International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 11 | Nov 2024 www.irjet.net p-ISSN: 2395-0072

Case Study on Data Leakage Prevention File System (DLPFS)

Kshitij Rai, Aman Tripathi, Vijesh Chaudhari, Dr. Sujata Bhairnallykar

[1]*Consultant at MindCraft Software Pvt. ltd, Mumbai, India*
[2]*Java Developer at Tata Consultancy Services Ltd, Thane, India*
[3]*Software Engineer at Code B solutions Pvt. ltd, Mumbai, India*
[4]*Head Of Department at Saraswati College of Engineering, Kharghar*

Abstract - *Data Leakage Prevention (DLP) is a crucial technique used by organizations to safeguard sensitive or confidential data. A core component of DLP is the de-identification of Personally Identifiable Information (PII) before sharing it with third parties or stakeholders. Techniques such as masking and replacement are employed to conceal or anonymize PII. Masking involves substituting specific elements, like names or addresses, with generic placeholders such as "<PERSON>" or "*". Replacement substitutes sensitive data with similar but fictitious values, for instance, replacing the name "Aman" with "Rishi". Beyond de-identification, DLP systems incorporate access control mechanisms for uploading and reviewing files, ensuring that only authorized personnel can access the data. Additionally, DLP systems encrypt stored data to prevent breaches, rendering the data inaccessible without the appropriate decryption key, even in the event of theft. As organizations increasingly need to share information with multiple parties, the adoption of DLP technology is essential for maintaining the privacy and security of sensitive information while enabling secure data sharing with relevant stakeholders.*

Keywords*: *De-identification, Extraction, Encryption, Pseudonymization, anonymization

1.INTRODUCTION

Data breaches are frequently attributed to human error, such as misconfigurations or inadequate data governance, rather than external hacking attempts. Misconfigured applications and software bugs pose a constant risk to the confidentiality of sensitive information. Common examples of data leakage include log files that inadvertently store sensitive details like usernames and passwords, as well as stack traces or core dumps from crashed applications that expose private data.

Traditional data protection strategies, such as access control and encryption, are often insufficient to address all potential data leakage scenarios. Data leakage can occur when data is shared among multiple users or systems, or when it must be accessible for purposes like auditing or debugging. The widely practiced approach of creating multiple versions of datasets for different purposes is costly and impractical, especially for large-scale data handling.

1.1 DLP File System

The Data Leakage Prevention File System (DLPFS) introduces an innovative approach to secure data sharing across applications and systems. By leveraging advanced data type identification and de-identification technologies, DLPFS provides robust data protection. It integrates seamlessly into existing infrastructures by exposing a POSIX file system API, enabling applications to access protected data subtrees without significant modifications.

DLPFS enables users to share data securely and maintain privacy across multiple systems without the need to generate custom copies of data for different applications. Additionally, it supports legacy applications by allowing them to operate on de-identified data in real-time, eliminating the need for modifications to the applications themselves.

1.2 Transitioning DLPFS to a Web-Based Application

While the original DLPFS was designed to secure data within POSIX-compliant file systems, its architecture is inherently tied to file-based operations, limiting its applicability in modern, web-based environments. In response to the growing need for privacy-preserving data handling in distributed and dynamic web systems, this project reimagines DLPFS as a **web-based application**, addressing the challenges of real-time data sharing over HTTP protocols.

The adapted system retains the core principles of DLPFS, such as sensitive data detection, de-identification, and robust access control, while extending its functionality to meet the demands of web-based workflows. By replacing the POSIX file system API with a scalable web architecture, this project integrates state-of-the-art data masking, redaction, and anonymization techniques into a middleware solution that operates seamlessly across distributed systems.

Key innovations in this adaptation include:

- **Real-Time Data Protection**: Implementation of automated workflows for sensitive data detection and de-identification during data uploads and retrievals.
- **Role-Based Access Control (RBAC)**: Design of a web-native access control mechanism, ensuring secure, role-specific access to sensitive information.
- **Scalability and Usability**: Optimization for dynamic, high-volume environments, enabling secure data sharing across diverse stakeholders without duplicating datasets.

