

A structured event logging framework for language model assisted security event analysis

Sirraaj Akhtar

`sirraaj.akhtar@hud.ac.uk`

University of Huddersfield

Saad Khan

University of Huddersfield

Simon Parkinson

Manchester Metropolitan University

Research Article

Keywords:

Posted Date: October 22nd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11190761/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

A structured event logging framework for language model assisted security event analysis

Sirraaj Akhtar^{1*}, Saad Khan¹ and Simon Parkinson²

¹School of Computing and Engineering, University of Huddersfield,
Huddersfield, UK.

²Manchester Metropolitan University, Manchester, UK.

*Corresponding author(s). E-mail(s): sirraaj.akhtar@hud.ac.uk;
Contributing authors: saad.khan@hud.ac.uk; S.Parkinson@mmu.ac.uk;

Abstract

Inconsistent responses and performance degradation are major limitations in LM-based (Language Model) event log literature. Previous literature has used various LM techniques to address this; however, it persists. Therefore, we suggest that the log format is the cause of this limitation and identify several event-log attributes that increase the likelihood of these limitations. This paper introduces a novel LM-friendly event log format with four novel components which address the limitations and expand upon existing logging frameworks: telemetry coverage, suitable fidelity level selection, elimination of parsing as a prerequisite for event log analysis by ensuring logs are consistent and structured at generation, and performing correlation at generation. To evaluate the format, we recorded and extracted the logs produced after several scenarios that span all event log categories and compared performance using multiple LMs and other event log recording formats. We found that performance was consistent across different LMs, whether they were trained initially or after fine-tuning on event log data, and our format significantly improves LM-friendliness when compared to other event log formats.

Keywords: Event Logs, Large Language Models, Cybersecurity

1 Introduction

Our previous research papers [Akhtar et al. \(2026, 2025\)](#) and other LM-based event log research [Mudgal and Wouhaybi \(2023\)](#); [He et al. \(2024\)](#); [Ji et al. \(2025\)](#) demonstrate

that LMs frequently hallucinate and experience performance degradation in various event log tasks. Event log analysis is critical, as the information provided can be used to detect or mitigate threats to systems or networks. If event log analysis is incorrect, organisations are at risk. Furthermore, if remedial actions are performed based on the hallucinated responses, this could be an additional threat to the system. Therefore, LM-based event log analysis tools are unsuitable for real-world deployment unless this limitation is addressed. As LMs can perform event log analysis to a high standard, we suggest that degradation may be because the format of event logs is unsuitable for LMs.

Past research uses various techniques, such as in-context learning and fine-tuning [Xu et al. \(2024\)](#); [Zhong et al. \(2024\)](#); [Zhang et al. \(2025\)](#); [Le et al. \(2025\)](#), to address this limitation. These techniques significantly improve results on various metrics; however, the limitations persist. This suggests that the root cause is unsuitable event log formats rather than the LMs themselves. Furthermore, there are no studies that address this limitation by modifying the recording process or the format of event logs. Consequently, a study to develop and evaluate an LM-friendly event log format is necessary.

To begin, we conducted a thorough literature review to identify the properties of event logs which cause LMs to hallucinate, review other literature based on event log formats, and produce a list of requirements based on our findings to ensure we achieve our objective. We then produced prototypes of our format and developed the tool. Our tool aimed to mitigate the limitations identified by ensuring event log recording was comprehensive (telemetry coverage), context within the log balanced between being sufficient and overwhelming (fidelity-aware logging), eliminating the need for additional steps such as parsing (logs are structured at generation) and correlation (correlation is performed at generation). During empirical testing, we recorded event logs generated after twenty scenarios, which comprised a wide range of event log categories, and sent these to LMs. Finally, we evaluated the tool using a range of custom metrics and compared results against other event log recording mechanisms to determine whether it was more LM-friendly. The code used for implementation, testing and evaluation is publicly available in our [GitHub repository](#).

The novel contributions of our format are:

- **Ensuring that our format has telemetry coverage.** We ensure that event logs from different parts of the system are recorded by one tool locally, excluding noisy logs to ensure operational viability.
- **Selecting the correct fidelity level.** We organise context into three tiers: low, medium and high to balance between providing sufficient information and overwhelming LMs.
- **Eliminating the requirement of parsing.** We guarantee that all logs are structured and consistent by ensuring adherence to set formatting at generation to avoid additional processing.
- **Correlating event logs at generation.** We correlate event logs as they are generated using time-proximity and ID matching to supersede external correlation.

Our paper begins with a literature review where we extracted the requirements for our format, which can be found in Section 2. We then provide our methodology in Section 3 and the implementation details for our format in Section 4. In Section 5, we present and discuss our results, and in Section 6 we discuss potential threats to validity and steps taken to mitigate them. Finally, we conclude our paper and discuss future work in Section 7.

2 Literature Review

2.1 Security, privacy and legislation

To begin, the authors of reports, [Australian Cyber Security Centre \(2024\)](#); [Office of the State Bank Commissioner of Kansas \(2024\)](#) make suggestions relating to the process of recording event logs. In particular, they suggest that they should be stored securely, collected efficiently, and that the format of the logs should be consistent across the system.

A hashing approach is used in research articles, [Rath \(2016\)](#); [Varanda et al. \(2021\)](#); the authors replace sensitive data within logs, which allows the other content to remain untouched for analysis. However, this is limited because it adds an extra step that makes recording event logs take longer. The study, [Aghili et al. \(2025\)](#), expands on this topic by conducting a literature review to establish the attributes of event logs that could breach legislation, and they found that "MAC addresses and file paths" are often disregarded.

2.2 Size and storage

Event logs are generated after every activity in a system; organisations comprise multiple systems, which means their storage consumption is high [He et al. \(2016\)](#). These logs must be analysed to identify threats and mitigate them; however, the format of logs is noisy, and if logs were structured efficiently, storage consumption would be reduced [Chen et al. \(2021\)](#).

2.3 Parsing

2.3.1 General parsing

Parsing is an essential component of event log analysis, as raw event logs are unstructured and unsuitable for event log analysis tools. However, current parsers suffer from several limitations [Zhang et al. \(2023\)](#). Firstly, as event log content and format change frequently, parsers become redundant or require updates [He et al. \(2017a\)](#); [Jiang et al. \(2024\)](#). In addition to this, parsers struggle to understand the content of logs and underperform in real-world scenarios [Zhu et al. \(2019\)](#); [Khan et al. \(2022\)](#); [Fu et al. \(2022\)](#). Furthermore, event log parsing increases the time before event logs are ready for analysis, and online-based parsers compromise the security of the logs as they are hosted externally [Du and Li \(2016\)](#); [He et al. \(2017b\)](#); [Tao et al. \(2022\)](#); [Nedelkoski et al. \(2020\)](#); [Liu et al. \(2022\)](#).

