] \leftarrow Gouts$
        $(Km\ (K_1, G_{1_z}[p_1]))$;
15:     if $Len(G_{2_j}) < K_2$ then $K_2 = 1$,$G_{2_j}[outs] \leftarrow Gouts$
        $(Km(K_2, G_{2_j}[p_2]))$;
16:     if $Len(G_{3_k}) < K_2$ then $K_2 = 1$, $G_{3_k}[outs] \leftarrow Gouts$
        $(Km(K_2, G_{3_k}[p_2]))$;
17:   end
18:   Calculate $D_{S_i}[abs]$ according to (2);
19:   $D_S\_L_0 \leftarrow D_S\_L_0.append(D_{S_i})$;
20: end
21: Sorting $D_S\_L_0$ by $abs$ from largest to smallest.
```

B.3 DFADR AND ATTACK MODELING

By correlating and fusing the output results of web front-end and back-end anomaly detection, the abnormal time period of web data leakage incidents and closely related abnormal hosts can be effectively extracted. Therefore, we propose an approach for decision fusion of anomaly detection results (DFADR).

The description of the DFADR method is shown in Algorithm 4. In order not to miss any abnormal hosts, we first use the known information of the actual abnormal hosts in the front-end and back-end to find the discriminant threshold that can make the abnormal host detection rate reach 1 from $D_H\_L_0$ and $D_S\_L_0$, respectively. Then the datasets DR_H and DR_S, consisting of records under the discriminant thresholds, are used as the analysis data for DFADR. Both DR_H and DR_S only keep the four information $\langle tih, sip, dip, num\rangle$ for correlation analysis. As our objective is the analysis of database leakage incidents, DFADR takes the anomaly detection results at the level of database access characteristics as the core. That is, the web back-end abnormal behavior is necessary to constitute a correlation event. We analyze each non-repeating time period t_i of DR_S (by *Dedup*) and the corresponding record DR_{S_i}, and use the range of τ hours before and after t_i to find the corresponding records DR_{H_i} in DR_H. Keep the records with IP intersection in DR_{H_i} and DR_{S_i}, denoted as DF_{H_i} and DF_{S_i}, where IP intersection IP_{inters} is defined as

$$\begin{aligned}IP_{inters} = (DR_{H_i}[sip]\cap DR_{S_i}[sip]) \cup (DR_{H_i}[sip]\cap DR_{S_i}[dip])\\ \cup(DR_{H_i}[dip]\cap DR_{S_i}[sip]) \cup (DR_{H_i}[dip]\cap DR_{S_i}[dip]),\end{aligned} \tag{4}$$

Finally, each IP and the *Sum* of *num* (*nums*) are extracted from DF_{H_i} and DF_{S_i}, and *nums* is normalized as the anomaly score.

Algorithm 4. Decision Fusion of Anomaly Detection Results (DFADR)

```
Input: The detected abnormal result datasets $DR_H$ and $DR_S$,
       with $\langle tih, sip, dip, num\rangle$ format; the extended time range $\tau$.
Output: Abnormal host and abnormal probability of correlated events $DF_{ip\_ns}$;
        the correlated event-related log records $D_{fb}$.
1: Initialize $DF_H$ and $DF_S$ to save records with correlated events;
2: for $t_i$ in $Dedup(DR_S[tih])$ do
3:   $DR_{H_i} \leftarrow DR_H[t_i - \tau \le tih \le t_i + \tau]$;
4:   $DR_{S_i} \leftarrow DR_S[tih = t_i]$;
5:   $DF_{H_i} \leftarrow DR_{H_i}[sip$ or $dip$ in $IP_{inters}]$;
6:   $DF_{S_i} \leftarrow DR_{S_i}[sip$ or $dip$ in $IP_{inters}]$;
7:   $DF_H \leftarrow DF_H.append(DF_{H_i})$;
8:   $DF_S \leftarrow DF_S.append(DF_{S_i})$;
9: end
10: $DF_{ip\_ns}[IP, nums] \leftarrow DF_H[sip, Sum(num)] \cup DF_H[dip, Sum(num)] \cup DF_S[sip, Sum(num)] \cup DF_S[dip, Sum(num)]$;
11: $D_{fb}[tih, sip, dip, num, tp] \leftarrow DF_H[tih, sip, dip, num, "tp_f"] \cup DF_S[tih, sip, dip, num, "tp_b"]$;
12: Normalize $DF_{ip\_ns}[nums]$, $D_{fb}[num]$.
```

The DFADR algorithm can correlate the anomaly detection results at each level to eliminate anomaly redundancy, and can further identify abnormal time periods and abnormal hosts that are closely related to web-based data leakage incidents, providing basic support for attack modeling. Moreover, abnormal events with correlation in multiple levels can be found by taking the abnormal time of the latter level as an extension point. Based on this idea, our DFADR can be extended to the fusion of anomaly detection results at multiple levels, which enhances the scalability of WeB-DLIR.

Algorithm 5. Attack Modeling

```
Input: Abnormal host and abnormal probability of correlated events
       $DF_{ip\_ns} = \{\langle IP, nums\rangle\}$; the correlated event-related log records
       $D_{fb} = \{\langle tih, sip, dip, num, tp\rangle\}$.
Output: Attack Graph.
1: for $i \leftarrow 1$ to $Len(DF_{ip\_ns})$ do
2:   $ip_i \leftarrow DF_{ip\_ns_i}[IP]$, $sip_i \leftarrow DF_{ip\_ns_i}[IP]$;
3:   $size_{n_i} \leftarrow DF_{ip\_ns_i}[nums]$, $color_{n_i} \leftarrow Col_1(ip_i)$;
4:   Initialize $Ldip_i, Lsize_i, Ltih_i, Lcolor_i$ as lists;
5:   for $j \leftarrow 1$ to $Len(D_{fb})$ do
6:     if $D_{fb_j}[sip] = ip_i$ then
7:       if $D_{fb_j}[dip]$ in $Ldip_i$ then
8:         $pos = Ldip_i.index(D_{fb_j}[dip])$;
9:         $Lsize_i[pos] \leftarrow Lsize_i[pos] + D_{fb_j}[num]$;
10:        $Ltih_i[pos] \leftarrow Ltih_i[pos] + '','' + D_{fb_j}[tih]$;
11:      end
12:      else
13:        $Ldip_i \leftarrow Ldip_i.append(D_{fb_j}[dip])$, $Ltih_i \leftarrow Ltih_i.append(D_{fb_j}[tih])$, $Lsize_i \leftarrow Lsize_i.append(D_{fb_j}[num])$, $Lcolor_i \leftarrow Lcolor_i.append(Col_2(D_{fb_j}[tp]))$;
14:      end
15:    end
16:  end
17:  $node_i \leftarrow \langle ip_i, size_{n_i}, color_{n_i}\rangle$;
18:  $link_i \leftarrow \langle sip_i, Ldip_i, Lsize_i, Ltih_i, Lcolor_i\rangle$;
19: end
20: Drawing attack graphs based on force-directed algorithms.