This transition not only modernizes the original DLPFS architecture but also addresses key limitations of POSIX-based systems, such as stateless data handling, interoperability with web protocols, and real-time scalability. By bridging the gap between file-based systems and web applications, the project demonstrates how DLP principles can be effectively extended to secure sensitive data in today's web-driven ecosystems.

1.3 Challenges Addressed by This Project

Transitioning DLPFS from a POSIX file system to a web-based architecture presented several unique challenges:

- **Data Accessibility vs. Privacy**: Balancing the need for seamless data accessibility with strong privacy measures for sensitive information.
- **Real-Time Processing**: Ensuring that sensitive data detection and de-identification occur in real-time without introducing significant latency.
- **System Interoperability**: Designing a solution compatible with existing web protocols and RESTful API standards to support widespread adoption.

By addressing these challenges, this project demonstrates how traditional file-system-based DLP mechanisms can be effectively reimagined for web environments.

1.4 Practical Applications of the Web-Based DLPFS

The web-based adaptation of DLPFS has significant implications across various industries and real-world applications:

- **Healthcare Systems**: Protecting patient records by de-identifying sensitive information such as names, addresses, and medical history before sharing for research, analytics, or regulatory purposes, ensuring compliance with **HIPAA** and similar standards.
- **Financial Institutions**: Securing sensitive financial data, including transaction logs and customer information, to mitigate risks of data leakage during audits, fraud detection, or third-party collaboration.
- **E-Commerce Platforms**: Enabling secure handling of user data and transaction logs while ensuring privacy during real-time analysis and reporting in distributed systems.

2. LITERATURE REVIEW

Data Leakage Prevention (DLP) and data de-identification techniques are critical for securing sensitive information, especially in web-based environments where data sharing is prevalent. This survey reviews foundational research on DLP techniques, with a focus on their application in file systems and their relevance to web-based implementations. It highlights the shift from POSIX-based DLP systems, like the original DLPFS, to web-based architectures.

2.1 POSIX File Systems and DLPFS:

The original DLPFS, introduced by Braghin et al. (2020), was developed as a middleware for POSIX file systems to secure sensitive data during read/write operations. It implemented de-identification techniques such as masking and redaction while maintaining compatibility with existing file systems. Although DLPFS effectively prevented data leakage in local and distributed POSIX environments, it lacked adaptability for web-based applications where data handling occurs over HTTP protocols rather than file system interfaces. This case study addresses that gap by reimagining DLPFS for web-based systems, ensuring robust security in a more dynamic environment.

2.2 Web-Based Data Leakage Prevention

Shapira et al. (2019) proposed a content-based data leakage detection system for cloud-based applications, emphasizing the need for real-time detection and secure data sharing. Their approach, while effective for cloud environments, did not address the challenges of adapting file-system-based solutions like DLPFS to web architectures. Our study bridges this gap by demonstrating how DLP techniques can be integrated into web applications for secure data handling.

2.3 Transition from File Systems to Web Applications

While POSIX file systems offer reliable methods for handling sensitive data, web applications present unique challenges, including stateless protocols and multi-layered architectures. Studies like those by Chellaprabha and Archana (2013) highlighted anomaly detection in dynamic environments, but their techniques were limited to large-scale data flows and did not explore file-system-level

integrations in web contexts. This case study explores the technical adaptations required to transition DLPFS from a file-based system to a web application, ensuring it retains its core functionality while addressing web-specific challenges.

2.4 De-Identification Techniques in Web Contexts

Masking, redaction, and anonymization are widely used in securing sensitive data. M. Young (2019) and Kumar et al. (2020) explored these techniques in various domains, such as financial and healthcare data. However, their focus was largely on data stored in structured databases or processed in cloud environments. Our work adapts these de-identification techniques for web-based file systems, ensuring real-time protection without compromising usability.

2.5 Differential Privacy and Scalability

Li and Lin (2018) demonstrated the utility of differential privacy in healthcare data, introducing noise for statistical protection. While their techniques were effective in ensuring privacy, they were not optimized for integration into middleware solutions like DLPFS. This case study evaluates the scalability of differential privacy techniques for web-based systems, addressing challenges like latency and performance.

2.6 Redaction and Real-Time Security

Grobauer et al. (2011) focused on redaction for secure data sharing in cloud-based systems but lacked automation for real-time implementations. This limitation is critical when adapting file-system-based solutions like DLPFS to web applications, where real-time data flow is common. Our work introduces automated redaction mechanisms suitable for web-based environments.