Moving on, traditional parsers could also introduce attributes to the logs, which could result in an increase in the likelihood of hallucinations or time taken. An example is in article [Tovarňák and Pitner \(2019\)](#) where the authors incorporate domain-specific terms into the logs, and as we will discuss in later sections, this could lead to hallucinations. Equally notable is organising logs into groups, which will overwhelm the LLMs because of their limited context windows [Li et al. \(2018\)](#).

2.3.2 LLM-based parsing

LLMs have been shown to require minimal training data to perform analysis; however, as raw logs are large in size, they hallucinate frequently [Beck et al. \(2025\)](#). Current research focuses on in-context learning, but when the format of the logs changes, the prompt will also need to change [Xu et al. \(2024\)](#); [Zhong et al. \(2024\)](#). The research article, [Xiao et al. \(2024\)](#), provides an answer to this limitation by using a cache of logs rather than prompts and examples, but this increases token consumption, which means parsing will take longer and the risk of hallucinations increases.

In addition, study, [Chai et al. \(2024\)](#), highlights that LLM-based parsers do not understand domain-specific terms. Another major limitation is an increase in time with additional techniques such as fine-tuning [Zhang et al. \(2025\)](#); [Le et al. \(2025\)](#); [Ma et al. \(2024\)](#).

2.3.3 Removing the need for parsing

The process of parsing event logs could be removed entirely if the event logs were in a structured format at generation. Current research leverages machine learning algorithms to understand the content of raw logs; however, they note that the presence of domain-specific terms leads to performance degradation [Le and Zhang \(2021\)](#); [Wang et al. \(2021\)](#). Another approach is extracting important attributes from real-world logs [Pi et al. \(2019\)](#), but like parsing, this is an additional step which increases the amount of time taken.

2.4 Correlation

Event log correlation is an essential part of event log analysis because if logs were analysed as individual entities, they may seem benign, but if looked at holistically, then they could clearly indicate malicious activity. However, currently, correlation takes place after the logs are recorded, increasing the time taken [Chuah et al. \(2022\)](#); tools are restricted to a single event log format [Shaukat et al. \(2026\)](#), and there are a large number of false positives, which could be due to a lack of telemetry [Liu et al. \(2019\)](#). If logs were correlated at generation and different types of event logs were correlated at the same time, false positive correlations are less likely to occur.

2.5 Language models

An evaluation study was conducted to determine what factors affect LM-based event log analysis [Mudgal and Wouhaybi \(2023\)](#). In general, LMs perform event log-related tasks well; however, results often decline because logs contain domain-specific terms that LMs cannot interpret.

In the study, [He et al. \(2024\)](#), the authors add context to an event log using an LLM. However, this method is limited because the LLM cannot access the system and therefore cannot access the context. A similar objective is explored in the article, [Huang et al. \(2025\)](#), where the authors specify a list of rules to which logs typically adhere. Despite this, if new formats of logs or different types of logs from those used to generate the rules are sent to the LLM, it will still be unable to analyse them.

Additionally, in the research article, [Ji et al. \(2025\)](#), specific terms that were unlikely to have appeared in the training data of the models are taught through fine-tuning and in-context learning. However, like the two previous articles, the limitations of resource consumption, time taken, and lack of extensibility are also present. These could be addressed if event logs are formatted in a clear manner and do not contain any domain-specific terms.

2.6 Recording event logs

2.6.1 Survey/evaluation articles

There is a lack of research based on determining the correct time to record event logs [He et al. \(2021\)](#) and which attributes define a good event log [Chen \(2019\)](#). Additionally, event logs from different parts of a system are recorded separately, which leads to incorrect analysis, as they are also analysed separately [Chen and Jiang \(2021\)](#); [Batoun et al. \(2024\)](#), and the format of event logs differs across event log types [Studiawan et al. \(2019\)](#).

In the evaluation study [Li et al. \(2023\)](#), the authors highlight that current recording mechanisms do not clearly distinguish between different attributes; the logs are difficult to understand, and context is limited. Moreover, the size of the logs is a major limitation, which renders them unsuitable for LM-based tools in particular. In a research article, [Yao et al. \(2020\)](#), the authors address this by zipping event logs, but this is unsuitable for LMs as they can not decompress them.

2.6.2 Selecting the correct fidelity level

A study based on the event description field of event logs found that current event log descriptions are generic statements and lack context [He et al. \(2018\)](#). However, this is not isolated to this field, but to the log as a whole. An anomaly detection research paper demonstrated that their tool suffered a performance reduction due to a lack of context within event logs. Study [Shkuro et al. \(2022\)](#) addresses this limitation by increasing the amount of metadata within the logs, but it is restricted to business process mining rather than cyber security auditing.

2.7 Research gap

We have identified four gaps from the literature that, if addressed, would make event logs more suitable for LMs. The first gap is using a telemetry approach where event logs are recorded in a consistent format across a system. The second is selecting the correct fidelity level to ensure that events which require less context are at a lower level and more context are at a higher level. The third is removing the need for event log

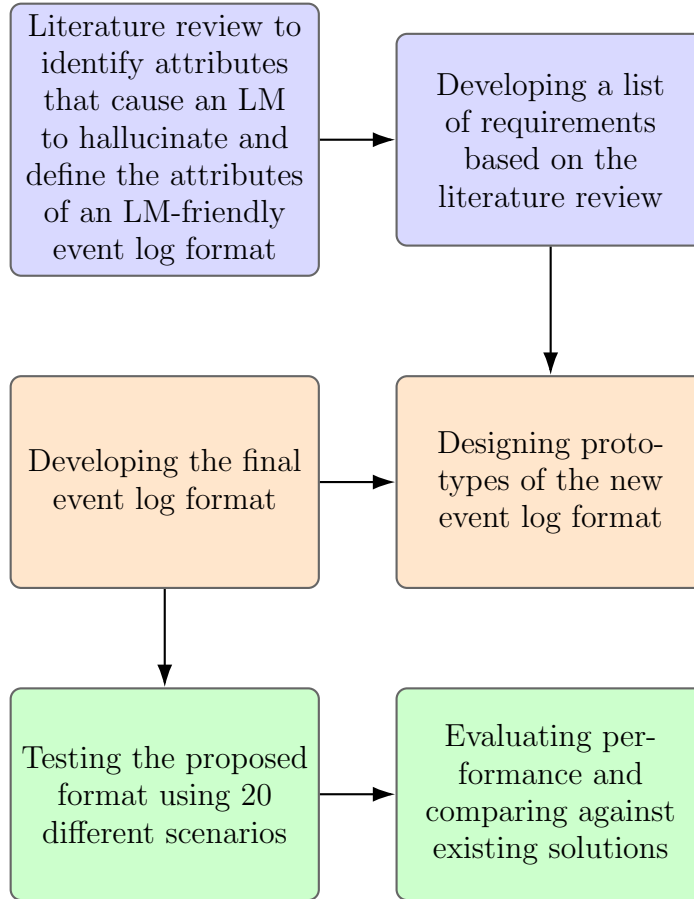


Fig. 1 Overview diagram of the methodology used in this research paper.

parsing by ensuring that event logs are structured at generation. Finally, correlating event logs at generation to reduce the likelihood of false positives.