```

After decision fusion, to visualize the time, process, and abnormal host situation of the attack and reconstruct the data leakage scenario, we propose an attack graph-based modeling method that can automatically build a visualization with rich information based on the results obtained from the previous anomaly detection and decision fusion.

Definition 6 (Attack Graphs). The attack graph of a data leakage incident consists of nodes=$\{\langle ip, size_n, color_n\rangle\}$ *and connection lines links=*$\{\langle sip, dip, size_l, tih, color_l\rangle\}$*. As for the links, the direction of the arrow distinguishes sip and dip, and the label on the connection line carries the tih set of communication. The* $size_n$ *is mapped to the size of the node token, which indicates the size of the abnormal traffic,* $color_n$ *is used to distinguish between intranet and extranet IP;* $size_l$ *is mapped to the thickness of the line segment, i.e., the weight, which indicates the abnormal traffic between sip and dip within tih, and* $color_l$ *indicates whether the abnormal level is web front-end or web back-end.*

The process of automatically constructing the attack grid graph is shown in Algorithm 5. First, each IP is extracted

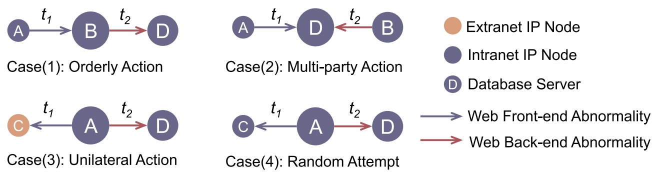


FIGURE 3. **Correlation event types in attack graphs ($t_1 \leq t_2$).**

from DF_{ip_ns} as sip_i and the corresponding $nums_i$ as $size_{n_i}$, and then mapped to $color_{n_i}$ (Col_1 function) according to the network segment of the IP address. Corresponding to sip_i, we extract lists $Ldip_i$, $Lsize_i$, $Ltih_i$, and $Lcolor_i$ from D_{fb}, where num with the same dip is accumulated as $Lsize_i$, tih with the same dip is spliced as $Ltih_i$, and $Lcolor_i$ is obtained by mapping the tp types to different colors (Col_2 function). Finally, the obtained nodes and links are visualized based on the force-oriented graph algorithm, which is a popular relational graph layout that supports interactivity [35].

After constructing the basic attack graph, we define the types of correlation events present in the attack graph as follows.

Definition 7 (Correlation event types in attack graphs). With the assumption that the database server would not initiate requests, four fundamental types of correlation events in the attack graph can be classified (Figure 3). `Case(1)` *is a load-injected B making database access to D, which is a sequential correlation;* `Case(2)` *is multiple attackers or multiple hosts under attack teaming up to attack the database;* `Case(3)` *is attacker A communicates abnormally with extranet host C and then accesses the database server abnormally;* `Case(4)` *is a random attempt or undifferentiated attack by a single attacker A, which does not form a strong association.*

As described in Section III, a data leakage incident includes complex multiple correlation steps, so we regard the maximum connected subgraph within the `Case(1),(2),(3)` condition in the attack graph as the attack scenario of the data leakage incident. The attack garph constructs a scenario from the perspectives of abnormal time sequences, participating abnormal hosts, and abnormal event causality in different attack phases. Specifically, directed edges with the concept of time span and IP information of the starting and ending hosts represent the communication relationship among abnormal hosts, and different abnormal types are identified by the color of line segments. In addition, the problem of finding attack scenarios for data leakage incidents is finally formalized as finding the maximum connected subgraph that satisfies the conditions. Note that after anomaly detection and decision fusion, the final attack graph is more concise, and hence the data leakage scenario can be found quickly after much less disturbance.

V. EVALUATION

In this section, we evaluate the performance of WeB-DLIR.

A. COMPARATIVE ANALYSIS

Here, we compare our WeB-DLIR with the existing works related to web back-end database leakage [1], [5]–[7] and attack incident scenario reconstruction [8], [9], as there are currently no studies closely related to web back-end database leakage incident reconstruction.

In the web back-end database leak detection (short for DLD) research, Costante et al. [1] detected potential data leakage by detecting anomalies in database transactions and introduced a feedback mechanism that was able to reduce the number of false positives. However, the scheme of [1] is unable to detect anomalies at multiple levels and cannot handle unlabelled data, in contrast to our WeB-DLIR. Different from [1], Li et al. [5] designed several analyzers for multi-source security logs, which include anomaly identification in an unsupervised method, but a lot of feature extraction work is required. Moreover, for both [1], [5], the methods to reduce the false positive rate are based on feedback from analysts or experts, while WeB-DLIR is automated through a decision fusion algorithm. Furthermore, the DLD approaches do not focus on attack modeling and scenario reconstruction. In web back-end database leakage forensics (short for DLF) research, Latib et al. [6] provided an investigation result for web intrusion in a Big Data environment by analyzing web logs. Yu et al. [7] visualized the propagation path of sensitive data. However, they both studied just a part of the forensic work, while our WeB-DLIR is more comprehensive. In terms of attack incident reconstruction (short for AIR), Hossain et al. [8] used multiple strategies for attack detection and were able to avoid some false positives. In [8], the attack modeling can support the scenario reconstruction, but it does not construct a more intuitive and clear topological map of the scenario as our WeB-DLIR does. Pei et al. [9], on the other hand, proposed a multi-stage log-based intrusion analysis system to reconstruct attack behaviors to discover attack communities with a low positive rate. However, [8] and [9] have a common drawback that cannot process unlabeled data. In WeB-DLIR, we perform different levels of anomaly detection on unlabeled multi-source logs and achieve data leakage incident scenario reconstruction (short for DLIR) based on attack modeling. We also pay attention to reducing the false alarm rate and remark that our WeB-DLIR can be extended to multi-level anomaly detection. A comparative summary of these works is shown in Table 2.

It shows that there are currently no solutions that can be directly applied to our application situations and no related works available for experimental comparison. Therefore, in the following experiments, when implementing the WeB-DLIR submodules, methods that can be applied to solve the problems of these submodules are taken for comparison, including popular machine learning techniques or existing schemes.

