ファイルの拡張子と中身の有無によるフィルタリングと情報エントロピーをもちいたログの特定

三上 翔太[1]　高橋 風太[2]　串田 高幸[1]

概要：ログはシステムの障害対応に使用される．ログの管理や検索には Elasticsearch を使用することがある．その際，ログファイルを収集するために Filebeat を使用し，収集したログは Logstash で変換され，Kibana で可視化する．Filebeat では収集対象のログファイルのディレクトリ指定を行う必要があり，ログファイルがシステム内のどこにあるのかを探す必要がある．しかし，/var/log ディレクトリ以外かつ，拡張子が.log 以外のログファイルの特定には時間がかかる．そこでログファイルが VM 内のどこにあるのかを特定する手法を提案する．提案手法では，ファイルの情報エントロピーをもちいてログファイルの特定を行う．ファイルの拡張子による除外を行った後，ファイルの各バイト値の出現頻度から情報エントロピーを計算し，情報エントロピー値が提案手法の基礎実験より算出した閾値 (0〜2 と 5〜6) の範囲にあるファイルをログファイルとする．評価結果として，ログファイル検出率は，MySQL では 10 個のログファイルのうち 9 つのファイルであり 90 %，Apache と SQLite では 7 つ，HAProxy と Redis では 6 つのログファイルのうちすべてのログファイルをしたので 100 %となった．ログファイル未検出率は，MySQL では 10 個のログファイルのうち 1 つのファイルであり 10 %，Apache と SQLite では 7 つ，HAProxy と Redis では 6 つのログファイルのうち未検出のログファイルはなかったので 0 %となった．ログファイル誤検出率は，MySQL では検出したファイル 29 個のうち非ログファイルは 20 個であり約 68.97 %，Apache では検出したファイル 7 個のうち非ログファイルは 7 個であり 0 %，SQLite では検出したファイル 10 個のうち非ログファイルは 3 個であり 30 %，HAProxy では検出したファイル 15 個のうち非ログファイルは 9 個であり 60 %，Redis では検出したファイル 8 個のうち非ログファイルは 2 個であり 25 %となった．

1. はじめに

背景

ログとはシステムに関する活動を時系列で記録している [1–3]．例えば，ログファイル内はタイムスタンプ，イベントまたはアクションの説明，システム情報，エラー情報で構成されている [4,5]．ログファイルは，オペレーティングシステム，データベース，ソフトウェアアプリケーションによって生成される [6]．ログ分析は，システム監視，トラブルシューティング，セキュリティ管理でもちいられる [7,8]．また，ログファイルにはイベントやシステム情報が記録されており，エンジニアはログを使ってシステムを把握し，システム障害を識別できる [9,10]．システムの大規模化と複雑化に伴い，生成されるログの量は増加する [11–13]．システムは人間が実際に処理できる量を上回るログエントリを生成するため，手動での分析は時間がかかる [14,15]．そのため，処理時間の短縮と分析精度の向上を両立したログファイルの管理と分析が求められている [16]．また，ログの収集，保存，検索，分析の一連のログ管理プロセスを実行するための手法が必要とされている．この一連の流れは ELKStack をもちいて行われる．ELKStack は Elasticsearch，Logstash，Kibana，Filebeat で構成されており，仮想ハードウェア環境で実行できる [17]．ELKStack はログ分析に使用され，ログ分析に役立つ機能が備わっている [18]．ログの解析や構造化には Logstash，ログの保存や検索には Elasticsearch，ログの可視化には Kibana, ログの収集には Filebeat を使用する [19]．Elasticsearch とは膨大なデータをリアルタイムで処理できる検索エンジンである [20,21]．Logstash とはログデータを取り込み，変換し，他の場所に送信するシステムである [22,23]．Kibana とはデータの分析，表示，検索に使用される分析ツールである [24]．Filebeat とは Logstash にデータを転送するためのログシッパーである [25,26]．Filebeat では収集するログ

[1] 東京工科大学コンピュータサイエンス学部
クラウド・分散システム研究室
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学院バイオ・情報メディア研究科
コンピュータサイエンス専攻
クラウド・分散システム研究室
〒 192–0982 東京都八王子市片倉町 1404-1

ファイルのディレクトリの指定を行う必要がある．

課題

課題は管理者がログファイルの特定を行う際に，/var/log ディレクトリ以外かつ，拡張子が.log 以外のログファイルを特定するのに時間がかかることである．また，特定したログファイルがログファイルであるか判定に時間がかかる．例として，Ubuntu24.04 に Mysql(version 8.0.39) をインストールした際の，MySQL に関するログファイルの探索時間は約 7 分 56 秒であった．