3 Methodology

3.1 Overview

As discussed in our literature review, the format of event logs is unsuitable for LMs, and this causes them to hallucinate. To ensure our format addressed this, we conducted a thorough review of the literature to identify gaps in event log formats, focusing on attributes that may cause LMs to hallucinate. Following this, we made a list of requirements which an LM-friendly event log format should adhere to and proceeded to develop the tool. Finally, we tested the tool with 20 activities and assessed it using several metrics which assess the responses of LMs on event log analysis tasks. A diagram of this methodology is presented in Figure 1.

Table 1 Summary table of tests based on event type.

Event Type	Activities	No.
Access Control	Password changes, account permission modifications, unauthorised access attempts	3
Network Activity	Network connection/disconnection	1
Application Usage	Application launches, crashes, and execution with administrative privileges	3
System Configuration	Windows Update, backup, and time-zone configuration modifications	3
Data Handling	External storage usage, screenshot capture, connecting an external display and data export activities	4
Security Controls	Firewall, Defender allow/block lists, BitLocker, real-time protection, and Defender settings modifications	6
Total		20

3.2 Requirements analysis

After reviewing the articles, we organised them into three tables. The first table was an extraction table, and the aim was to identify the semantic structure, content, extent of correlation and limitations such as storage and privacy. The second was the "Fixed vs extension" table, which identified whether the formatting adhered to a set standard or could change where required. The final table was the gap analysis table, which compiled a set of limitations so we could clearly identify what we need to address in the development of our format to make it LM-friendly.

3.3 Development

Before developing the tool, we gathered a range of event logs in the original Windows event log format and made a prototype around them that meets the requirements. We sent the prototype and the original to LMs and modified our format until it produced strong responses. Following this, we developed the event log recording tool in stages, iteratively introducing new features that were identified in the requirement analysis stage. Between each version, we tested the new features to ensure they were successfully implemented.

3.4 SLM selection

To ensure we accurately performed the assessment and avoided bias, we compared our format using a range of SLMs based on our previous paper [Akhtar et al. \(2026\)](#). Gemma 3 4b [Team et al. \(2025\)](#) was selected as it was the best-performing model; Bloomz 3b [Muennighoff et al. \(2023\)](#) was selected as a slightly larger model (4B) was used but was unavailable, and Phi-3-mini-4k was selected as BTLM-3 4b was deprecated, but it still meets our criteria as it achieved high results in other research articles [Büchel et al. \(2026\)](#); [Ranjit et al. \(2024\)](#). Furthermore, we fine-tuned Gemma 3 4b and compared it against the original model.

3.5 Testing details

As illustrated in Table 1, we selected twenty tests which corresponded to a range of different event types. This ensured that testing of our event log tool and others considered different scenarios and was thorough. However, it is important to note that this study is an initial validation study with limited scenarios and is restricted to the Windows operating system, and our evaluation will be expanded in future research.

The other event log recording formats that we sourced and that were available for comparison were Sysmon [Mavroeidis and Jøsang \(2018\)](#), SilkETW [Mandiant \(2020\)](#) and Winlogbeat [Elastic NV \(2024\)](#). These formats allowed us to compare our format against real-world event log formats to observe whether our format improves LM-friendliness. There were two objectives: the first was to assess whether the formats successfully record the events (meeting our requirement of telemetry), and secondly to assess whether the recorded logs are LM-friendly. We created a test script which allowed us to record events in our format, the other three formats and the original simultaneously, to ensure that the testing environment was consistent. All activities were executed using the script via PowerShell commands except "connecting an external display" as it is not possible to simulate this.

3.6 Evaluation

To the best of our knowledge, there are no set metrics to determine whether an event log format is LM-friendly. Therefore, we use three custom evaluation metrics from our previous paper [Akhtar et al. \(2026\)](#). The first is Detection Accuracy (DA), which calculates whether the model correctly identified whether the logs indicate a problem. The second is Resolution Relevance (RR), which calculates whether the response is relevant. Finally, Hallucination Rate (HR) calculates the number of hallucinations in the model's responses. In addition, we calculate the time taken to perform analysis of all logs for each of our testing scenarios. As only our tool performs correlation at generation, we compare it against a well-established correlation method, DepCOMM [Xu et al. \(2022\)](#), using pairwise agreement. Pairwise agreement refers to the percentage of logs where our format and DepCOMM correlate the same logs together.

4 Implementation and architectural design

4.1 Overview

Our event log format comprises three components, as portrayed in Figure 2; we will briefly provide an explanation and elaborate on details in the following subsections. The first component is retrieving information from different channels depending on the type of event log. Following this is the processing component, which accepts or denies the logs (filtering out noisy entries), extracts the information we require as context, structures the log according to our requirements, and finally assigns a clear, relevant, natural language description. Parallel to this is our correlation component, where we extract the type of log, check the correlation index to assign it an ID (as logs in our format do not contain IDs), we then observe if the IDs are identical and are within a