B. EXPERIMENTAL DATASETS AND METRICS

B.1 DATASETS

The datasets that can be used to evaluate our proposed WeB-DLIR should satisfy the following conditions:

1) The datasets should contain web-related multi-source log files during web back-end database leakage incidents.

TABLE 2. **Comparative summary.**

Category/Function	[1]	[5]	[6]	[7]	[8]	[9]	Ours
Task	DLD	DLD	DLF	DLF	AIR	AIR	DLIR
Analyzed Data	Database Transactions	Multi-source Security Logs	Web Logs	Sensitive Data	Attack Datasets	Multiple System Logs	Multi-source Web-related Logs
Process Unlabeled Data	×	✓	×	×	×	×	✓
Anomaly Detection	✓	✓	✓	×	✓	×	✓
Level of Anomaly Detection (if Any)	One	Multiple	One	N.A.	Multiple	N.A.	Multiple
Reduce False Alarms	✓	✓	×	×	✓	✓	✓
Attack Modeling	×	×	×	✓	✓	×	✓
Support Scenario Reconstruction	×	×	×	×	✓	✓	✓
Available for Experimental Comparison with Ours	×	×	×	×	×	×	N.A.

'N.A.' denotes not comparable.

2) The datasets may be unlabeled but should have a score or metric for reconstructing the final data leakage scenario.

In the field of cybersecurity, to the best of our knowledge, there are currently no known public datasets that meet both of the above conditions. Therefore, we choose log records from an enterprise during a web back-end data leakage incident, which can be obtained through a competition held by CCF.[2] The log files include data records of header in HTTP requests (Flow) and traffic records of database login and operation execution (Login and Db). The details of the obtained original compressed files are shown in Table 3.

Pre-Processing. For the mentioned logs, all files are first converted to CSV format uniformly and then stored in the Mysql database by Flow, Login and Db categories for analysis. We use database commands to exclude logs with missing important fields (e.g., URL) to prevent data flooding. As the attack model described in Section III-A, the data leakage occurs around the servers inside the organization, so we keep records containing all server IP addresses (487 IPs), which is known from the Preliminary Round of the competition. Further, we extract the dataset (D_H) containing web front-end payload features from the Flow records remained, and the dataset (D_S) of web back-end database access features from the Login and Db records remained. Specifically, we extract *tih*, *sip*, *dip*, *url*, *unum* from Flow logs and *tih*, *sip*, *dip*, *num*, *sqlnum*, *dbtype* (here are *login* or *db*), number of different communication source ports (*difsport*), password (*difpw*), information (*difinfo*), username (*difuser*), and database access information (*difsqlinfo*) for each $\langle tih, sip, dip \rangle$ from Login and Db logs. The details of the obtained experimental dataset are shown in Table 4.

B.2 METRICS

Corresponding to the detection tasks and modeling task defined in Section III-B, the experimental evaluation is divided into two parts: 1) In the web front-end and back-end anomaly detection task, the main evaluation is on the identification effect of abnormal hosts at the corresponding level; in the anomaly detection result decision fusion task, the main evaluation is the recognition effect of abnormal hosts closely related to the data leakage incident; 2) In attack scenario modeling and data leakage incident scenario reconstruction, the main evaluation is the intuitive rendering ability of attack graph and the degree of reconstruction of the data leakage incident.

Since the original datasets are unlabeled, the above two evaluations are mainly based on the expert scoring of the final results of the competition proposal.[3] In order to quantify the evaluation metrics, based on the final results, we give in Table 5 the number of normal and abnormal hosts and abnormal time periods for the front and back ends and the data leakage incident.[4] Particularly, there are many abnormal events at the web front and back ends, and there are no centralized abnormal time periods, while the number of normal hosts for data leakage incidents depends on the detection results at the web front and back ends.

The first part of the evaluation can be viewed as a binary classification task. We now define *TP*, *FN* and *TN*, *FP* as follows: the actual abnormal hosts are classified as abnormal hosts, normal hosts; the actual normal hosts are classified as normal hosts, abnormal hosts. From this, the detection rate, i.e., Recall (R) or True Positive Rate (TPR), Precision (P), and False Positive Rate (FPR) can be defined as shown in Equation (5). Since abnormal hosts account for only a small fraction of the hosts, we plot *TPR-FPR* curves (also called ROC) to measure the overall effect of classification and the Precision-Recall curves (PR) to measure the effect of identifying the imbalance category, which are plotted by varying parameters such as the anomaly ratio. We then define AUC-ROC and AUC-PR as the area under the curves.

$$\begin{cases} R = TPR = \frac{TP}{TP+FN} \\ P = \frac{TP}{TP+FP} \\ FPR = \frac{FP}{TN+FP} \end{cases}. \quad (5)$$

[2]Available at https://www.datafountain.cn/competitions/358. We used the data provided by the Semi-final round

[3]https://discussion.datafountain.cn/questions/2252/answers/23376

[4]More specific information such as the host IP can be found in our source code, available at https://github.com/Cristliu/WeB-DLIR

TABLE 3. **The details of the original compressed files.**

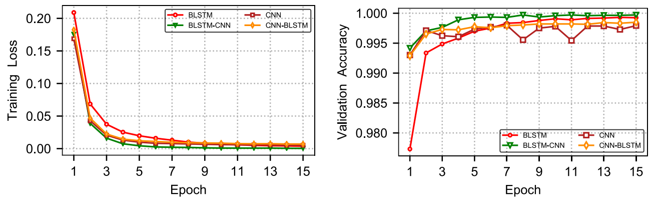
Data	Number of Files	Record	Size
Flow	40	23,834,302	1.06 GB
Login	2	10,027,010	176 MB
Db	2	3,878,893	21.9 MB

TABLE 4. **Experimental data set details.**

Data	Record	Feature	Num. of hosts	Size
D_H	993,923	$tih, sip, dip, url, unum$	867	157 MB
D_S	1630	$tih, sip, dip, num, sqlnum, dbtype, difsport, difpw, difinfo, difuser, difsqlinfo$	35	118 KB

TABLE 5. **The correct results.**

Type	Front-end	Back-end	Data Leakage Incident
Num. of Nor. Hosts	841	25	N.A.