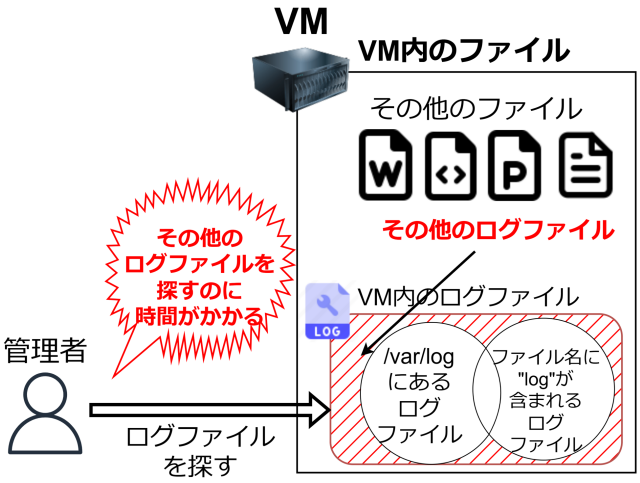

図 1 VM 内からログファイルを探す際の流れ

図 1 は，管理者が仮想マシン (以下 VM) 内のログファイルを探す流れを表している．/var/log ディレクトリにあるログファイルは，/var/log ディレクトリの直接参照で特定できる．ファイル名に.log が含まれているログファイルは find コマンドをもちいて探すことができる．しかし，どちらにも該当しないログファイルを探す際には VM 内全体を探す必要があるため時間がかかる．したがって，ログデータを手動で分析することは困難となっている [11].

各章の概要

第 2 章では，本稿の関連研究について述べる．第 3 章では，本稿の課題について解決するための提案方式について述べる．第 4 章では，提案した手法の実装について述べる．第 5 章では，評価実験として実験内容と実験結果と分析について述べる．第 6 章では，提案手法についての議論を述べる．最後に，第 7 章にて結論を述べる．

2. 関連研究

構造化テキストからエントロピー重み付け手法をもちいてテキスト内の文書の出現頻度と分布を考慮したキーワードや専門用語を抽出する研究を行っている論文がある [27]. 構造化テキストの各モジュールの情報量の違いに着目し，エントロピー重み付け手法をもちいて各モジュールがキーワード抽出にどの程度寄与しているかを評価している．本稿では情報エントロピーをファイル全体のバイト列の特徴量として使用してログファイルを特定するのに対し，テキストの構造的な特徴を活かしたキーワードの抽出に焦点を当てているため，ログファイルの特定は行っていない．

Linux OS イベントログの自動解析にもとづいてセキュリティイベントを検出する方法を提案している論文がある [28]．提案手法は，イベントログを配列に読み込み，k-means クラスタリングで構造が 98 %以上一致したログをグループ化し，25 個の正規表現パターンでイベントを分類している．Linux イベントログのみを対象としており，本稿が目的とするシステム全体からのログファイル特定は行っていない．

正規表現によるログ解析システムの構築に関する研究を行っている論文がある [29]．ログフォーマットの記述とログ内容の解析を分離するアプローチを採用している．XML を使用してログフォーマットの記述とログデータ項目の属性記述のためのスクリプトドキュメントを作成し，50 個の正規表現パターンをログ解析の設定ファイルとして使用している．正規表現によるログ解析システムの構築に焦点を当てており，入力がログファイルであることを前提としているため，ログファイルの特定は行っていない．

3. 提案方式

本稿では，情報エントロピーをもちいてログファイルを識別する手法を提案する．情報エントロピーは情報の予測不可能性を表す尺度である [30]．情報エントロピーは，値が大きいほど情報の不規則性が高いことを示し，値が小さいほどには情報に規則性があり予測が容易であることを示す．ログファイルは規則性の高い記述形式で構成されているため，情報エントロピーの値は低くなる．この特性を活用し，本稿ではファイルの情報エントロピーを計算することでログファイルを識別する手法を提案する．提案手法は，ファイルフィルタリングフェーズ，情報エントロピー計算・ログ判定フェーズの 2 つのフェーズから構成される．

提案方式

提案手法は，ファイルフィルタリングフェーズ，情報エントロピー計算・ログ判定フェーズの 2 つのフェーズから構成される．ファイルフィルタリングフェーズでは，特定の拡張子を持つファイルや空ファイルを除外することで，分析対象を絞り込む．情報エントロピー計算・ログ判定フェーズでは，残ったファイルに対して情報エントロピーを計算し，その値に基づいてログファイルかどうかを判定する．

ファイルフィルタリングフェーズ

このフェーズでは，分析対象のファイルから拡張子の種類とファイルの中身の有無によってファイルを除外する．

表 1　除外する拡張子一覧

分類	拡張子
システム・ライブラリ	.so, .bin, .db, .service, .wants, .d, .successful, .target, .lock
プログラミング	.pm, .pl, .py, .cgi, .pod, .ix
ドキュメント	.md, .html, .copyright, .md5sums, .shlibs, .triggers, .symbols, .templates, .postinst, .postrm, .preinst, .prerm, .config, .ini
圧縮・アーカイブ	.gz, .zip, .tar, .bz2, .xz
メディア・データ	.gif, .png, .jpg, .jpeg, .ico, .csv, .def, .dic
その他設定	.sources, .fallback, .conffiles