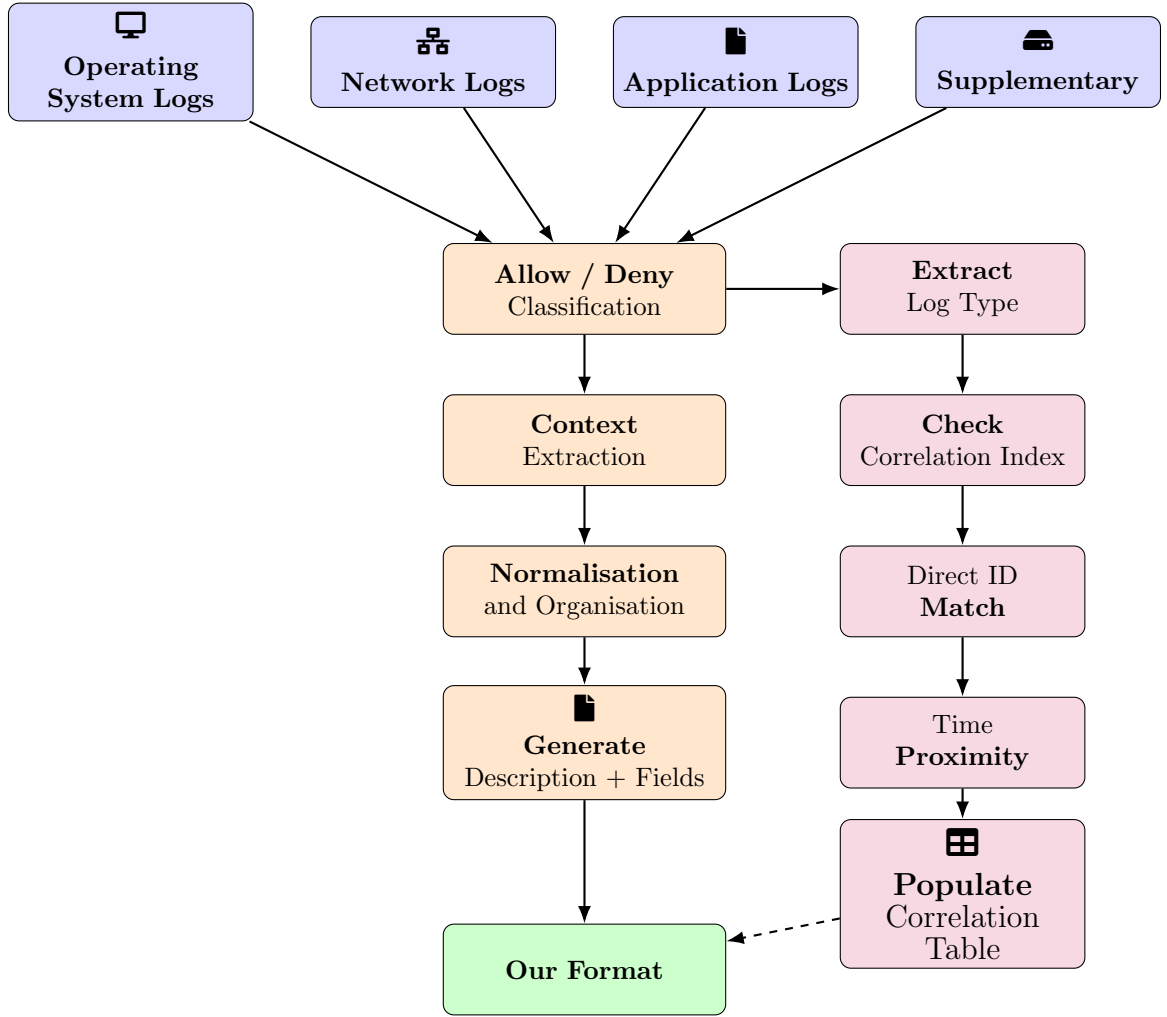


Fig. 2 Overview diagram of the architecture for our format, split into three components: event recording, processing and correlation.

set time proximity, and finally we present the correlated logs in a table for users and administrators to view them.

4.2 Event recording

4.2.1 Capture layer

Event logs are generated constantly, but some events may not occur frequently; due to this, some event log recording tools listen for logs at intervals. This approach means that a threat could remain undetected for a period of time. To address this, we ensure that logs are recorded instantly by using the EventLogWatcher API to constantly

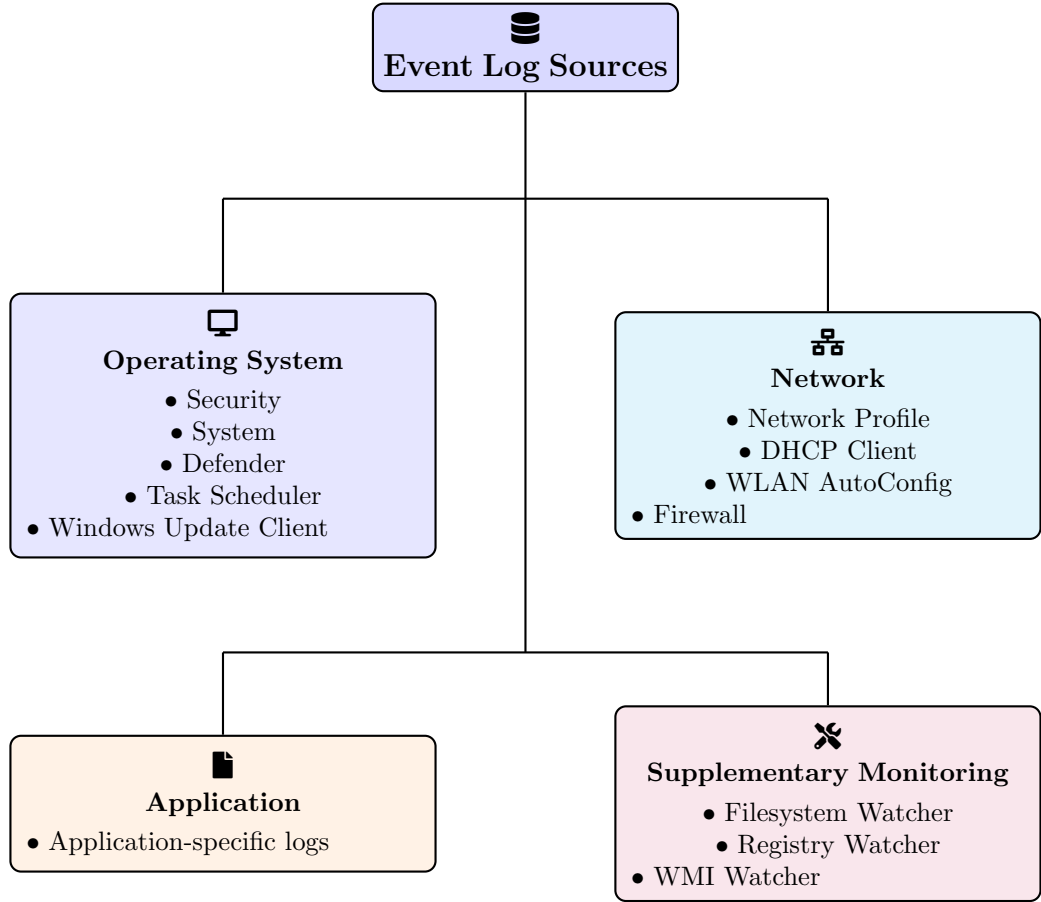


Fig. 3 Event log map detailing channels used to obtain logs for different types of events.

listen for new information in the channels (as illustrated in Figure 3) and if information is found, then the recording process commences. Additionally, certain channels such as security can only be accessed with administrator privileges; therefore, we ensured that our tool can only open in administrator mode. If a channel does not exist on a machine, then our tool overlooks it to avoid premature termination.

4.2.2 Recording new event logs

EventLogWatcher only records certain events, and we found in our preliminary testing that certain events were not recorded because of this. Therefore, we developed additional channels for certain events to ensure that our tool does not overlook them. This was for two different types of events, as illustrated in the supplementary section of Figure 3. The first was for screenshots, where our tool watches the "Screenshots" folder. Recording logs related to screenshots is important, as a malicious actor could use screenshots to capture sensitive data or deploy a tool that uses screenshots to observe the system state while making changes. The second was for logs relating to

Windows Updates, where we watch the registry. Windows Updates are an important part of the system as they address vulnerabilities; if a malicious actor disables the updates, the system is vulnerable.