Num. of Abnor. Hosts	26	10	7
Abnormal Time	N.A.	N.A.	June 11, 18:00-22:00



(a) Training Loss (b) Validation Accuracy

FIGURE 4. **Determination of epoch.**

C. EXPERIMENTAL SETTINGS AND RESULTS

First, anomaly detection is performed for D_H and D_S respectively, and widely used or representative anomaly detection models are experimentally compared. Then, the DFADR method is performed on anomaly detection results to verify the necessity and effectiveness of correlation analysis and decision fusion. Finally, the attack graph realizes the modeling of the attack scenario, and then we analyze the relevant log data of the attack graph to realize the reconstruction of the data leakage incident.

We trained and tested models for the web front-end level on a workstation with a Tesla P100-PCIE GPU and Intel(R) Xeon(R) CPU E5-2620 v4 CPU @2.10 GHz, and the rest of the experiments were performed on the same local machine (lntel(R) Core(TM) i5-9500F CPU @3.00 GHz; 8 GB RAM), and all methods are implemented in Python.

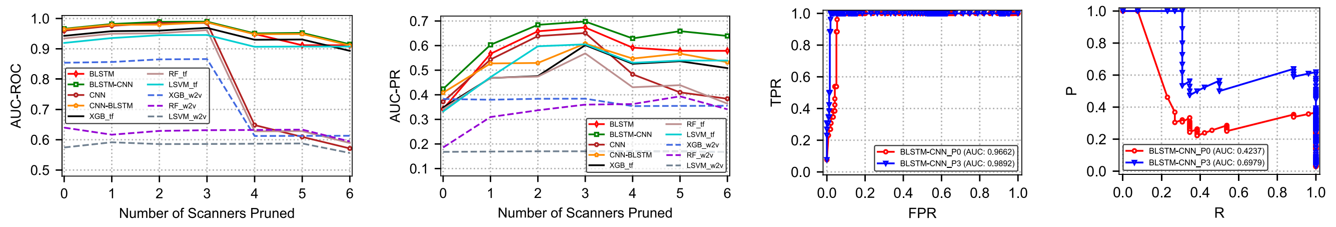
C.1 ANOMALY DETECTION RESULTS

(1) Web Front-End Payload Anomaly Detection. BLSTM-CNN is compared with six methods, including BLSTM, CNN, and CNN-BLSTM, which are also families of neural networks. As well as the widely used SVM model, Random Forest (RF), an ensemble learning method based on bagging that achieved the best classification performance in [36], and XGBoost (XGB), used by Chen et al. [15], which uses an enhanced ensemble learning technique.

Settings. The neural network models are implemented using Keras. We choose Adam, an efficient optimizer capable of adapting the learning rate, as the optimizer [37]. We use the default parameter of Adam provided by Keras, i.e., learning rate $lr = 0.001$, which follows the value provided in [37]. The loss function used is *binary_crossentropy*, as it is more suitable for binary classification problems. Specifically, the size of the embedding layer is 200×100, and the memory cells in LSTM are set to 64, which means the output dimension is 200×128 through the bidirectional LSTM. 128 filters are used in the convolutional layer and the length of the convolutional window is taken as 3. The batch size is set to 64 and the dropout is set to 0.5 to prevent overfitting. The other BLSTM, CNN, and CNN-BLSTM parameters are similar to BLSTM-CNN.[5] We divide 10% from the training set as the validation set and then determine the appropriate epochs by the variation of the loss function and validation accuracy with epochs. As shown in Figure 4, the loss of training set and accuracy of validation set converge gradually around the 7th epoch and fluctuate after that, so we set the epoch to 7. The remaining comparison models are implemented based on the Scikit-learn package and Gensim package. The SVM uses the "Linear" kernel function, and other kernel functions such as "Gaussian" functions are discarded because of too long running time. Depending on the difference between using TfidfVectorizer and Word2Vec methods to vectorize the URLs representation for extracting the feature matrix, we denote the comparison models as XGB_tf, RF_tf, LSVM_tf and XGB_wv, RF_wv, LSVM_wv. We determine the parameters that work better by convergence or by their effectiveness on the validation set.

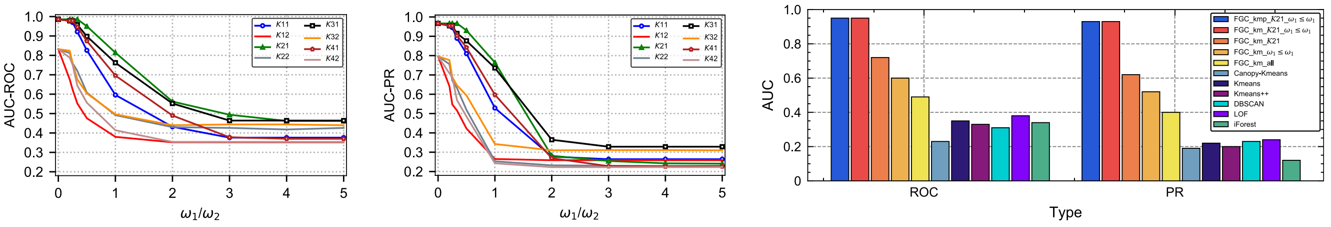
To evaluate the effectiveness of Algorithms 1 and 2, we compare the AUC-ROC and AUC-PR of all models with the different number of scanners pruned in Figures 5(a) and 5(b). We can see that: 1) BLSTM-CNN has the best performance in the front-end anomaly detection, other neural network models are the next. If a BLSTM is attached after the CNN has extracted local information, it is just equivalent to attaching a fully connected layer, which is less effective than preferentially attaching a BLSTM capable of extracting sequence information in front of the CNN [20]. The results of RF, XGB, and LSVM implemented based on the TfidfVectorizer method are generally better than those implemented based on Word2Vec. Word2Vec makes the models based on it less efficient because of more complex parameter setting; 2) The

[5]More detailed model settings can be found in the WeB-DLIR/1_Front_End/images folder of our source code



(a) AUC-ROC with different pruning numbers (b) AUC-PR with different pruning numbers (c) ROC curve comparison of BLSTM-CNN (d) PR curve comparison of BLSTM-CNN

FIGURE 5. Evaluation of web front-end anomaly detection.



(a) AUC-ROC with K_1, K_2, ω_1, ω_2 (b) AUC-PR with K_1, K_2, ω_1, ω_2 (c) AUC comparison

FIGURE 6. Evaluation of web back-end anomaly detection.

