# Automating Cloud-Based Device Log Processing Using Google Apps Script

Muhammad Hasbi Asshidiqi[1], Muhammad Hilman Naufal[2], Muhamad Fikri[3], Oky Tria Saputra[4]
Teknologi Rekaya Jaringan Komputer, Politeknik IDN Bogor
Email: hasbi.asshidiqi@idn.ac.id, hilmannaufal212@gmail.com, fikrimuhammadhilabi@gmail.com, okytria@gmail.com

Device log management is often a challenge for system administrators due to large data volumes and unstructured formats. Manual processing is time-consuming and prone to human error. This research aims to implement an automated device log processing system utilizing Google Apps Script (GAS) as a cloud-based processing engine. The research method used is Research and Development (R&D), consisting of log data collection, automation script design, integration with Google Sheets as a database, and functional testing. The results show that the developed system is capable of parsing log data in real-time, categorizing log types based on urgency levels, and presenting them in structured reports. Efficiency testing demonstrates a reduction in data processing time by [X]% compared to conventional methods. The conclusion of this study is that Google Apps Script provides a cost-effective and efficient solution for managing device logs for medium-scale institutions.
**Keywords**: Google Apps Script, Device Log, Automation, Cloud Computing, Network Management.



**Corresponding Author:**
Muhammad Hasbi Asshidiqi
Teknologi Rekaya Jaringan Komputer, Politeknik IDN Bogor
Bogor
hasbi.asshidiqi@idn.ac.id

## 1. Introduction

Device log management is a critical component in maintaining the reliability of information technology infrastructure, particularly in supporting systematic network troubleshooting processes [1]. One of the primary challenges in this domain lies in the massive volume of data and the highly heterogeneous structure of computer system logs, which necessitates automated tools for data structuring and normalization [2]. In addition, continuous log monitoring plays a vital role in aggregating access records to identify and prevent illicit activities [3]. However, for small and medium-sized organizations, the adoption of comprehensive security systems is often constrained by cost–benefit considerations [4].

Conventional log management practices face significant obstacles, including the complexity of parsing raw log data and transforming it into actionable information. Although advanced technologies such as Security Information and Event Management (SIEM) systems are available, their operation requires substantial expertise in security architecture and system integration [5]. In the absence of efficient processing mechanisms, identifying attack chains within encrypted system logs becomes increasingly difficult for system administrators [6].

As an alternative approach, serverless cloud platforms offer substantial cost efficiency by eliminating the need for dedicated physical server infrastructure. Google Apps Script (GAS) is a cloud-based platform that enables automation without server management overhead [7]. Previous studies have demonstrated the effectiveness of GAS in automating scientific data analysis [8], improving administrative efficiency through workflow automation [9], and developing cloud-based data retrieval applications [7]. Its seamless

integration with Google Sheets further enhances its functionality as a flexible database solution for various monitoring purposes [10].

Despite its widespread application in improving organizational cost efficiency [11], research focusing on the specific utilization of Google Apps Script for automated device log processing remains limited. This study aims to design an automated device log management system using Google Apps Script to enable continuous and efficient monitoring processes. Through this approach, system administrators are expected to manage device logs using a zero-cost solution while maintaining reliable and functional monitoring capabilities [12].

## 2. Literature Review

Existing studies on log management and security monitoring underline the strategic role of structured log analysis in supporting incident detection, forensic investigation, and system accountability. From a theoretical standpoint, log data represents a chronological record of system activities that can be analyzed to identify abnormal behavior, correlate security events, and trace potential attack paths [13]. However, the effectiveness of log-based monitoring is highly dependent on the availability of automated mechanisms for parsing, normalization, and aggregation, particularly when dealing with heterogeneous log formats generated by diverse devices and applications. Analytical models in information security emphasize that without systematic preprocessing and automation, raw log data provides limited analytical value and significantly increases the workload of system administrators, especially in resource-constrained environments [14].