4.2.3 Allow List

Some event logs are noisy as they represent generic system activities that do not require analysis. Initially, we found that these logs were generated every second in large quantities, and if these were included in our tool, memory-related faults would be likely, and it is certain that LMs would be overwhelmed. To address this, we had two options: an allow list or deny list; we chose to produce an extensive allow list that covers all relevant logs to the best of our ability. A deny list was not suitable as we were unable to form a comprehensive list of noisy logs due to their scale. The allow list also ensured that every entry conforms to our structure, since other formats lack strict formatting, making parsing imperative.

4.2.4 Verification of event log IDs

Event log IDs were processed in the background to ensure that they could be used in later components. However, we found that the documented event log IDs sometimes differ from those produced by the system. In the first case, we performed testing and found that the higher numerical identifier was correct, but in later instances, we found this assumption was incorrect. Therefore, we concluded that we cannot use rules to determine the correct event log ID. To remediate this issue, we triggered the events and monitored which ID was assigned to the logs.

4.3 Processing

4.3.1 Fidelity model

As there are a considerable number of event types, we assign a fidelity level to each type rather than modifying context individually. Events such as state transitions and routine PnP migration are classified as low fidelity, and events in this category only include provider and channel as context values. This is because minimal context is required to resolve any issues implied by them. In the medium tier are events that require 5-10 context fields, such as modifying backup configuration, application warnings and network-related activities. Finally, in the high fidelity tier are events that require a large amount of context, as they are sensitive and critical (between 10-16 fields that contain all relevant information relating to the event). This tier consists of events such as security-related logs and modifications to user accounts and permissions.

4.3.2 Extracting context

The primary attributes that result in inconsistent performance are large codes, domain-specific abbreviations and domain-specific terms, which LMs cannot understand because this information is not typically included in the datasets that were used for training. Therefore, after extracting context, we remove hex codes, message-table references and IDs, and translate important numerations such as LogonType, RIDs,

NTSTATUS codes and elevation types into natural-language statements. Furthermore, we found that events relating to account changes did not include any attributes that we could extract as context. To address this limitation, for the account modification event type, we record numerous attributes using the tool itself (relating to account status) and populate the context field with them.

4.3.3 Formatting event description

To ensure event descriptions were clear, we extracted the generated description provided by the EventLogWatcher API and implemented three rules to preserve LM-friendliness. The first rule is to remove any hex codes, the second is to define domain-specific vocabulary, and the third is to expand abbreviations.

4.4 Correlation architecture

4.4.1 Extraction

Identifier codes are not directly present in our logs to ensure LM-friendliness. Therefore, extraction used two implicit identifiers, process ID (for all events) and GUID (for network events). We implemented a rule to return a default value of "null" if an ID is not returned to mitigate exceptions.

4.4.2 Correspondence checks

Two correspondence checks determine correlation. The first is an exact match between the IDs of two logs within thirty minutes because an event with the same ID could be generated after an unrelated process after time has passed, but if it occurs within a short time span, it strongly indicates that it is related to the initial event. The second is a time-proximity match, which is for event logs that differ in ID but occur within four seconds of each other; it is reasonably assumed that sequential events are related.

4.4.3 Out-of-band correlation

Once event logs are correlated, they are organised into a table that presents the correlated groups and the correlation technique used (direct ID match or time-proximity). At this stage, correlated groups are only presented in the GUI interface to ensure that our operational details are equal in this article, as the other event log recording tools do not perform correlation.

5 Results and discussion

5.1 Results

As our testing spans multiple scenarios, we will only provide summarised results tables using average scores. However, detailed results are available in the results branch of the GitHub [repository](#), accompanying this paper.

Format	DA	RR	HR
Our Format	0.63	0.61	0.00
Original Windows	0.36	0.34	0.00
SilkETW	0.15	N/A	0.00
Winlogbeat	0.35	0.50	0.00
Sysmon	0.47	0.70	0.00

■ Low
 ■ Moderate
 ■ High
 ■ N/A

Fig. 4 Comparison of our format and others across our custom evaluation metrics, using a heatmap with a key to explain colouring.

Table 2 Processing time table to compare results between our format and others.

Event Log Format	Processing Time
Our Format	9 minutes
Event Viewer	10 h 04 min
SilkETW	3 h 59 min
Winlogbeat	8 h 16 min
Sysmon	7 minutes

Model	DA	RR	HR
Gemma 3 4B	0.63	0.61	0.00
Bloomz 3B	0.15	N/A	0.00
Phi-3-mini-4k	0.71	0.66	0.00

■ Low
 ■ Moderate
 ■ High
 ■ N/A

Fig. 5 Heatmap showing results achieved on our evaluation metrics with a key to explain colouring.

5.1.1 Comparison with other event logging tools

Figure 4 illustrates the results of our format and others as a heatmap, using the standard version of Gemma 3 4b and our evaluation metrics. A low score is below 40%, moderate up to 60% and high 60% or higher. Our format achieved the best results in DA and high results in RR. Sysmon achieved the highest RR score but had a moderate DA score, whereas the other three formats all had low scores for DA and RR, except Winlogbeat, which had a moderate RR score. All formats had a 0% hallucination rate. Table 2 highlights the total time taken to analyse all twenty scenarios, and we observe that our format and Sysmon completed analysis in under ten minutes, whereas the others prove time-intensive.

5.1.2 Performance using different SLMs

Figure 5 demonstrates performance when analysing all scenarios with our format on three different SLMs. We observe that Gemma 3 4b and Phi 3 achieved high scores for DA and RR, and again the HR rate was 0% for all models. Bloomz 3b achieved a low DA score, and the RR score could not be calculated. Table 3 details the time taken. All models perform analysis quickly, and the quickest was Bloomz 3b.

Table 3 Table detailing time taken for analysis across different SLMs.

Model	Processing Time
Gemma 3 4B	9 minutes
BLOOMZ 3B	3 minutes
Phi-3-mini-4k	11 minutes

Table 4 Comparison of the base and fine-tuned Gemma 3 4B models; an up arrow means fine-tuning model’s results are better, and a down arrow means the base model performed better.

Metric	Base	Fine-tuned
Detection Accuracy (↑)	0.630	0.700
Resolution Relevance (↓)	0.614	0.609
Hallucination Rate	0.000	0.000
Processing Time (↓)	9 min	17 min

5.1.3 Performance with and without fine-tuning

Table 4 compares scores on our evaluation metrics and total processing time using a fine-tuned version of Gemma 3 4b and the original model. The fine-tuned model achieved an increase in DA, but processing time is nearly double the amount taken by the original.