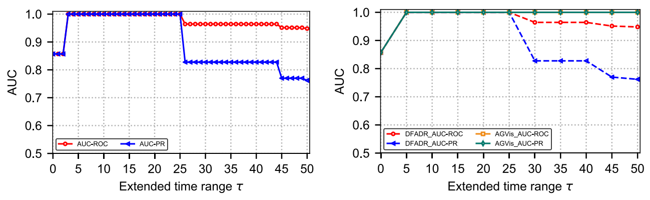
AUC-ROC and AUC-PR of most models increase and then decrease with the number of prunings, and the highest AUC value is taken at $DelNum = 3$. Therefore, the number of scanners in D_H is most likely to be 3. When $DelNum \leq 3$, the pruning strategy can delete misidentified scanner-related records and thus increase the AUCs. But after that, the pruning strategy removes too many abnormal hosts, resulting in the failure to reach $R = TPR = 1$, and thus the AUC values decrease; 3) The anomaly detection results at the web front-end payload level also have a low AUC-PR, mainly because the unlabeled data to be detected in the actual environment is more complex than the training set we constructed, indicating that other levels of anomaly detection need to be correlated to more accurately analyze data leakage incidents.

We further compare the ROC and PR curves without pruning strategy (P0) and when $DelNum = 3$ (P3). As shown in Figures 5(c) and 5(d), the increase in AUCs brought by the pruning strategy is mainly in terms of lower FPR and higher P. It shows that a proper pruning strategy can reduce the false alarm rate of anomaly detection, improve the performance of the model, and reduce the subsequent workload on false alarm troubleshooting.

(2) Web Back-End Database Access Anomaly Detection. FGC is compared with six unsupervised methods widely used for anomaly detection, including the baseline model Kmeans, Kmeans++, Canopy-Kmeans which can determine the initial centroids and the number of clusters for Kmeans to improve the efficiency of clustering [38], the density clustering-based DBSCAN used by Ramachandran et al. [39] for clustering to detect database intrusions, the local outlier factor (LOF) algorithm applied by Kim et al. [40] to capture anomalous database access logs, and the ensemble-based outlier detection method iForest used by Gavai et al. [41] to identify anomalous events.

Settings. The implementation of FGC is detailed in Section IV-B2. We use the normalization interval 0-1 and then discuss the impact of K_1, K_2, ω_1 and ω_2 on the effect of FGC, and we also compare the difference between Kmeans and Kmeans++ as our clustering methods in FGC. The rest of the methods are implemented based on the Scikit-learn package. For Kmeans, Kmeans++, and Canopy-Kmeans, we use the same anomaly score measure as FGC, and the K value for Kmeans and Kmeans++ is determined based on the elbow method. DBSCAN, LOF, and iForest discriminate outliers as abnormal records.

The comparison results are shown in Figure 6. In Figures 6 (a) and 6(b), we compare the AUCs of FGC under different K_1, K_2 combinations using the ratio of ω_1 to ω_2 as the horizontal coordinate, where the abbreviation $K11$ is used to represent $K_1 = 1$ and $K_2 = 1$. Other K_1, K_2 combinations are not plotted because the effect is lower than either of the combinations shown in the figure. We can see that: 1) better results can be obtained when $K_1 \geq K_2$ and the best results are obtained when $K_1 = 2$ and $K_2 = 1$. It shows that the $\langle tih, sip, dip \rangle$ communication mode indicated by K_1 is mainly divided into two categories, while the host communication mode indicated by K_2 is more fixed within the enterprise; 2) the value of AUCs decreases as the ratio of ω_1/ω_2 increases. It shows that the value of ω_2 should be larger than ω_1, that is, for this case, the host communication mode is more important than the $\langle tih, sip, dip \rangle$ communication mode. As shown in Figure 6 c: 1) Using Kmeans++ (kmp) or Kmeans (km) as



(a) AUCs of DFADR with τ

(b) AUC comparison of DFADR and after attack graph visualization

FIGURE 7. Evaluation of DFADR and attack graph.

our base clustering method has no effect on the AUC values, and our choice of FGC_km reduces the time efficiency due to the extra processing time required by Kmeans++; 2) Although other models can complete anomaly detection in a short time, they are even less effective than the average effect of our FGC under unknown conditions (*FGC_km_all*) due to coarse-grained detection and more complex parameter settings. In particular, Canopy-Kmeans without guidelines is worse than Kmeans with K determined by the elbow method because of the uncertainty in parameter selection.

From the above discussion, it can be observed that our FGC algorithm can obtain the optimal anomaly detection model within a very small range of clusters by fine-grained grouping. Moreover, in practical applications, the values of K_1, K_2, ω_1, and ω_2 can be easily given by the administrator within the organization, and the average effect of FGC can reach AUC > 0.9 when the administrator knows the basic settings of $K_1 = 2$, $K_2 = 1$ and $\omega_1 \leq \omega_2$.

C.2 DECISION FUSION RESULTS AND ATTACK GRAPH

In data leakage incident analysis, the abnormal host information at the front and back end of the web can be regarded as known prior information. Therefore, the detection results to be decision fused come from the BLSTM-CNN with $DelNum = 3$ strategy and the $FGC\_km\_K21\_\omega_1 \leq \omega_2$ strategy that can achieve better results at each level. As described in Section IV-B3, in order not to miss abnormal hosts, we use the abnormal records DR_H and DR_S under the discriminant threshold of reaching $R = 1$ at the respective levels as the analysis data for DFADR. After decision fusion, the attack graph is automatically visualized based on Algorithm 5.

Settings. For τ, the only parameter of DFADR, we discuss the effect of decision fusion when τ=0-50. When τ is close to 0, DFADR is more likely to detect data leakage incidents that are concentrated in a very short time, while when τ is larger, it is more suitable for more complex data leakage incidents of longer duration. For force-oriented layout-based attack graphs, we use the pyecharts[6] tool to build and output the scene graph in HTML format.

As shown in Figure 7(a), both AUC-ROC and AUC-PR rise and then fall with τ. Figure 8 shows the attack graphs when τ is 1, 3, 50. To save space, we give a compact version of the visualization view. The colors and thicknesses of the attack graph nodes and connection lines follow the description in Definition 6. In particular, the text labels attached to the connection lines indicate the abbreviation of the communication *tih*, e.g., the 11th 19 h to 20 h is indicated as 11d19. We observe that before $\tau = 3$, the AUCs fail to reach 1 because the extended time range is not enough to identify all the abnormal hosts associated with data leakage incidents, i.e., R and TPR cannot reach 1. After that, the AUCs remain at 1 for a long time, which indicates that there are no omissions and no misclassifications in the interval; when the value of τ is larger, there is a decrease in the identification effect at 26, 45, and 50 respectively, which is due to more false positives, i.e., the events that are unrelated to the data breach but have a front and back-end correlation are identified.