表 1 は本稿の提案手法で分析対象から除外した拡張子一覧である．表 1 のファイルは拡張子でログファイルではないと判断ができるため，分析対象から除外した．ログファイルではないファイルの例として，システム・ライブラリに付随する拡張子や，プログラミング言語に付随する拡張子があげられる．また，データサイズが 0 のファイルは空ファイルとして除外する．

情報エントロピー計算・ログ判定フェーズ

このフェーズでは，フィルタリングで除外されなかったファイルに対して情報エントロピーの計算とログファイルの判定を行う．バイト値の出現頻度をカウントし，式 (1) に示す情報エントロピーを計算する．

$$H = -\sum_{i=1}^{n} p_i \log_2(p_i) \qquad (1)$$

- H：情報エントロピー
- p_i：各バイト値 i の出現確率
- n：異なるバイト値の種類数

p_i は各バイト値の出現確率を表す．n は異なるバイト値の種類数を表す．ログファイルは一定の形式で記録される特徴を持つため，情報エントロピー値が低くなる性質を持つ．この特性を利用し，情報エントロピー値を算出することでログの判定を行う．具体的な判定基準については，提案手法の基礎実験の結果にもとづいて決定する．

図 2 は，情報エントロピーの計算例を示している．10 文字からなるサンプルデータ”AABACACBBA”に対して，まず各文字の出現回数をカウントする（A:5 個，B:3 個，C:2 個）．次に，文字列の総数 10 で各出現回数を割ることで，各文字の出現確率 p_i を算出する（p(A)=0.5，p(B)=0.3，p(C)=0.2）．図 2 ではバイト値の種類が 3 種類なので n は 3 となる．最後に，これらの確率を式 (1) に代入することで，このデータの情報エントロピー H=1.485 が得られる．

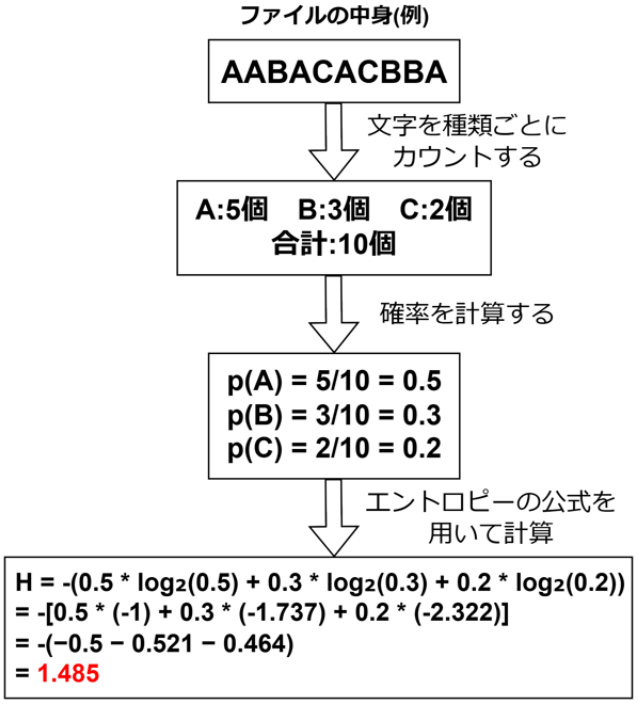

図 2　情報エントロピーの計算例

提案手法の基礎実験

基礎実験では，ミドルウェアがインストールされた VM とインストールされていない VM の差分を取得し，差分として取得したファイルの情報エントロピー値を算出した．実験環境として，以下の仕様の VM を 6 台用意した．

- CPU：2 コア
- メモリ：4GB
- ストレージ：25GB
- OS：Ubuntu24.04

差分を取得するために用意したミドルウェアは以下の通りである．

表 2　ミドルウェアのバージョンとファイル数

ミドルウェア	バージョン	ファイル数 (個)
MySQL	8.0.39	610
Apache	2.4.58	742
SQLite	3.45.1	19
HAProxy	2.8.5-1ubuntu3.1	101
Redis	7.0.15	70

表 2 は，ミドルウェアのバージョンとファイル数を表している．ファイル数は MySQL で 610 個，Apache で 742 個，SQLite で 19 個，HAProxy で 101 個，Redis で 70 個である．差分を取得するために実行したコマンドは下記の通りである．