5.1.4 Impact of correlation component

The impact of the correlated component of our format was assessed by comparing it against the DepCOMM correlation method using pairwise agreement scores. Table 5 highlights that scores fluctuate significantly depending on the scenario, as scenario 6 was an exact match whereas others can drop below 30%.

5.2 Side-by-side comparison

To begin, we provide a side-by-side comparison of logs that were generated in scenario 20 in Figure 6. The original format spans multiple lines, has several identifier codes, abbreviations, domain-specific terms and does not directly record that a major change occurred (changing the configuration of the anti-virus). On the other hand, our format is a single line (was separated in the figure for readability), contains no identifier codes, abbreviations or domain-specific terms, correctly records the issue and provides a detailed description. These factors mean that LMs can clearly understand the activity and provide detailed solutions to the problem without hallucinating, compared to the original, where the probability of hallucinations is high. If a coherent response is produced, the model will incorrectly categorise the log as benign, as it was not provided with any information indicating a critical change was made. In the case of

Table 5 Correlation comparison between our component and DepCOMM using pairwise agreement scores.

Scenario	Pairwise Agreement
1	0.618
2	0.444
3	0.282
4	0.317
5	1.000
6	0.492
7	0.597
8	0.429
9	0.486
10	0.352
11	0.436
12	0.462
13	0.308
14	0.597
15	0.429
16	0.698
17	0.580
18	0.471
19	0.585
20	0.721
Average	0.515

this scenario, this poses a significant risk to the system if this issue is not resolved, as the anti-virus is disabled and therefore malware could be installed.

5.3 Correlation at generation

As tests took place within a small time window, the tool correlated all logs for a scenario together because of the time-proximity rule, and it is reasonable to assume this is correct and that they relate to the scenario. However, we cannot definitively conclude whether our tool performed correlation correctly or which method is better, because the impact of our tool’s correlation is only apparent for scenarios where event logs span a large time window. However, we found that DepCOMM correlated using attributes within the context field and did not consider time proximity or event IDs, which led to incorrect correlation for unrelated events where the same process name appears in the context field.

5.4 Hallucination rate

A major and unexpected finding in this study was that HR was 0% universally, across all experiments (including other event log recording formats). We analysed the outputs from each experiment and also could not identify any hallucinations.

There are two interrelated reasons for this: the first is that we sent a relatively small number of logs to the models compared to other research papers, where testing datasets comprise thousands of entries. The larger entries are likely to exceed the

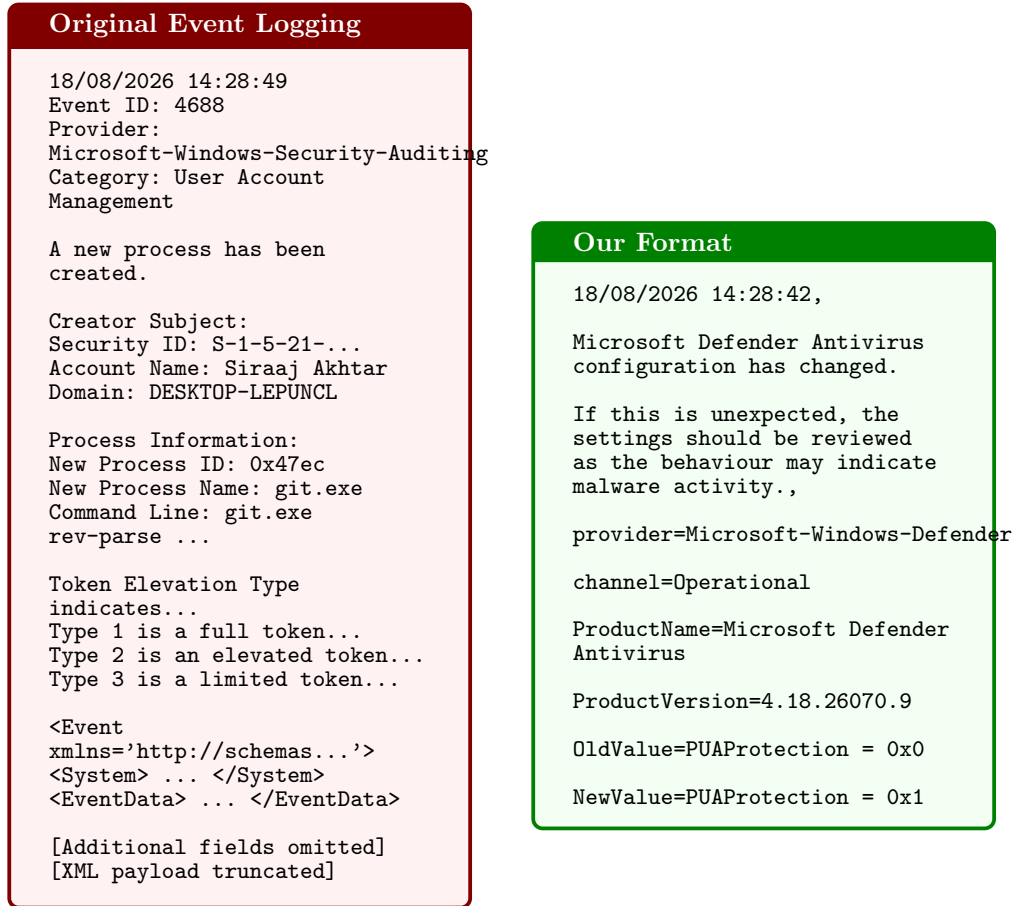


Fig. 6 Side-by-side comparison of two logs from scenario 20 (modifying Windows Defender settings). Long fields and XML content from the original are truncated to facilitate readability.

limited context window of the models and result in hallucinations in other research, but in our scenarios this does not occur, as our entries are of a suitable size.

The second reason could be that in our experiments we processed logs for each scenario individually, and other researchers process all without separation, which may mean that the models fail to disambiguate the logs and hallucinate when analysing them. However, this finding does not imply that all the responses in this paper are correct or relevant, as no single iteration achieves a perfect DA or RR score.

For these two reasons, we cannot correctly determine if hallucinations have been eliminated. In future work, we can send a larger number of event logs, which are of a similar size to the datasets used in other literature, and observe whether the hallucination rate remains at 0%.

5.5 Evaluation of the impact of the telemetry component

When analysing the outputs of the different tests, we found that the original Windows event log format and Sysmon did not record any event logs in 9 of the 20 scenarios. These scenarios were largely related to security logs but did include some network and application logs. This is a major limitation, as LMs have no information about these events and therefore will not produce any correct or relevant responses. Additionally, this renders them unsuitable for any other tool that uses event logs to resolve issues, for the same reasons.

Our results may indicate that events were recorded incorrectly for three scenarios (2, 15, and 17), as DA was 0% and RR and HR could not be recorded; however, this stems from an instrumental error rather than our format or the models used. The scenarios were critical and related to the security of the system, but test accounts were used and appear in the context field. The test harness filters out noise, and the word "test" led to the incorrect assumption that these events were benign noise; as a result, they were never sent to the models. This means that our format recorded event logs for all scenarios successfully regardless of event type.