The above analysis illustrates that there are misclassification scenarios when τ is selected improperly, which makes it necessary to continue the troubleshooting audit after decision fusion. However, benefitting from the intuitive presentation of the attack graph and the correlation event types in the attack graph described in Definition 7, we can quickly find the data leakage event scenario in the visual view. For the case of $\tau = 50$ (shown in Figure 8(c)), the connections of 10.49.231.206 and 10.49.253.194 do not satisfy the condition of $t_1 \leq t_2$ and do not belong to the category of `Case (1)-(4)`; the subgraph formed by the three connections from 10.56.148.238 belongs to the category of `Case(4)`, but since they all emanate from the same host and do not communicate with extranet hosts, they do not form a strong correlation event. On the other hand, the eligible maximum subgraph of Figure 8(c) is the part with shading on the left, so we can exclude the other misidentified hosts. Therefore, when $\tau > 3$, the final data leakage incident can also be locked by visually troubleshooting through the attack graph. That is, in Figure 7(b), when $\tau \geq 3$, both AUC-ROC and AUC-PR can reach 1.0 after the attack graph visualization (AGVis).

Furthermore, in order to compare and highlight the necessity of decision fusion for the identification of abnormal hosts in data leakage incidents, we use the abnormal hosts of data leakage to measure the results of single-level and decision fusion. That is, we evaluate the BLSTM-CNN with a pruning strategy for the web front-end (WFDR) and the *FGC_km* strategy for the web back-end (WBDR) using the same 7 hosts from the final data leakage incident as abnormal hosts. Here we assume that the parameters of both web front-end and back-end are optimal under the guidance of the administrator, while DFADR may get the worst results due to the lack of experience in selecting τ. The identification results of abnormal hosts in data leakage are shown in Table 6, and we can see that: 1) Decision fusion for the data leakage incident abnormal host identification is significantly better than the anomaly detection method with isolated web front-end and back-end, especially in AUC-PR; 2) the reason for achieving higher AUC-ROC of WFDR is that there are many normal

[6]https://github.com/pyecharts/pyecharts/

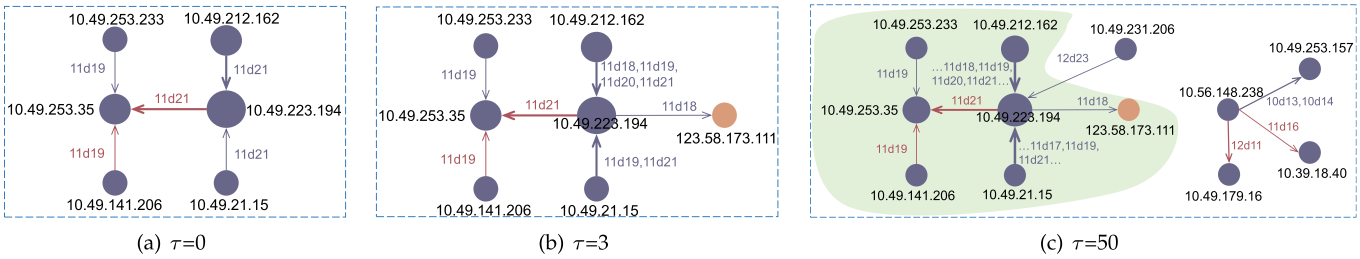


FIGURE 8. Attack graphs with different τ.

TABLE 6. Comparation of DFADR and WFDR, WBDR.

Level	Num. of Hosts	AUC-ROC	AUC-PR
DFADR_τ0	51	0.8517	0.8517
DFADR_τ50	51	0.9481	0.7620
WFDR	867	0.8460	0.5092
WBDR	35	0.4286	0.4286

hosts at the web front-end level, and WFDR achieves better results in most of the normal hosts leading to the improvement of the overall AUC-ROC, but the identification results of the unbalanced category, i.e., abnormal hosts, are poorer, which is reflected in the AUC-PR; 3) for WBDR, the AUCs are less than 0.5, which is due to the fact that many abnormal hosts of data leakage incidents do not appear in the web back-end logs.

Discussion. Abnormal hosts in data leakage incidents may appear only in web front-end logs or in web back-end logs, e.g., some web servers have only front-end communication logs, while some database servers' communication records appear only in Db logs. It means that measuring single-level detection results with abnormal hosts in data leakage incidents, even using a ”perfect” classifier at single-level detection, is impossible to achieve 100% correct evaluation metrics. On the other hand, it explains the urgent need for multi-level anomaly detection and decision fusion steps in analyzing data leakage incidents. It is more important to detect abnormal events that are correlated at the web front and back ends when making decision fusion. With the same abnormal hosts as the correlation, it can diffuse to find the cluster of hosts for the data leakage incident.

In summary, more accurate detection results at each level will inevitably provide more favorable support for decision fusion and attack modeling. Moreover, in practical data leakage incident analysis, integrated web front and back ends and even other levels for correlation fusion are necessary and effective.

C.3 DATA LEAKAGE INCIDENT RECONSTRUCTION

Based on the attack graph and its properties, it can be seen from Figure 8 b: the time period of the data leakage incident is between 18:00-22:00. By further auditing the relevant abnormal host log data, reviewing the basic settings within the enterprise or analyzing the host communication patterns, the administrator can know that: 10.49.21.15 is a legitimate internal enterprise scanner, 10.49.253.233 is a web proxy server, 10.49.141.206 is a server, 10.49.212.162 is a client host, 10.49.223.194 is a web server, and 10.49. 253.35 is the database server. In addition, it can also find communication logs that do not show web front-end or web back-end anomalies, but nevertheless form a correlation with data leakage incidents, such as the communications with client host 10.56.144.126. Then, the web back-end database data leakage scenario is reconstructed from the attack graph and audit results, which occurred on June 11, 2019, and the detailed scenario reconstruction topology is shown in Figure 9. The basic evolution of this web data leakage is as follows.

1) The attacker manipulated the proxy server 10.49.253.233 to perform a scanning probe on the 10.49.253 network segment at around 19:00 (step i.).
2) After obtaining the vulnerability information of the internal enterprise server, 10.49.212.162 executed SQL injection and other attacks on the web server 10.49.223.194 in 18:00-22:00 (step ii.). During this period, the enterprise external network was also accessed to obtain scanning and cracking tools.