Research on cloud computing and serverless architectures further demonstrates a shift in system management paradigms, where infrastructure abstraction enables more efficient and scalable data processing. Serverless platforms support event-driven automation and reduce operational complexity by eliminating the need for direct server maintenance. Within this context, Google Apps Script has been theoretically positioned as a lightweight automation framework capable of integrating data processing logic with cloud-based storage systems [15]. Despite its proven effectiveness in general automation and data management scenarios, its potential application in automated device log processing remains underexplored. This condition reveals a conceptual and practical research gap between log management theory and low-cost serverless automation tools. Based on this gap, the research problem formulated in this study is to determine how an automated device log processing system can be designed using Google Apps Script to enable structured log analysis and continuous monitoring while maintaining minimal cost and technical complexity.

## 3. Metode

This study employs a system development methodology consisting of several structured stages, including data collection, system design, code implementation using cloud-based scripting, and functional testing. Overall, the system architecture adopts a zero-cost approach by fully leveraging the Google ecosystem, thereby minimizing infrastructure costs and operational expenses [12].

### Log Data Collection

The log data utilized in this study is heterogeneous, covering activity records from network devices and computer systems [2]. This process aims to capture critical variables required to support systematic troubleshooting when disruptions occur within the infrastructure [1]. The collected log data forms the basis for analyzing system behavior and identifying potential technical or security-related issues.

### Cloud-Based System Design

The system is designed using a serverless architecture based on Google Apps Script (GAS) to avoid reliance on expensive physical infrastructure [7]. Device logs are transmitted to Google Sheets, which functions as a flexible database for storing semi-structured data [10]. The system workflow consists of several stages: data ingestion, where logs are delivered via HTTP protocols (GET/POST) or retrieved through periodic file reading; data parsing, in which GAS separates raw log entries into structured fields such as timestamp, host, activity, and severity; and automation, which enables automatic data aggregation to support the detection of abnormal or illegal [3].

### Code Implementation (Cloud Scripting)

The implementation phase involves developing script modules within the Google Apps Script editor. The use of cloud scripting is selected due to its stable performance in processing data within Google Sheets [16]. The program code is designed to execute automated data processing tasks to improve the administrative efficiency of log monitoring activities [9].

### Testing and Evaluation

The final stage involves functional testing to ensure that the system is capable of automatically structuring diverse log formats [2]. Evaluation is also conducted to assess cost effectiveness and operational benefits by comparing the proposed system with conventional manual log processing methods [5].

## 4. Results and Discussion

### System Implementation

The system implementation was carried out by integrating Google Drive services as a storage medium for raw data and Google Apps Script (GAS) as the automated processing engine. In accordance with the established methodology, the system was designed to process log files in .txt or .log formats containing the output of network commands, specifically *show mac-address*.


**Figure 1.** Log File

This implementation relies on a cloud-based ecosystem to ensure efficiency and real-time data availability [17]. The process begins with the use of the DriveApp service in Google Apps Script to automatically scan designated folders in order to identify available log files [18]. Once a file is detected, the script extracts the entire textual content using the getBlob().getDataAsString() method. At this stage, a cloud scripting–based parsing algorithm operates to decompose heterogeneous data so that it can be transformed into structured information [19].

The script is developed using text structure interpretation techniques to recognize line patterns containing MAC address, VLAN, and device interface information, while excluding header data or non-essential characters. The application of Regular Expressions (Regex) within the script ensures high extraction accuracy, even when dealing with semi-structured data sources. This approach aligns with advancements

ISSN 2828-9463


in automation on modern spreadsheet platforms, which enable the automated processing of complex data structures [20].

```
------ Device1#show mac address ------

Legend: * - primary entry
        age - seconds since last seen
        n/a - not available
        S - secure entry
        R - router's gateway mac address entry
        D - Duplicate mac address entry
        # - Fabric mac address entry. Clear mac cli doesn't clear this entry

Displaying entries from active supervisor::

      vlan   mac address     type    learn    age              ports
-----+----+---------------+-------+-----+----------+--------------------------
       464 aaaa.d228.b3a6  dynamic  Yes      130     Po17
      1228 aaaa.905a.218e  dynamic  Yes      130     Po18
      1269 aaaa.5610.0b5d  dynamic  Yes       45     Po27
       464 aaaa.8be7.edfe  dynamic  Yes      130     Po17
      1286 aaaa.0a2e.bca5  dynamic  Yes       35     Po32
```