5.6 Evaluation of the impact of the fidelity component

As we mentioned in previous sections, to select the correct fidelity level for LMs, an event log format should provide sufficient context without overwhelming the models. We split fidelity into three levels based on event type, and our results highlight that this approach was successful, as the RR score is consistently high for our format. This suggests that other formats produce irrelevant responses because the information is either insufficient or overwhelming.

5.7 Consistent results across different models

Bloomz had a low DA score, and RR could not be calculated, but Gemma and Phi had high scores. Processing time was the lowest for Bloomz; however, this is due to abrupt analysis. This suggests that the original Bloomz model was not trained on data relevant to event log analysis and therefore could not perform analysis correctly. Therefore, we conclude that if an LM is not trained either initially or after fine-tuning on event log data, it cannot perform event log analysis correctly regardless of the structure and characteristics of the logs. However, for other formats that are trained on event log data, our format prevents major limitations and significantly improves performance.

5.8 Impact of fine-tuning

This conclusion is further supported by the comparison between the fine-tuned and original version of Gemma 3 4b, as the fine-tuned version has an increase in accuracy whilst remaining relevant. However, processing time increases by nearly 50%. This observation can be attributed to the additional weights added to the model during fine-tuning. Furthermore, all aspects of testing in this paper had no costs except fine-tuning, as API costs when generating the dataset were \$8.96. The difference in

performance is much larger in other studies, which led researchers to conclude that fine-tuning (or other training methods such as in-context learning) is a prerequisite for event log analysis using LMs. However, our study demonstrates that if the format is LM-friendly, training is not necessary. This finding is notable as it reduces time taken, costs and resource requirements, which means that performing event log analysis using LMs is operationally more straightforward.

5.9 LM-friendliness of our tool compared to others

Our results clearly demonstrate that our format is more LM-friendly, as it achieves high scores for all metrics and processes the logs in minutes, whereas the other formats take multiple hours. When logs produced in the format are sent to LMs, responses are correct, relevant, detailed, and consistent. Sysmon does have a higher RR score and processes the logs slightly faster, but this does not mean it is more LM-friendly because, as we mentioned previously, it fails to record critical events, and our results show that the DA score is moderate. Furthermore, processing time would have been much longer than our format if logs were produced for all of the scenarios. These findings highlight that telemetry coverage, selecting the correct fidelity level and ensuring the format does not contain domain-specific terms and abbreviations are essential to ensure that event logs are LM-friendly. Moreover, the significant performance advantage indicates that a clear structure at generation obviates the need for parsing.

6 Threats to validity

In this study, we have two threats to construct validity. The first was that the metrics were calculated using keywords either with a custom map or with terms within input logs; this may mean some aspects of the models' responses were not considered or reflected in the results. This threat can be minimised but cannot be eliminated, as LMs are not deterministic. To address this, we performed a thorough search to identify as many relevant terms as possible. The second threat was a limited number of tests; to address this, our scenarios included a comprehensive set of event log categories.

Additionally, we had an external threat as we did not test our solution on multiple apparatuses or on other operating systems. As our event log format is consistent, the logs produced by machines on the same operating system (Windows) will not differ; therefore, these additional tests were not necessary. However, in future work we can increase generalisability by implementing and evaluating the tool using other operating systems, such as Linux.

7 Conclusion and future work

In conclusion, our study aims to address the limitations of performance degradation and inconsistent responses for LM-based event log analysis by producing a suitable event log format. In our literature review, we identified that the limitations are attributable to unstructured formatting, domain-specific terms, incorrect fidelity levels and other factors. As a result of this, we expand upon previous event log recording

mechanisms with four main contributions to improve LM-friendliness. These were ensuring telemetry coverage, selecting the correct fidelity level, removing the need for parsing and performing correlation at generation. To assess our tool, we performed activities, recorded the event logs generated, sent them to LMs and calculated results using various custom evaluation metrics. We compared results against other formats and found that our format significantly improves LM-friendliness, as event logs were recorded successfully and correctly for all different types of event logs, results were consistently high, responses were consistent and detailed and processing time was significantly reduced.

In future work, we can expand upon this study by implementing the tool in other operating systems and evaluating performance. This will ensure that performance is consistent across environments and not generalised to particular operating systems. Moreover, as we have produced an LM-friendly format and research based on processing event logs and analysing them using LMs is extensive, we can now progress to post-processing solutions.

Supplementary information. The repository for this project is publicly available on [GitHub](#).

Declarations

Funding. The authors received no funding for this work.

Competing interests. The authors have no competing interests to declare.

Ethics approval and consent to participate. Not applicable.

Consent for publication. No human participants were used for this work; therefore, this is not applicable.

Data availability. All data generated in this study are publicly available.

Materials availability. The materials used in this study are publicly available.

Code availability. The source code associated with this work is available on Github.

Author contributions. Siraa Akhtar: Conceptualisation, Methodology, Software, Investigation, Validation, Writing - Original Draft, Writing - Review & Editing.

Saad Khan: Supervision, Methodology, Writing - Review & Editing.

Simon Parkinson: Supervision.

Competing interests. The authors declare no conflict of interest.

References

- Aghili R, Li H, Khomh F (2025) Protecting privacy in software logs: What should be anonymized? *Proceedings of the ACM on Software Engineering* 2(FSE):1317–1338
- Akhtar S, Khan S, Parkinson S (2025) Evaluating prompt engineering for event log parsing with large language models: A comparative study. *TechRxiv*

- Akhtar S, Khan S, Parkinson S (2026) Fine-tuning small language models for solution-oriented windows event log analysis. arXiv preprint arXiv:260506330
- Australian Cyber Security Centre (2024) Best practices for event logging and threat detection. Guidelines, Australian Signals Directorate
- Batoun MA, Sayagh M, Aghili R, et al (2024) A literature review and existing challenges on software logging practices: From the creation to the analysis of software logs. *Empirical Software Engineering* 29(4):103
- Beck V, Landauer M, Wurzenberger M, et al (2025) System log parsing with large language models: A review. arXiv preprint arXiv:250404877
- Büchel J, Chalas I, Acampa G, et al (2026) Analog foundation models. *Advances in Neural Information Processing Systems* 38:61179–61228
- Chai X, Zhang H, Zhang J, et al (2024) Log sequence anomaly detection based on template and parameter parsing via bert. *IEEE Transactions on Dependable and Secure Computing* 22(2):1150–1167
- Chen B (2019) Improving the software logging practices in devops. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, pp 194–197
- Chen B, Jiang ZM (2021) A survey of software log instrumentation. *ACM Computing Surveys (CSUR)* 54(4):1–34
- Chen Z, Liu J, Gu W, et al (2021) Experience report: Deep learning-based system log analysis for anomaly detection. arXiv preprint arXiv:210705908
- Chuah E, Jhumka A, Malek M, et al (2022) A survey of log-correlation tools for failure diagnosis and prediction in cluster systems. *IEEE Access* 10:133487–133503
- Du M, Li F (2016) Spell: Streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, pp 859–864
- Elastic NV (2024) Winlogbeat reference
- Fu Y, Yan M, Xu J, et al (2022) Investigating and improving log parsing in practice. In: Proceedings of the 30th ACM joint european software engineering conference and symposium on the foundations of software engineering, pp 1566–1577
- He M, Jia T, Duan C, et al (2024) Llmelog: An approach for anomaly detection based on llm-enriched log events. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE), IEEE, pp 132–143
- He P, Zhu J, He S, et al (2017a) Towards automated log parsing for large-scale log data analysis. *IEEE Transactions on Dependable and Secure Computing* 15(6):931–944