(1)　差分を取得する両方の VM をアップデートする

```
1 sudo apt update
2 sudo apt upgrade
```

(2) 差分を取得する

```
find / \( -type f -o -type d \) ! -path "/proc/*" ! -path "/sys/*" ! -path "/dev/*" ! -path "/run/*" 2>/dev/null | sort > app_list.txt

ssh ユーザ名@VM名 "find / \( -type f -o -type d \) ! -path \"/proc/*\" ! -path \"/sys/*\" ! -path \"/dev/*\" ! -path \"/run/*\" 2>/dev/null" | sort > normal_list.txt
```

(3) 差分をファイルに出力

```
comm -13 normal_list.txt app_list.txt > app_sabun_files.txt
```

(1) のコマンドでは，パッケージリポジトリの更新とシステムの更新を行っている．これらの更新を行うことで，両方の VM のシステムファイルを最新のバージョンに揃えることができ，OS のアップデートによる差分を排除することができる．(2) のコマンドでは，find コマンドをもちいてプロセス情報やデバイス情報を除くすべてのファイルとディレクトリのリストを取得し，2 つの VM のリストを作成する．(3) のコマンドでは，comm コマンドをもちいて 2 つのリストの差分を抽出し，ファイルに出力する．(1), (2), (3) のコマンドをもちいることによって，ミドルウェアのインストールによって追加されたファイルの差分の抽出を行うことができる．

抽出された差分ファイルには，1 行ごとに差分として検出されたファイルパスが記述される．

基礎実験の結果を分析するにあたり，正解データとしてのログファイルを表 3 に定義した．各ファイルの内容を手動で確認し，表 3 のいずれかの特徴を確認したファイルをログファイルとして判定した．

表 3 は，ログファイル判定基準の一覧を示している．表 3 のいずれかの判定基準に該当する特徴を確認したファイルはログファイルとして判定し「,1」，該当しない場合は「,0」とした．この数値をもちいて，プログラム内でログファイル検出率 (情報エントロピー値ごとのファイルのうち，ログファイルの検出数) の計算を行った．ログファイルの判定は以上の手順で行った実験の結果を図 3 に表す．

図 3 は，5 種類のミドルウェア（MySQL, Apache, SQLite, HAProxy，Redis）がインストールされた環境で収集した差分ファイルについて，情報エントロピー値ごとのログファイル検出率を示している．情報エントロピー値 0〜2 と 5〜6 の範囲において，それぞれのミドルウェアでログファイルがある．この結果から，情報エントロピー値 0〜2 と 5〜6 を閾値とする．

表 3 ログファイル判定基準一覧

分類	判定基準
プロセス状態	start, stop, restart, shutdown, launch, terminate, running, suspended, killed, executed, initialized, finalized, booting, rebooting
実行結果・警告	success, failed, error, warning, info, debug, notice, critical, alert, emergency, fatal, panic
接続状態	connected, disconnected, timeout, established, refused, listening, accepted, closed, bound, unbound
時間情報	タイムスタンプ（年月日時分秒）と共にイベントが記録
ネットワーク情報	IP アドレス，ポート番号，プロトコル名（TCP/UDP/HTTP 等）
システム操作	ユーザ ID，プロセス ID，権限変更，認証，アクセス制御
リソース状態	メモリ使用量，CPU 使用率，ディスク容量，負荷状態

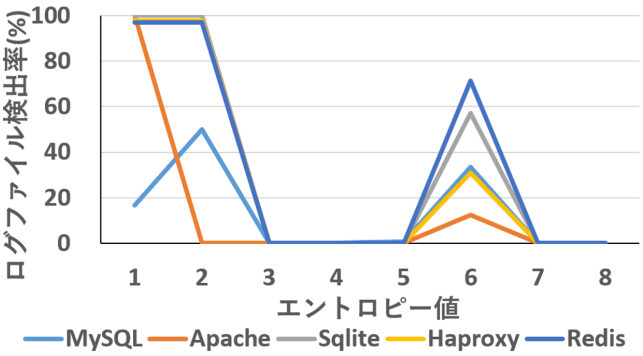

図 3 各ミドルウェアにおける情報エントロピー値とログファイル検出率の関係

ユースケース

本稿では，VM のログファイルを収集するユースケースとする．ログの管理は Elasticsearch で行う．その際，VM からのログの収集では Filebeat を使用するため，Filebeat でのディレクトリの指定が必要になる．その際，ログファイルのディレクトリは管理者が特定する．そこで提案ソフトウェアをもちいてログファイルを特定する．特定したディレクトリを管理者が Filebeat で指定し，収集する．収集したログは Elasticsearch に集約され，Kibana を通じて可視化，分析される．

図 4 は VM のログを Elasticsearch に送信するまでの流れを表している．まず，(1) で管理者がログファイル収集先で提案ソフトウェアを実行する．(2) で提案ソフトウェアがログファイルの特定を行う．(3) で提案ソフトウェアのログファイルの特定結果が出力され，(4) で管理者に結果が表示される．(5) で提案ソフトウェアの結果をもとに管理者が Filebeat にて管理者がログファイルのファイルパスの指定を行う．