2.7 Integration of DLP Techniques into Web-Based Systems

Venkata Kumar et al. (2020) explored encryption and de-identification in cloud storage, emphasizing the importance of securing data at rest and in transit. However, their methods were primarily designed for cloud storage services and did not address the transition from file-based to web-based architectures. This case study demonstrates how DLP techniques can be seamlessly integrated into web applications, ensuring robust security and scalability.

Gaps in the Literature

While significant research has been conducted on DLP techniques and POSIX-based file systems, there is limited exploration of transitioning such systems to web-based architectures. Existing studies focus on either file systems or cloud environments but rarely address the unique challenges of web applications. This case study addresses these gaps by adapting DLPFS for a web-based application, ensuring secure data storage and retrieval in dynamic, stateless environments.

3. IMPLEMENTATION

The implementation of our case study builds upon the Data Leakage Prevention FileSystem (DLPFS) framework, extending its concept to a web-based application for secure file handling and privacy-preserving transformations. It is designed to mitigate data leakage risks by intercepting file I/O operations, identifying sensitive data, and applying privacy transformations.

3.1 System Architecture

The system comprises the following key components:

- Data Identification Module: Detects sensitive data patterns using regex-based matching.
- Transformation Engine: Applies privacy-preserving transformations(redaction,masking,anonymization)
- Knowledge Base: Stores patterns and transformation rules in JSON format for flexibility.
- Web Interface: Facilitates file uploads and configuration management for privacy policies.
- Secure File Management: Ensures processed data is stored securely with access controls.

The architecture integrates seamlessly with web-based infrastructures and supports real-time data processing

3.2 Features and Modules

For Data Identification - sensitive data patterns are defined using regex rules stored in the Knowledge Base. Key patterns include

```
[
    {
        type:'Email',
        reg: /\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b/gi,
        redact:'<EMAIL_ID>'
    },
    {
        //add formate checker
       type:'Phone_Number',
        reg:/(\+\d{1,2}\s)?\(?\d{3}\)?[/\s.-]?\d{3}[/\s.-]?\d{4}/g,
        redact:'<PHONE_NUMBER>'
    },
   {
       type:'SSN',
        reg:/\d{3}-?\d{2}-?\d{4}/g,
        redact:'<SSN>'
    },
]
```

Fig -1: Example of PII Type

For Data Transformation - Detected sensitive data undergoes transformation based on configurable rules:

- Redaction: Replaces sensitive data with asterisks (e.g., john.doe@example.com → ********).
- Masking: Substitutes sensitive data with syntactically similar but fictional values.
- Anonymization: Applies differential privacy techniques to numerical data.

These transformations are applied in compliance with user-defined configurations.

For file Read and Write - The system intercepts file uploads and processes them in the following steps:

- Upload: The user uploads a file through the web interface.
- Inspection: The backend scans the file for sensitive data patterns.
- Transformation: Detected patterns are transformed based on Knowledge Base rules.
- Storage: Transformed files are securely stored, and the original files are discarded.

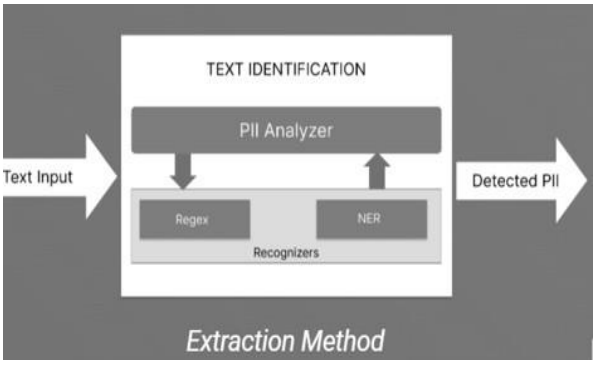

Fig-2: Processing of Text Identification

The above diagram illustrates the process of PII (Personally Identifiable Information) Detection and Extraction. It begins with the input of text, which is then processed by a PII Analyzer. This analyzer employs two primary methods: Regex (Regular Expressions) and NER (Named Entity Recognition). Regex identifies PII based on predefined patterns, while NER leverages machine learning models to recognize entities like names, addresses, and social security numbers. The detected PII is then outputted as the final result.