**Figure 1.** Log Contents 1

```
------ Device2#show mac address ------

Legend: * - primary entry
        age - seconds since last seen
        n/a - not available
        S - secure entry
        R - router's gateway mac address entry
        D - Duplicate mac address entry
        # - Fabric mac address entry. Clear mac cli doesn't clear this entry

Displaying entries from active supervisor::

      vlan   mac address     type    learn    age              ports
-----+----+---------------+-------+-----+----------+--------------------------
       527 bbbb.56aa.a420  dynamic  Yes       90     Po1
       511 bbbb.e75c.b8e5  dynamic  Yes       90     Po22
       661 bbbb.32a7.0563  dynamic  Yes       90     Po1
       525 bbbb.d228.cf2a  dynamic  Yes       90     Po16
      1448 bbbb.8859.2119  dynamic  Yes       90     Po49
```

**Figure 2.** Log Contents 2

The data that has been successfully extracted is then systematically mapped into structured columns in Google Sheets using the setValues() function [20]. This integration demonstrates that the Google ecosystem is capable of substituting conventional database systems by offering a high level of flexibility in managing data acquired remotely from network devices. The entire workflow operates in a serverless manner, ensuring that log data processing can be performed without the need for additional physical infrastructure or operational costs, thereby supporting a zero-cost approach [17]. This implementation aligns with cloud-based data monitoring principles, which enable centralized and efficient access to information for users. The complete script used in this case study is presented below:

**Table 1.** Script Used

```
function parseMacTableFromFolder() {
  const FOLDER_ID = "Folder_ID";

  const folder = DriveApp.getFolderById(FOLDER_ID);
  const filesIter = folder.getFiles();

  // Ubah iterator → array
  const files = [];
  while (filesIter.hasNext()) {
```

ISSN 2828-9463


```
  files.push(filesIter.next());
}

// SORT BERDASARKAN NAMA FILE (A-Z)
files.sort((a, b) =>
  a.getName().localeCompare(b.getName(), 'en', { numeric: true })
);

const headers = [
  "No",
  "VLAN",
  "MAC Address",
  "Type",
  "Learn",
  "Age",
  "Port",
  "Source File"
];

const data = [];
let counter = 1;

const regex = /^\s*(\d+)\s+([0-9a-fA-F\.]+)\s+(\w+)\s+(\w+)\s+(\d+)\s+(\S+)/;

files.forEach(file => {
  if (!file.getName().toLowerCase().endsWith(".txt")) return;

  const sourceFileName = file.getName();
  const content = file.getBlob().getDataAsString();

  content.split("\n").forEach(line => {
    const m = line.match(regex);
    if (m) {
      data.push([
        counter++,
        m[1],
        m[2],
        m[3],
        m[4],
        m[5],
        m[6],
        sourceFileName
      ]);
    }
  });
});
```

```
  if (data.length === 0) {
    throw new Error("Tidak ada data MAC table yang ditemukan.");
  }

  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.clear();

  sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
  sheet.getRange(2, 1, data.length, headers.length).setValues(data);

  const lastRow = data.length + 1;
  const lastCol = headers.length;

  sheet.getRange(1, 1, lastRow, lastCol)
    .setHorizontalAlignment("center")
    .setVerticalAlignment("middle")
    .setBorder(true, true, true, true, true, true);

  sheet.getRange(1, 1, 1, lastCol)
    .setBackground("#1E88E5")
    .setFontColor("#FFFFFF")
    .setFontWeight("bold");

  sheet.autoResizeColumns(1, lastCol);
}
```