3) The attacker manipulated 10.49.141.206 to login the Mysql server 10.49.253.35 but did not execute SQL-related commands (step iii.). Subsequently, 10.49.223.194 login and execute SQL commands on 10.49.253.35 (step iv.), causing an internal data leakage.
4) After 21:00, the enterprise internal scanner 10.49.21.15 discovered vulnerabilities such as SQL injection and arbitrary code execution in 10.49.223.194 during the scanning work (step v.), and fixed the vulnerabilities in time, making the subsequent attack attempt fail.

In addition, we compare our proposed scenario reconstruction approach with some attack modeling techniques [8], [29], [30] mentioned in Section II-C, the comparison is summarized as shown in Table 7. Our scheme identifies the attackers, victims, and other identities of web back-end database leakage incidents. The reconstructed topology diagram of the data leakage scenario visualizes the time, cause, participating hosts, and evolution of the incident. Moreover, the reconstructed results largely match the actual enterprise data leakage incident situation, which validates the effectiveness of our proposed WeB-DLIR in the real-world web security application.

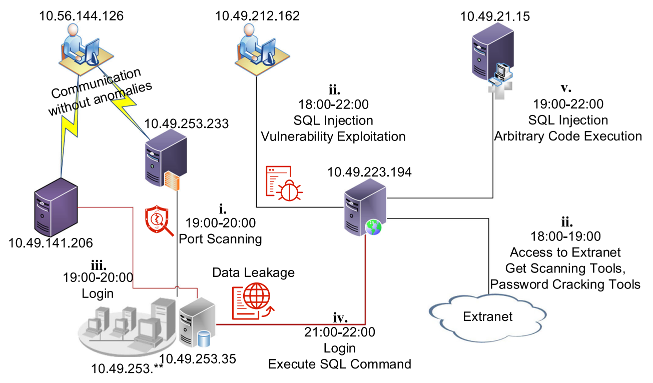

10.56.144.126
10.49.212.162
10.49.21.15
Communication without anomalies
10.49.253.233
ii. 18:00-22:00 SQL Injection Vulnerability Exploitation
v. 19:00-22:00 SQL Injection Arbitrary Code Execution
10.49.223.194
10.49.141.206
i. 19:00-20:00 Port Scanning
ii. 18:00-19:00 Access to Extranet Get Scanning Tools, Password Cracking Tools
iii. 19:00-20:00 Login
Data Leakage
Extranet
iv. 21:00-22:00 Login Execute SQL Command
10.49.253.**
10.49.253.35

FIGURE 9. **The result of web back-end database leakage incident scenario reconstruction.**

TABLE 7. **Comparative summary among the attack modeling techniques.**

Category/Function	[8]	[29]	[30]	Ours
Type	AG	AG	AT	AG
Include Time Perspectives	×	×	×	✓
Present Causality	✓	✓	✓	✓
Clear Views	✓	×	×	✓
Include Hosts	✓	×	×	✓
Mark The Host Identity	×	×	×	✓

'AG' is short for 'Attack Graph,' 'AT' is short for 'Attack Tree'.

VI. CONCLUSION

In this paper, we proposed WeB-DLIR, a web back-end database leakage incident reconstruction framework over unlabeled logs, which is designed to enhance the intelligence and automation of web back-end database leakage scenario reconstruction in a real-world network environment. To achieve this goal, we proposed a series of new methods supporting unlabeled web-related multi-source log anomaly detection, anomaly event correlation fusion, attack scenario modeling, and data leakage scenario reconstruction. The utility and practicality of our framework were demonstrated using evaluations.

Future research will include the following aspects.

1) From web-based data leakage post-event analysis and forensic investigation scenarios to the online analysis scenarios in applications, it is necessary to consider the time efficiency evaluation of comprehensive analysis methods. Higher accuracy and faster comprehensive analysis methods need to be explored for online security incident analysis scenarios.
2) Due to the existence of attacks that tamper with the system time or errors in the recording of the data capture stage, which may lead to the appearance of untrue and incomplete time in the data. It is necessary to consider the credibility of web-related log data and design mechanisms such as weight allocation, error correction or feedback.

ACKNOWLEDGMENTS

The authors thank the editor-in-chief, associate editor, and reviewers for their valuable comments and suggestions. We would like to thank the 2019 CCF Big Data & Computing Intelligence Contest and QI-ANXIN. They provided experimental data for our work and evaluated our competition proposal, and finally, the result of our proposal won first place in the competition.

REFERENCES

[1] E. Costante, J. den Hartog, M. Petković, S. Etalle, and M. Pechenizkiy, "A white-box anomaly-based framework for database leakage detection," *J. Inf. Secur. Appl.*, vol. 32, pp. 27–46, 2017.
[2] T. K. George, K. P. Jacob, and R. K. James, "Token based detection and neural network based reconstruction framework against code injection vulnerabilities," *J. Inf. Secur. Appl.*, vol. 41, pp. 75–91, 2018.
[3] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "A survey on data leakage prevention systems," *J. Netw. Comput. Appl.*, vol. 62, pp. 137–152, 2016.
[4] S. Liang, Y. Zhang, B. Li, X. Guo, C. Jia, and Z. Liu, "SecureWeb: Protecting sensitive information through the web browser extension with a security token," *Tsinghua Sci. Technol.*, vol. 23, no. 5, pp. 526–538, 2018.
[5] Z. Li and A. Oprea, "Operational security log analytics for enterprise breach detection," in *Proc. IEEE Cybersecurity Develop.*, 2016, pp. 15–22.
[6] M. A. Latib, S. A. Ismail, O. M. Yusop, P. Magalingam, and A. Azmi, "Analysing log files for web intrusion investigation using hadoop," in *Proc. 7th Int. Conf. Softw. Inf. Eng.*, 2018, pp. 12–21.
[7] J. Yu, S. Zhang, P. Liu, and Z. Li, "LeakProber: A framework for profiling sensitive data leakage paths," in *Proc. 1st ACM Conf. Data Appl. Secur. Privacy*, 2011, pp. 75–84.
[8] M. N. Hossain et al., "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 487–504.
[9] K. Pei et al., "HERCULE: Attack story reconstruction via community discovery on correlated log graph," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, 2016, pp. 583–595.
[10] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, May 2020.
[11] T. Nishiyama, A. Kumagai, K. Kamiya, and K. Takahashi, "SILU: Strategy involving large-scale unlabeled logs for improving malware detector," in *Proc. IEEE Symp. Comput. Commun.*, 2020, pp. 1–7.