- He P, Zhu J, Zheng Z, et al (2017b) Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE international conference on web services (ICWS), IEEE, pp 33–40
- He P, Chen Z, He S, et al (2018) Characterizing the natural language descriptions in software logging statements. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp 178–189
- He S, Zhu J, He P, et al (2016) Experience report: System log analysis for anomaly detection. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE), IEEE, pp 207–218
- He S, He P, Chen Z, et al (2021) A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* 54(6):1–37
- Huang X, Zhang T, Zhao W (2025) Logrules: Enhancing log analysis capability of large language models through rules. In: Findings of the Association for Computational Linguistics: NAACL 2025, pp 452–470
- Ji Y, Liu Y, Yao F, et al (2025) Adapting large language models to log analysis with interpretable domain knowledge. In: Proceedings of the 34th ACM International Conference on Information and Knowledge Management, pp 1135–1144
- Jiang Z, Liu J, Huang J, et al (2024) A large-scale evaluation for log parsing techniques: How far are we? In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp 223–234
- Khan ZA, Shin D, Bianculli D, et al (2022) Guidelines for assessing the accuracy of log message template identification techniques. In: Proceedings of the 44th international conference on software engineering, pp 1095–1106
- Le VH, Zhang H (2021) Log-based anomaly detection without log parsing. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp 492–504
- Le VH, Xiao Y, Zhang H (2025) Unleashing the true potential of semantic-based log parsing with pre-trained language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), IEEE, pp 975–987
- Li Z, Davidson M, Fu S, et al (2018) Converting unstructured system logs into structured event list for anomaly detection. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, pp 1–10
- Li Z, Chen AR, Hu X, et al (2023) Are they all good? studying practitioners’ expectations on the readability of log messages. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp 129–140

- Liu J, Zhu J, He S, et al (2019) Logzip: Extracting hidden structures via iterative clustering for log compression. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp 863–873
- Liu Y, Zhang X, He S, et al (2022) Uniparser: A unified log parser for heterogeneous log data. In: Proceedings of the ACM Web Conference 2022, pp 1893–1901
- Ma Z, Chen AR, Kim DJ, et al (2024) Llmpraser: An exploratory study on using large language models for log parsing. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp 1–13
- Mandiant (2020) Silketw: C# wrapper for event tracing for windows. GitHub
- Mavroeidis V, Jøsang A (2018) Data-driven threat hunting using sysmon. In: Proceedings of the 2nd International Conference on Cryptography, Security and Privacy (ICCSP), pp 104–110, <https://doi.org/10.1145/3199478.3199490>
- Mudgal P, Wouhaybi R (2023) An assessment of chatgpt on log data. In: International Conference on AI-generated Content, Springer, pp 148–169
- Muennighoff N, Wang T, Sutawika L, et al (2023) Crosslingual generalization through multitask finetuning. arXiv preprint arXiv:221101786
- Nedelkoski S, Bogatinovski J, Acker A, et al (2020) Self-supervised log parsing. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 122–138
- Office of the State Bank Commissioner of Kansas (2024) Cybersecurity resources: Logging fact sheet. Fact sheet, Office of the State Bank Commissioner, Topeka, KS
- Pi A, Chen W, Zeller W, et al (2019) It can understand the logs, literally. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, pp 446–451
- Ranjit M, Srivastav S, Ganu T (2024) Radphi-3: Small language models for radiology. arXiv preprint arXiv:241113604
- Rath C (2016) Usable privacy-aware logging for unstructured log entries. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), IEEE, pp 272–277
- Shaukat SU, Khan S, Parkinson S (2026) Event log correlation for multi-step attack detection. Security and Privacy 9(1):e70151
- Shkuro Y, Renard B, Singh A (2022) Positional paper: Schema-first application telemetry. ACM SIGOPS Operating Systems Review 56(1):8–17

- Studiawan H, Sohel F, Payne C (2019) A survey on forensic investigation of operating system logs. *Digital Investigation* 29:1–20
- Tao S, Meng W, Cheng Y, et al (2022) Logstamp: Automatic online log parsing based on sequence labelling. *ACM SIGMETRICS Performance Evaluation Review* 49(4):93–98
- Team G, Kamath A, Ferret J, et al (2025) Gemma 3 technical report. arXiv preprint arXiv:250319786
- Tovarnák D, Pitner T (2019) Normalization of unstructured log data into streams of structured event objects. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, pp 671–676
- Varanda A, Santos L, Costa RLdC, et al (2021) Log pseudonymization: Privacy maintenance in practice. *Journal of Information Security and Applications* 63:103021
- Wang Q, Zhang X, Wang X, et al (2021) Log sequence anomaly detection method based on contrastive adversarial training and dual feature extraction. *Entropy* 24(1):69
- Xiao Y, Le VH, Zhang H (2024) free: Towards more practical log parsing with large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, pp 153–165
- Xu J, Yang R, Huo Y, et al (2024) Divlog: Log parsing with prompt enhanced in-context learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp 1–12
- Xu Z, Fang P, Liu C, et al (2022) Depcomm: Graph summarization on system audit logs for attack investigation. In: 2022 IEEE symposium on security and privacy (SP), IEEE, pp 540–557
- Yao K, Li H, Shang W, et al (2020) A study of the performance of general compressors on log files. *Empirical Software Engineering* 25(5):3043–3085
- Zhang C, Xu W, Liu J, et al (2025) Semanticlog: Towards effective and efficient large-scale semantic log parsing. *IEEE Transactions on Software Engineering*
- Zhang T, Qiu H, Castellano G, et al (2023) System log parsing: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35(8):8596–8614
- Zhong A, Mo D, Liu G, et al (2024) Logparser-llm: Advancing efficient log parsing with large language models. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp 4559–4570
- Zhu J, He S, Liu J, et al (2019) Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software

Engineering in Practice (ICSE-SEIP), Ieee, pp 121–130