1. Data Connectivity (Google Drive Integration). The system utilizes a specific folder ID to access a collection of .txt files stored in Google Drive. The script iterates through all files within the designated folder and stores them in an array for subsequent processing.
2. Data Ordering (Natural Sorting). To maintain the integrity of data sequencing based on file names, the system implements the .sort() method with the numeric: true parameter. This approach ensures that files with numeric naming conventions (e.g., *Switch 1*, *Switch 2*, *Switch 10*) are ordered logically rather than lexicographically.
3. Data Extraction Using Regular Expressions (Regex). The core mechanism of the system involves the use of Regular Expressions to parse each line of text within the log files in order to accurately extract relevant information.

**Table 1.** Regular Expression

| /^\s*(\d+)\s+([0-9a-fA-F\.]+)\s+(\w+)\s+(\w+)\s+(\d+)\s+(\S+)/ |
|---|

The script intelligently separates columns:

a. VLAN (Angka)
b. MAC Address (Heksadecimal)
c. Type & Learn (Teks)
d. Age (Number)
e. Port (Interface Identity)

4. Interface Automation (Spreadsheet Formatting) After the data is extracted, the system writes it to the worksheet collectively using the .setValues() method. Visual implementation includes:
   a. Automatic clearing of old data (clear).
   b. Assigning blue header color (#1E88E5) with white text.
   c. Setting center alignment and automatic border creation.
   d. Auto-resize column feature so that cell width dynamically adjusts to text length.

### System Functionality Testing

The functionality testing phase was conducted to ensure that each script component operates in accordance with the predefined system requirements. This testing focuses on validating the serverless workflow, which ensures that log data processing can be performed without additional physical infrastructure or high operational costs. The test plan evaluated in this case study is outlined as follows:

**Table 2.** Test Plan 1

| | |
|---|---|
| **No** | TP-01 |
| **Test Item** | Folder Connectivity |
| **Test Scenario** | |
| Executing the getFolderById function using the specified folder ID. | |
| **Expected Result** | |
| The script successfully accesses the folder and counts the number of files contained within it. | |
| **Status** | Successful |

**Tabel 3.** Test Plan 2

| | |
|---|---|
| **No** | TP-02 |
| **Test Item** | Parsing Accuracy |
| **Test Scenario** | |
| Inputting standard switch log entries (VLAN, MAC, Type, Port). | |
| **Expected Result** | |
| The Regular Expression correctly separates each column and ignores empty lines or header information. | |
| **Status** | Successful |

**Tabel 4.** Test Plan 3

| | |
|---|---|
| **No** | TP-03 |
| **Test Item** | Format Automation |
| **Test Scenario** | |
| Verifying the appearance of headers and columns after script execution. | |
| **Expected Result** | |
| Headers are displayed with a blue background, white text, visible borders, and columns are automatically resized (auto-resize). | |
| **Status** | Successful |

### TP-01: Folder Connectivity

This test aims to validate the integration between the script and Google Drive cloud storage. The primary focus is to ensure that the script accurately recognizes the predefined FOLDER_ID. The system is tested to access the directory, identify all files within it, and perform numeric sorting of file names. Successful execution is indicated by the script's ability to retrieve the relevant list of log files without encountering permission issues or folder ID errors.

The folder ID can be obtained from the shared Google Drive link. The folder ID format is illustrated below:

**Table 5.** Format Folder ID

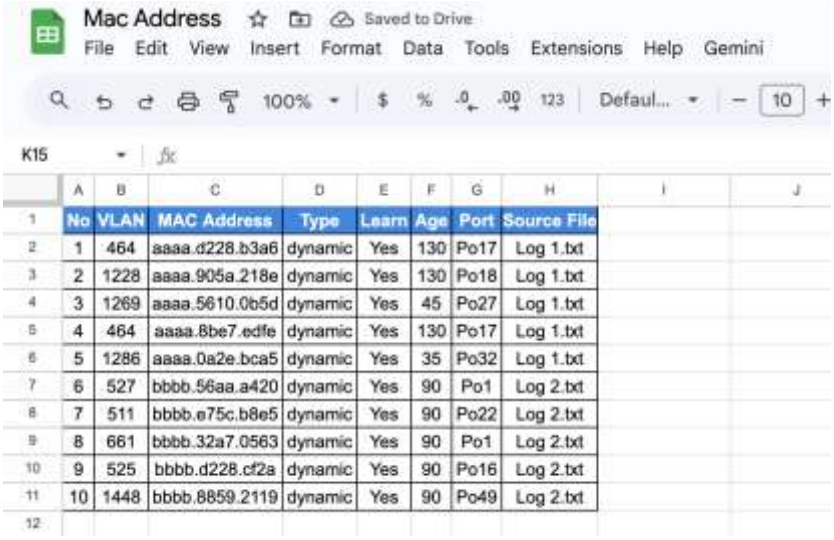
| https://drive.google.com/drive/folders/xxxxxxxxxxxxxxxxxxxxxxxxx |
|---|


**Figure 3.** Script Execution

After executing the script, the output shown in Figure 2 indicates the message "execution completed," confirming that the script ran successfully.

### TP-02: Parsing Accuracy

This stage represents the core of the functionality testing, focusing on validating the accuracy of the Regular Expression (Regex) algorithm in parsing raw log data. The test is conducted by inputting heterogeneous log text lines, including empty lines and header text generated by network devices. The system is considered to have passed the test if it is able to accurately extract specific variables such as VLAN, MAC address, and port into their corresponding columns, while simultaneously eliminating irrelevant non-data characters. The parsing results generated by the executed script are presented below:




| No | VLAN | MAC Address | Type | Learn | Age | Port | Source File |
|---|---|---|---|---|---|---|---|
| 1 | 464 | aaaa.d228.b3a6 | dynamic | Yes | 130 | Po17 | Log 1.txt |
| 2 | 1228 | aaaa.905a.218e | dynamic | Yes | 130 | Po18 | Log 1.txt |
| 3 | 1269 | aaaa.5610.0b5d | dynamic | Yes | 45 | Po27 | Log 1.txt |
| 4 | 464 | aaaa.8be7.edfe | dynamic | Yes | 130 | Po17 | Log 1.txt |
| 5 | 1286 | aaaa.0a2e.bca5 | dynamic | Yes | 35 | Po32 | Log 1.txt |
| 6 | 527 | bbbb.56aa.a420 | dynamic | Yes | 90 | Po1 | Log 2.txt |
| 7 | 511 | bbbb.e75c.b8e5 | dynamic | Yes | 90 | Po22 | Log 2.txt |
| 8 | 661 | bbbb.32a7.0563 | dynamic | Yes | 90 | Po1 | Log 2.txt |
| 9 | 525 | bbbb.d228.cf2a | dynamic | Yes | 90 | Po16 | Log 2.txt |
| 10 | 1448 | bbbb.8859.2119 | dynamic | Yes | 90 | Po49 | Log 2.txt |

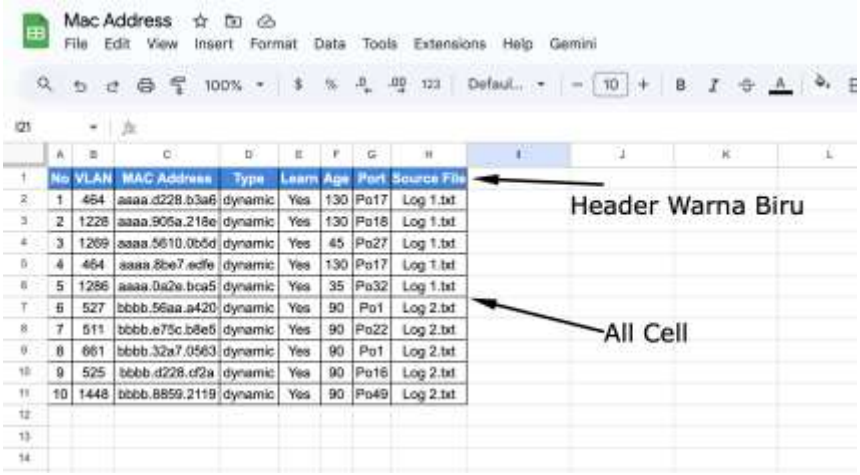
**Figure 4.** Parsing Results

The results indicate that the parsing output matches the expected structure.