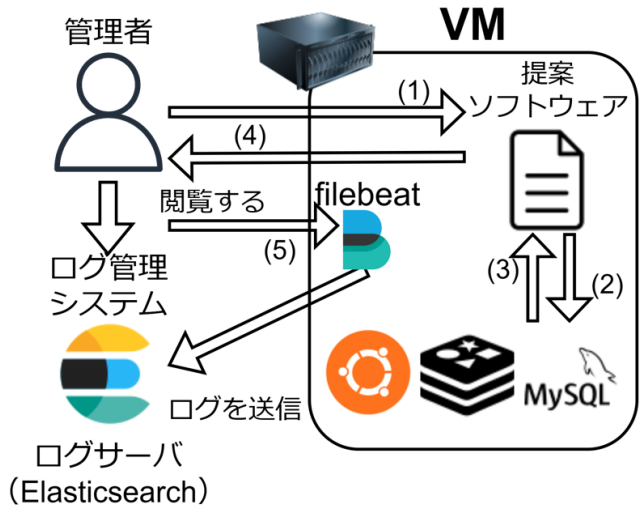

図 4　ログを収集する際の流れ

4. 実装

ソフトウェアの作成には，開発言語である Python3.9 を使用した．開発したソフトウェア (LF-search) の機能と処理の流れについて説明する．

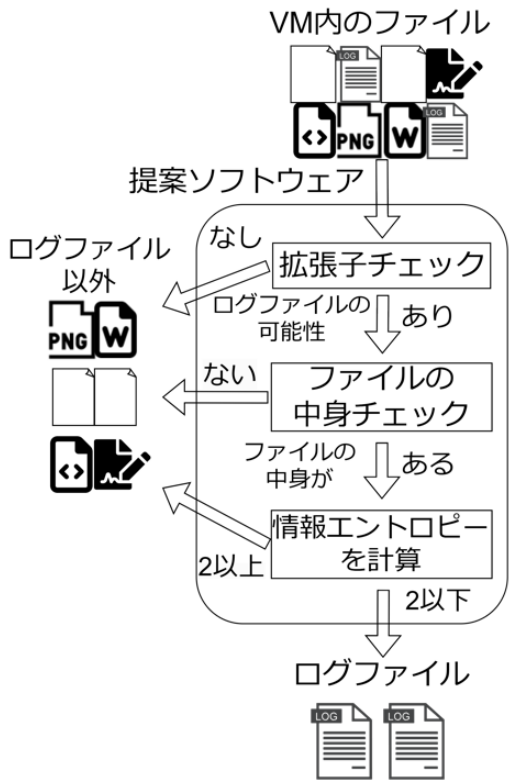

図 5　LF-search の概要

図 5 は，提案ソフトウェアの概要を表している．VM 内のファイルを入力とし，拡張子チェック，ファイルの中身チェック，情報エントロピー計算の処理を経てログファイルの判別を行う．

拡張子チェック

ファイルの拡張子をチェックする．表 1 に示した拡張子を持つファイルは，明確にログファイルではないと判断できるため除外する．また，空ファイルも処理対象から除外する．

ファイルの中身チェック

拡張子チェックを通過したファイルに対して，ファイルのデータサイズの確認を行う．この段階では，ファイルがアクセス可能で，かつ読み取り可能であることを確認する．プログラムは root 権限で実行されることを前提とし，権限による読み込みエラーを防止している．

情報エントロピー計算

ファイルの中身に対して情報エントロピーを計算する．バイト出現頻度から情報エントロピーを算出する．提案手法の基礎実験の結果から，情報エントロピー値が 0〜2 と 5〜6 のファイルをログファイルと判定する．