[12] W. Chen, F. Kong, F. Mei, G. Yuan, and B. Li, "A novel unsupervised anomaly detection approach for intrusion detection system," in *Proc. IEEE 3rd Int. Conf. Big Data Secur. Cloud*, 2017, pp. 69–73.
[13] A. Pomeroy and Q. Tan, "Effective SQL injection attack reconstruction using network recording," in *Proc. IEEE 11th Int. Conf. Comput. Inf. Technol.*, 2011, pp. 552–556.
[14] M. Babiker, E. Karaarslan, and Y. Hoscan, "Web application attack detection and forensics a survey," in *Proc. 6th Int. Symp. Digit. Forensic Secur.*, 2018, pp. 1–6.
[15] Y.-C. Chen, Y.-W. Ma, and J.-L. Chen, "Intelligent malicious URL detection with feature analysis," in *Proc. IEEE Symp. Comput. Commun.*, 2020, pp. 1–5.
[16] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.
[17] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, "CNN-webshell: Malicious web shell detection with convolutional neural network," in *Proc. 6th Int. Conf. Netw., Commun. Comput.*, 2017, pp. 75–79.
[18] P. Li et al., "Act an attentive convolutional transformer for efficient text classification," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 13261–13269.
[19] Q. Xiao, M. Gao, S. Wu, and X. Sun, "Attention-based improved BLSTM-CNN for relation classification," in *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 34–43.
[20] Y. Zhao, T. Zhou, Z. Chen, and J. Wu, "Improving deep CNN networks with long temporal context for text-independent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6834–6838.

[21] G. Kul et al., "Ettu: Analyzing query intents in corporate databases," in *Proc. 25th Int. Conf. Companion World Wide Web*, 2016, pp. 463–466.
[22] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *VLDB J.*, vol. 17, no. 5, pp. 1063–1077, 2008.
[23] S. Yuan and C. Zou, "The security operations center based on correlation analysis," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, 2011, pp. 334–337.
[24] S. Zhang, Y. Gao, M. Zhang, J. Ge, and S. Wang, "The study of network security event correlation analysis based on similar degree of the attributes," in *Proc. 4th Int. Conf. Digit. Manuf. Automat.*, 2013, pp. 1565–1569.
[25] J. Liu, L. Gu, G. Xu, and X. Niu, "A correlation analysis method of network security events based on rough set theory," in *Proc. 3rd IEEE Int. Conf. Netw. Infrastructure Digit. Content*, 2012, pp. 517–520.
[26] A. Ambre and N. Shekokar, "Insider threat detection using log analysis and event correlation," *Procedia Comput. Sci.*, vol. 45, pp. 436–445, 2015.
[27] H. S. Lallie, K. Debattista, and J. Bal, "An empirical evaluation of the effectiveness of attack graphs and fault trees in cyber-attack perception," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1110–1122, May 2018.
[28] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Comput. Sci. Rev.*, vol. 35, 2020, Art. no. 100219.
[29] W. Niu, X. Zhang, G. Yang, R. Chen, and D. Wang, "Modeling attack process of advanced persistent threat using network evolution," *IEICE Trans. Inf. Syst.*, vol. 100, no. 10, pp. 2275–2286, 2017.
[30] R. Maciel, J. Araujo, J. Dantas, C. Melo, E. Guedes, and P. Maciel, "Impact of a DDoS attack on computer systems an approach based on an attack tree model," in *Proc. Annu. IEEE Int. Syst. Conf.*, 2018, pp. 1–8.
[31] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics review, taxonomy, and open challenges," *J. Netw. Comput. Appl.*, vol. 66, pp. 214–235, 2016.
[32] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious URL detection using machine learning a survey," 2017, *arXiv1701.07179*.
[33] F. Chollet et al., "Keras," 2015. [Online]. Available: http://sgithub.comfcholletkeras
[34] N. Schagen, K. Koning, H. Bos, and C. Giuffrida, "Towards automated vulnerability scanning of network servers," in *Proc. 11th Eur. Workshop Syst. Secur.*, 2018, pp. 1–6.
[35] A. Suh, M. Hajij, B. Wang, C. Scheidegger, and P. Rosen, "Persistent homology guided force-directed graph layouts," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 697–707, Jan. 2020.
[36] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, 2019.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980.*
[38] G. Zhang, C. Zhang, and H. Zhang, "Improved k-means algorithm based on density canopy," *Knowl.-Based Syst.*, vol. 145, pp. 289–297, 2018.
[39] R. Ramachandran, P. Arya, and P. Jayanthy, "A novel method for intrusion detection in relational databases," in *Proc. Int. Conf. Adv. Comput., Commun. Informat.*, 2017, pp. 230–235.
[40] S. Kim et al., "Application of density-based outlier detection to database activity monitoring," *Inf. Syst. Front.*, vol. 15, no. 1, pp. 55–65, 2013.
[41] G. Gavai, K. Sricharan, D. Gunning, R. Rolleston, J. Hanley, and M. Singhal, "Detecting insider threat from enterprise social and online activity data," in *Proc. 7th ACM CCS Int. Workshop Manag. Insider Secur. Threats*, 2015, pp. 13–20.

YANHUA LIU received the BS and MS degrees from the College of Computer and Data Science, Fuzhou University, China, in 1996 and 2003, respectively, and the PhD degree from the College of Physics and Information Engineering, Fuzhou University, China, in 2016. He is currently working as an associate professor and researcher of the Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University. His research interests are intelligent computing, computer security, and Big Data. His research work has won several government awards.

ZHIHUANG LIU received the BS degree in information security from Fuzhou University, in 2020. He is currently working toward the MS degree with the College of Computer and Data Science, Fuzhou University. His research interest includes machine learning, visual analysis, and cyberspace security.

XIMENG LIU (Senior Member, IEEE) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the PhD degree in cryptography from Xidian University, China, in 2015. He is currently a full professor with the College of Computer and Data Science, Fuzhou University. He was a research fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 250 research articles including *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Cloud Computing*. He was awarded the Minjiang Scholars Distinguished professor and ACM SIGSAC China Rising Star Award, in 2018. He served as a program committee for several conferences, such as the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, and the 2017 IEEE Global Communications Conference. He served as the lead guest editor for WCMC, IJDSN, and ETT. His research interests include cloud security, applied cryptography, and Big Data security.

WENZHONG GUO received the BS and MS degrees in computer science from Fuzhou University, Fuzhou, China, in 2000 and 2003, respectively, and the PhD degree in communication and information system from Fuzhou University, in 2010. He is currently a full professor with the College of Computer and Data Science, Fuzhou University. His research interests include cloud computing, mobile computing, and evolutionary computation.