### TP-03: Format Automation

The final functionality test focuses on the visualization of the processed data in Google Sheets. The primary objective is to validate the data-clearing function and the bulk writing of new data using the setValues() method. In addition, automated formatting aspects such as header coloring, border application, text

alignment, and automatic column width adjustment (auto-resize) are examined to ensure that the final report is readable and ready for use by network administrators without requiring manual editing. The formatting automation results produced by the executed script are shown below:




| No | VLAN | MAC Address | Type | Learn | Age | Port | Source File |
|---|---|---|---|---|---|---|---|
| 1 | 464 | aaaa.d228.b3a6 | dynamic | Yes | 130 | Po17 | Log 1.txt |
| 2 | 1228 | aaaa.906a.218e | dynamic | Yes | 130 | Po18 | Log 1.txt |
| 3 | 1269 | aaaa.5610.0b5d | dynamic | Yes | 45 | Po27 | Log 1.txt |
| 4 | 464 | aaaa.8be7.edfe | dynamic | Yes | 130 | Po17 | Log 1.txt |
| 5 | 1286 | aaaa.0a2e.bca5 | dynamic | Yes | 35 | Po32 | Log 1.txt |
| 6 | 527 | bbbb.56aa.a420 | dynamic | Yes | 90 | Po1 | Log 2.txt |
| 7 | 511 | bbbb.e75c.b8e5 | dynamic | Yes | 90 | Po22 | Log 2.txt |
| 8 | 661 | bbbb.32a7.0563 | dynamic | Yes | 90 | Po1 | Log 2.txt |
| 9 | 525 | bbbb.d228.cf2a | dynamic | Yes | 90 | Po16 | Log 2.txt |
| 10 | 1448 | bbbb.8859.2119 | dynamic | Yes | 90 | Po49 | Log 2.txt |

**Figure 5.** Format Automation Results

The output confirms that the automated formatting functions operate as expected.

### Efficiency and Performance Analysis

An efficiency analysis is conducted to assess the extent to which the cloud scripting–based automation system optimizes time and resource utilization compared to manual methods. In this functionality test, the system is evaluated using a limited scenario consisting of two log files with a total of ten MAC address entries. Although the data volume in this test is relatively small, the extraction results demonstrate 100% accuracy in mapping heterogeneous data into a systematically structured format. This small-scale testing serves as a proof of concept that the integration of a cloud-based ecosystem is capable of handling remote data acquisition with minimal processing time. However, the actual efficiency gains of the system are expected to become more apparent when applied to large-scale network environments involving dozens or hundreds of log files with thousands of MAC address entries. Under such conditions, manual methods of copying and formatting data would be highly prone to human error and require substantial operational time. The following section presents a comparison of the time duration required for log management using manual and automated approaches.

**Tabel 6.** Duration comparison

| Number of Files | Number of MACs | Manual | Automation |
|---|---|---|---|
| 2 File | 10 Data | 5 Minute | < 3 Second |
| 50 File | 500 Data | 2 Hours | < 15 Second |
| 100 File | 1000 Data | > 4 Hours | < 30 Second |

### Limitations and Scalability Analysis

Following the functionality testing and performance evaluation, several technical limitations and scalability potentials were identified that should be considered for long-term system implementation.