5. 評価実験

実験環境

評価実験は，第 3 章の提案手法の基礎実験で述べた実験環境と同一の環境で実施した．

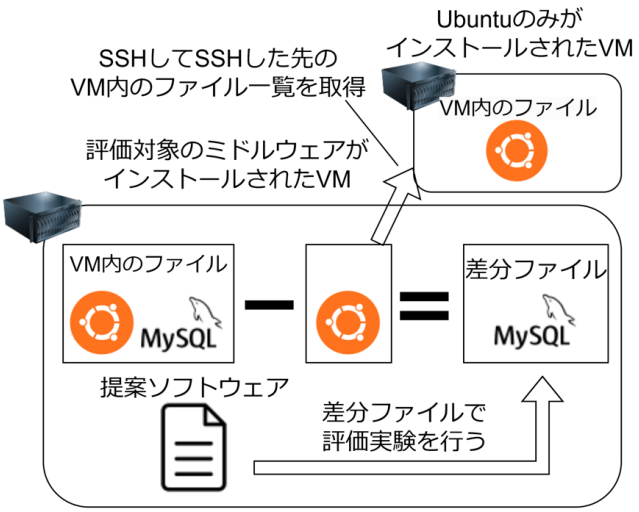

図 6　評価実験の環境構成

図 6 は評価実験の環境構成を示している．実験では 2 台の VM を用意し，1 台目の VM には Ubuntu24.04 と評価対象のミドルウェアをインストールし，2 台目の VM には Ubuntu24.04 のみをインストールした．例として，図 6 での評価対象のミドルウェアは MySQL(version 8.0.39) としている．評価対象のミドルウェアがインストールされた VM から SSH を使用して Ubuntu24.04 のみがインストールされた VM 内のファイル一覧を取得する．その後，評価対象の VM のファイル一覧との差分を取得し，その差分ファ

イルに対して提案ソフトウェアをもちいて評価実験を実施した．

実験結果と分析

評価実験では，5 種類のミドルウェア（MySQL，Apache，SQLite，HAProxy，Redis）に対してログファイル検出率とログファイル未検出率とログファイル誤検出率を評価した．ログファイル検出率は，全ログファイル数のうちの提案ソフトウェアによるログファイル検出数を示す．ログファイル未検出率は，全ログファイルのうちの提案ソフトウェアによるログファイル未検出数を示す．ログファイル誤検出率は，提案ソフトウェアで検出したファイルのうちの非ログファイル数を示す．

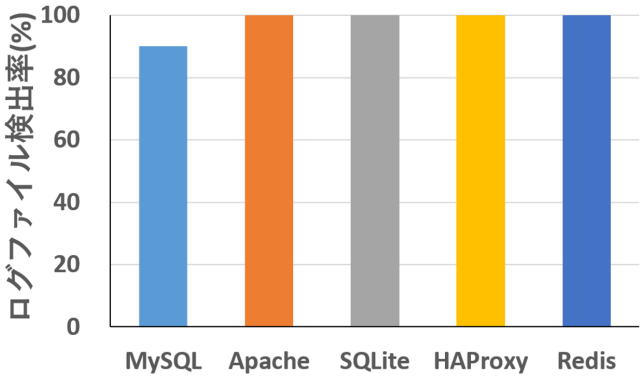

図 7　各ミドルウェアのログファイル検出結果

図 7 は各ミドルウェアにおけるログファイル検出率を示している．MySQL ではログファイル 10 個のうち 9 つのログファイルを検出し，ログファイル検出率は 90 %となった．Apache では全 7 つ，SQLite では全 7 つ，HAProxy では全 6 つ，Redis では全 6 つのログファイルを検出し，ログファイル検出率は 100 %となった．

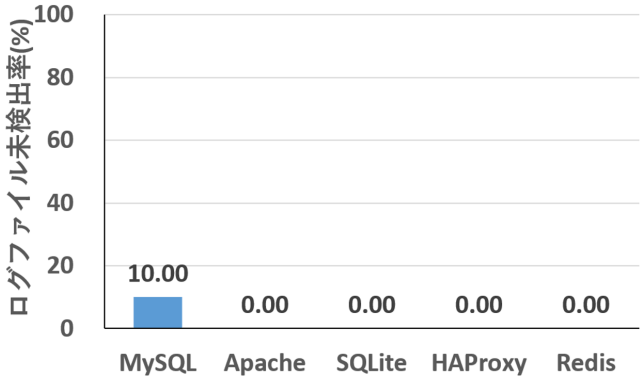

図 8　各ミドルウェアのログファイル未検出率

図 8 は各ミドルウェアのログファイル未検出率を示している．MySQL では全 10 個のログファイルのうち検出できなかったログファイルは 1 つであり，ログファイル未検出率は 10 %となった．一方，Apache では全 7 つ，SQLite では全 7 つ，HAProxy では全 6 つ，Redis では全 6 つのログファイルをすべて検出することができ，ログファイル未検出率は 0 %となった．

MySQL ではログファイル未検出率が 10 %となった．これは，検出できなかったログファイルの情報エントロピー値が 4〜5 の範囲であり，提案手法によりログファイルと判定する情報エントロピー値の範囲 (0〜2 と 5〜6) では検出できなかったためである．