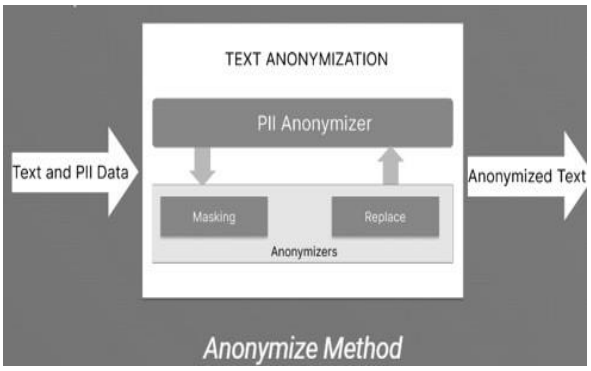

Fig -3: Processing of Text Anonymization

The above diagram illustrates the process of Text Anonymization. It begins with the input of text containing PII (Personally Identifiable Information). This text is fed into the PII Anonymizer, which employs two anonymization methods: Masking and Replace. The Masking method obscures PII by replacing sensitive characters with special characters, while the Replace method substitutes PII with generic or placeholder values. The anonymized text, devoid of sensitive information, is then outputted as the final result.

5. CONCLUSIONS

Data Leakage Prevention File System extracts the PII data efficiently and Masks or Replace PII data based on users choice to store or share with 3rd parties and stakeholders. Extraction of Address is still one of the issues which we need to work on. In conclusion, a data leakage file system is a critical element in ensuring the security of sensitive organizational information. It allows organizations to regulate access to information and control who has access to specific data. A well-designed data leakage file system should allow for easy administration, real-time monitoring, and a robust audit trail system. This will help in identifying any unauthorized access to sensitive data and respond to such incidents promptly. Overall, organizations should invest in a robust data leakage file system as part of their overall security strategy to prevent data breaches and data loss.

6. REFERENCES

[1] Braghin, S., Sinn, M., Simioni, M., "Data Leakage Prevention File System (DLPFS)," IEEE Security & Privacy, 2020.

[2] Shapira, Y., Shabtai, A., "Content-Based Data Leakage Detection using Extended Fingerprinting," Journal of Network and Computer Applications, 2019.

[3] Chellaprabha, B., Archana, "Anomaly Data Leakage Detection," International Journal of Engineering and Innovative Technology (IJEIT), 2013.

[4] Young, M., "Data Masking for Financial Information Security," Journal of Information Security, 2019.

[5] Li, T., Lin, J., "Differential Privacy for Healthcare Data Anonymization," Journal of Data Privacy, 2018.

[6] Venkata Kumar, D., et al., "Data De-Identification and Encryption for Cloud Systems," International Journal of Computer Science, 2020

[7] A.Yuri Shapira, Bracha Shapira, Asaf Shabtai,"Content-Based Data Leakage Detection using ext-fingerprinting"

[8] Dilip Venkata Kumar Vengala1, D. Kavitha, A.P.Siva Kumar, "Three factor authentication system with modified ECC based secured data transfer: untrusted cloud environment

[9] Subramanian, Helen Nissenbaum, Prateek Mittal, "VACCINE: Using Contextual Integrity for Data Leakage Detection", 2019 World Wide Web Conference(WWW'19)

[10]Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N. and Lo Iacono, L. (2011). All your clouds belong to us. Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11

[11] Narayanan, A., Shmatikov, V., *"Robust De-anonymization of Large Sparse Datasets,"* IEEE Symposium on Security and Privacy, 2008.

[12] Abadi, M., Chu, A., Goodfellow, I., et al., *"Deep Learning with Differential Privacy,"* Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.

[13] Ristenpart, T., Tromer, E., Shacham, H., Savage, S., *"Hey, You, Get Off of My Cloud: Exploring Information Leakage in Shared Cloud Computing,"* ACM Conference on Computer and Communications Security, 2009.

BIOGRAPHIES

Software Developer at Mindcraft Software Pvt Ltd.

Java Developer at Tata Consultancy Services Ltd.

Software Developer at Code B Solutions Pvt Ltd.

Head of Department at Saraswati College of Engineering, Kharghar