### System Limitations

Although the proposed automation system offers high efficiency, several operational constraints arise from the policies and characteristics of the cloud ecosystem in use:

1. Execution Time Limitations: Google Apps Script enforces a maximum execution time (approximately 6 minutes for free accounts and up to 30 minutes for Google Workspace accounts). If the number of

log files reaches hundreds and file sizes are large, the script may terminate before all data is fully processed.

2. Log Structure Dependency: The parsing algorithm relies heavily on fixed Regular Expression patterns. If network devices (such as switches) operate on different firmware versions that produce varying MAC address table formats, modifications to the Regex logic are required to ensure accurate data extraction.
3. Internet Connectivity: As a cloud-based system, the data acquisition and synchronization processes depend on stable internet connectivity between Google Drive and Google Sheets.

### System Scalability

The system is designed with an architecture that supports further development to accommodate increased workloads:

1. Bulk Data Processing: The use of the setValues() function demonstrates the system's readiness to handle data surges (vertical scalability) without significant performance degradation, as data writing is executed in a single batch operation.
2. Scheduled Automation (Triggers): Operational scalability can be enhanced by implementing time-driven triggers, enabling periodic log monitoring (e.g., hourly execution) without manual intervention. This approach aligns with data-driven intelligent monitoring concepts.
3. Multi-Device Centralization: The system can be extended to scan subfolders from multiple branch offices or buildings, allowing MAC address data from network devices across an entire organization to be consolidated into a single centralized master spreadsheet.

## 5. Conclusion

Based on the results of the implementation and testing phases, it can be concluded that the use of a cloud-based ecosystem through Google Apps Script is highly effective in automating centralized network device log processing. The integration of cloud storage and data processing services successfully replaces conventional log management methods by providing a high degree of flexibility, particularly in handling remote data acquisition without the need for additional physical infrastructure. The application of Regular Expression–based parsing algorithms ensures high accuracy in interpreting semi-structured text formats, allowing critical information such as MAC addresses and interface identifiers to be systematically mapped into modern spreadsheet platforms. Although the current evaluation was conducted on a relatively small scale, the adopted serverless architecture demonstrates significant operational efficiency and shows strong potential to accommodate larger data volumes in the future using a low-cost or zero-cost approach.

Nevertheless, this study is subject to several limitations that should be acknowledged. The testing environment was limited to a specific type of network log format, which may restrict the generalizability of the system to other devices or vendors with different log structures. In addition, performance evaluation under high-frequency log generation and large-scale datasets was not comprehensively assessed. Therefore, future research is recommended to extend the system's compatibility with a wider range of log formats and network platforms, as well as to evaluate scalability and performance under real-time, high-volume conditions. Further studies may also explore the integration of advanced analytical techniques, such as anomaly detection or machine learning–based log analysis, to enhance the intelligence and security monitoring capabilities of the proposed system.

## 6. Reference

[1] K. B. Sowmya And A. Thejaswini, "Systematising Troubleshooting Of Disputes In Network," *International Journal Of Reconfigurable And Embedded Systems (Ijres)*, Vol. 10, No. 1, P. 32, 2021, Doi: 10.11591/Ijres.V10.I1.Pp32-36.

[2] I. Boyagane, O. Katulanda, S. Ranathunga, And S. Perera, "Vue4logs - Automatic Structuring Of Heterogeneous Computer System Logs," *Arxiv*, Vol. Abs/2202.0, 2022.

[3] J. J. Siang, R. R. Rivanka, And A. Wibowo, "Automation Of Daily Access Data Recap Of Employee Data Tables Using Apps Script To Detect Illegal Activities," *Jurnal Teknologi Informatika Dan Komputer*, 2025.

[4] N. Rawindaran, A. Jayal, E. Prakash, And C. Hewage, "Cost Benefits Of Using Machine Learning Features In Nids For Cyber Security In Uk Small Medium Enterprises (Sme)," *Future Internet*, Vol. 13, P. 186, 2021.

[5] C. Macaneata, "Overview Of Security Information And Event Management Systems," *Informatica Economica*, 2024.