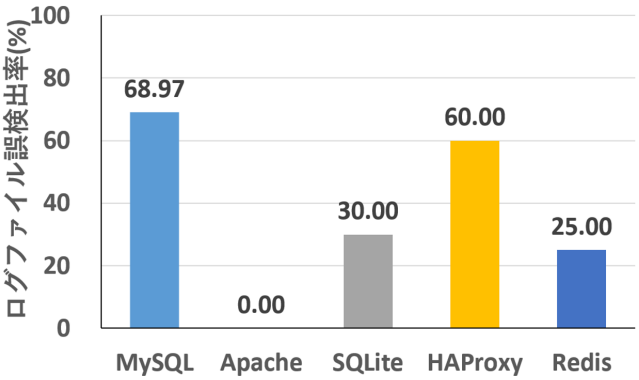

図 9　各ミドルウェアのログファイル誤検出率

図 9 は各ミドルウェアのログファイル誤検出率を示している．MySQL では提案ソフトウェアで検出したファイル 29 個のうち非ログファイルは 20 個であり，誤検出率は約 68.97 %となった．Apache では提案ソフトウェアで検出したファイル 7 個のうち非ログファイルは 7 個であり，誤検出率は 0 %となった．SQLite では提案ソフトウェアで検出したファイル 10 個のうち非ログファイルは 3 個であり，誤検出率は 30 %となった．HAProxy では提案ソフトウェアで検出したファイル 15 個のうち非ログファイルは 9 個であり，誤検出率は 60 %となった．Redis では提案ソフトウェアで検出したファイル 8 個のうち非ログファイルは 2 個であり，誤検出率は 25 %となった．

MySQL では誤検出率が約 68.97 %となった．これは，MySQL のインストールディレクトリには設定ファイルやヘルプファイル，ドキュメントが含まれており，これらのファイルの情報エントロピー値が 0〜2 のため，提案手法によってログファイルとして誤って検出されたためである．Apache では誤検出率が 0 %となった．これは，Apache の非ログファイルの提案手法で除外対象としている拡張子が，フィルタリングフェーズで除外されたためである．除外対象としている拡張子の例として.conf や.htm があげられる．

6.　議論

本稿の提案手法ではログファイル誤検出率が一番高いミドルウェア (MySQL) において約 68.97 %となった．これを改善するため，ASCII コードを提案手法に取り入れる．これは，ログファイルが人間が読める形式で記録される必要

があり，テキストとしての情報量が多いほど，ASCII コードの 32（空白文字）から 126（チルダ）の間の文字の割合が高くなるためである．これらの ASCII 文字がファイル内の何%を占めているかでログファイルであるかどうか判断することでログファイル誤検出率を削減することができる．

本稿の提案手法ではログファイル未検出率が MySQL のみ 10 %であった．これは，検出できなかったログファイルの情報エントロピー値が 4〜5 の範囲であり，提案手法によりログファイルと判定する情報エントロピー値の範囲 (0〜2 と 5〜6) では検出できなかったためである．この範囲を 0〜2 と 4〜6 に広げることで未検出のログファイルを検出することができる．この範囲を広げると，ログファイル誤検知率が増加する．したがって，ファイルパスでのフィルタリングを追加することで，ログファイル誤検知率の増加を防ぐことができる．

本稿の実験で使用した VM は Ubuntu24.04 をインストールした直後の環境であった．しかし，インストール直後の VM と一定期間使用した VM ではログファイルの特徴や量が異なる．作成してから一定期間使用している VM と作成直後の VM の差分ファイルをもちいて情報エントロピー値ごとのログファイル検出率を特定する実験によって，ログファイルの特徴や量が異なっている環境でもログファイルを特定することができる．システムの稼働時間によるログファイルの特徴の変化がなくなった期間を一定期間と定める．

7. おわりに

ログはシステムの障害対応に使用され，Elasticsearch でログの管理と検索が行われることがある．その際，Filebeat ではログの収集が行われる．Filebeat の設定には収集対象のログファイルのディレクトリを指定する必要がある．しかし，/var/log ディレクトリ以外かつ，拡張子が.log 以外のログファイルの特定には時間がかかる．そこでログファイルが VM 内のどこにあるのかを特定する手法を提案する．提案手法では，ファイルの情報エントロピーをもちいてログファイルの特定を行う．ファイルの拡張子による除外を行った後，ファイルの各バイト値の出現頻度から情報エントロピーを計算し，情報エントロピー値が提案手法の基礎実験より算出した閾値 (0〜2 と 5〜6) の範囲にあるファイルをログファイルとする．評価結果として，ログファイル検出率は MySQL では 10 個のログファイルのうち 9 つのファイルであり 90 %，Apache と SQLite では 7 つ，HAProxy と Redis では 6 つのログファイルのうちすべてのログファイルをしたので 100 %となった．ログファイル未検出率は MySQL では 10 個のログファイルのうち 1 つのファイルであり 10 %，Apache と SQLite では 7 つ，HAProxy と Redis では 6 つのログファイルのうち未検出のログファイルはなかったので 0 %となった．ログファイル誤検出率は MySQL では検出したファイル 29 個のうち非ログファイルは 20 個であり約 68.97 %，Apache では検出したファイル 7 個のうち非ログファイルは 7 個であり 0 %，SQLite では検出したファイル 10 個のうち非ログファイルは 3 個であり 30 %，HAProxy では検出したファイル 15 個のうち非ログファイルは 9 個であり 60 %，Redis では検出したファイル 8 個のうち非ログファイルは 2 個であり 25 %となった．

参考文献

[1] Yi, S., Hu, X. and Wu, H.: An automatic reassembly model and algorithm of log file fragments based on graph theory, *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 686–689 (online), DOI: 10.1109/ICSESS.2015.7339150 (2015).