[6] W. Lin, J. Ma, T. Li, H. Ye, J. Zhang, And Y. Xiao, "Crptac: Find The Attack Chain With Multiple Encrypted System Logs," *Electronics*, 2024.

[7] A. I. Asry, "Implementation Of Google App Script In Cloud-Based Data Search Application," *Jeat : Journal Of Electrical And Automation Technology*, 2022.

[8] O. Vasenko, V. Yakuba, And I. Havrylov, "Automating Data Analysis In Scientific Research Of Future Education Specialists Using The Google Apps Script Platform," *Professional Education: Methodology, Theory And Technologies*, 2023.

[9] D. K. Anggraeni And T. D. Widajantie, "Peningkatan Efisiensi Administrasi Keuangan Melalui Sistem Penagihan Otomatis: Studi Kasus Pada Pt Java Energy Semesta," *Jurnal Akutansi Manajemen Ekonomi Kewirausahaan (Jamek)*, 2025.

[10] O. Tirschwell And N. J. Horton, "Pivoting The Paradigm: The Role Of Spreadsheets In K-12 Data Science," *Arxiv*, Vol. Abs/2506.0, 2025.

[11] T. Wijaya, "Pemanfaatan Google App Script Dalam Merancang Aplikasi Web Guna Meningkatkan Efisiensi Biaya Perusahaan," *Journal Islamic Global Network For Information Technology And Entrepreneurship*, 2024.

[12] Y. R. Thota, B. R. Bandlapalli, M. Iytha, And T. Nikoubin, "Zero Cost Approach For Nlp Based, Serverless Voicemail Monitoring Automation Pipeline," *2025 Ieee 18th Dallas Circuits And Systems Conference (Dcas)*, Pp. 1–6, 2025.

[13] Y.-C. Chen, "The Framework For Automating Social Media Content Publishing Using Google Apps Script," *Iet Conference Proceedings*, Vol. 2024, No. 28, Pp. 81–83, Sep. 2025, Doi: 10.1049/Icp.2025.0198.

[14] J. P. Duterte, "Technology-Enhanced Learning Environments: Improving Engagement And Learning," *International Journal Of Research And Innovation In Social Science*, Vol. Viii, No. X, Pp. 1305–1314, 2024, Doi: 10.47772/Ijriss.2024.8100111.

[15] M. Szulawski, I. Kaźmierczak, And M. Prusik, "Is Self-Determination Good For Your Effectiveness? A Study Of Factors Which Influence Performance Within Self-Determination Theory," *Plos One*, Vol. 16, No. 9, P. E0256558, Sep. 2021, Doi: 10.1371/Journal.Pone.0256558.

[16] L. J. Ekanayake, D. Ihalage, And Sachith. P. Abyesundara, "Performance Evaluation Of Google Spreadsheet Over Rdbms Through Cloud Scripting Algorithms," *2021 International Conference On Computer Communication And Informatics (Iccci)*, Pp. 1–7, 2021.

[17] R. S. Alejandrino, M. C. G. Diomampo, And J. R. Balbin, "Smart Water Meter With Cloud Database And Water Bill Consumption Monitoring Via Sms And Mobile Application," *2022 Ieee International Conference On Automatic Control And Intelligent Systems (I2cacis)*, Pp. 90–95, 2022.

[18] R. Z. Fitriani, C. Kuncoro, And Y. Kuan, "Internet-Based Remote Setting And Data Acquisition For Fuel Cell," *Sensors And Materials*, 2021.

[19] S. Gulwani, V. Le, A. Radhakrishna, I. Radicek, And M. Raza, "Structure Interpretation Of Text Formats," *Proceedings Of The Acm On Programming Languages*, Vol. 4, Pp. 1–29, 2020.

[20] R. Zhu *Et Al.*, "Sheetmind: An End-To-End Llm-Powered Multi-Agent Framework For Spreadsheet Automation," *Arxiv*, Vol. Abs/2506.1, 2025.