[2] Liu, Z., Zhang, X., Li, G., Cui, H., Wang, J. and Xiao, B.: A Secure and Reliable Blockchain-based Audit Log System, *ICC 2024 - IEEE International Conference on Communications*, pp. 2010–2015 (online), DOI: 10.1109/ICC51166.2024.10623012 (2024).

[3] Cao, Q., Qiao, Y. and Lyu, Z.: Machine learning to detect anomalies in web log analysis, *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 519–523 (online), DOI: 10.1109/CompComm.2017.8322600 (2017).

[4] Etoh, F., Takahashi, K., Hori, Y. and Sakurai, K.: Study of Log File Dispersion Management Method, *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, pp. 371–374 (online), DOI: 10.1109/SAINT.2010.104 (2010).

[5] Nagappan, M. and Vouk, M. A.: Abstracting log lines to log event types for mining software system logs, *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 114–117 (online), DOI: 10.1109/MSR.2010.5463281 (2010).

[6] Teixeira, C., de Vasconcelos, J. B. and Pestana, G.: A knowledge management system for analysis of organisational log files, *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–4 (online), DOI: 10.23919/CISTI.2018.8399229 (2018).

[7] Nagappan, M.: Analysis of execution log files, *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 2, pp. 409–412 (online), DOI: 10.1145/1810295.1810405 (2010).

[8] Chen, P., Chao, G., Yang, L., He, H., Hong, L., Li, M., Gao, D. and Guo, S.: A Robust Log Parsing Algorithm—Practice of Logslaw in Heterogeneous Logs of Pacific Credit Card Center of Bank of Communications(PCCC), *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*, pp. 126–133（online）, DOI: 10.1109/ICIPCA59209.2023.10257676 (2023).

[9] Nguyen, H.-T., Nguyen, L.-V., Le, V.-H., Zhang, H. and Le, M.-T.: Efficient Log-based Anomaly Detection with Knowledge Distillation, *2024 IEEE International Conference on Web Services (ICWS)*, pp. 578–589 (online), DOI: 10.1109/ICWS62655.2024.00078 (2024).

[10] Fu, Q., Lou, J.-G., Wang, Y. and Li, J.: Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis, *2009 Ninth IEEE International Conference on Data Mining*, pp. 149–158 (online), DOI: 10.1109/ICDM.2009.60 (2009).

[11] Saygılı, M. c., Özelgül, S. B., Öztürk, c. S., Karaca, K. c., Gedik, A. O. and Akcayol, M. A.: Anomaly Detection on Servers Using Log Analysis, *2024 8th International Artificial*

Intelligence and Data Processing Symposium (IDAP), pp. 1–5 (online), DOI: 10.1109/IDAP64064.2024.10710799 (2024).

[12] Liu, X., Zhu, Y. and Ji, S.: Web Log Analysis in Genealogy System, *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pp. 536–543 (online), DOI: 10.1109/ICBK50248.2020.00081 (2020).

[13] Rastogi, R., Akash, S., Shobha, G., Poonam, G., Pratiba, D. and Singh, A.: Design and development of generic web based framework for log analysis, *2016 IEEE Region 10 Conference (TENCON)*, pp. 232–236 (online), DOI: 10.1109/TENCON.2016.7847996 (2016).

[14] Havens, R. W., Lunt, B. and Teng, C.-C.: Naïve Bayesian filters for log file analysis: Despam your logs, *2012 IEEE Network Operations and Management Symposium*, pp. 627–630 (online), DOI: 10.1109/NOMS.2012.6211972 (2012).

[15] Zhu, J., He, S., He, P., Liu, J. and Lyu, M. R.: Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics, *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 355–366 (online), DOI: 10.1109/ISSRE59848.2023.00071 (2023).

[16] Cardo, N. P.: Logjam: A scalable unified log file archiver, *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–9 (online), DOI: 10.1145/2063348.2063379 (2011).

[17] Son, S. J. and Kwon, Y.: Performance of ELK stack and commercial system in security log analysis, *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pp. 187–190 (online), DOI: 10.1109/MICC.2017.8311756 (2017).

[18] Prakash, T., Kakkar, M. and Patel, K.: Geo-identification of web users through logs using ELK stack, *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 606–610 (online), DOI: 10.1109/CONFLUENCE.2016.7508191 (2016).

[19] Rochim, A. F., Aziz, M. A. and Fauzi, A.: Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack, *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pp. 338–342 (online), DOI: 10.1109/ICECOS47637.2019.8984494 (2019).

[20] Langi, P. P. I., Widyawan, Najib, W. and Aji, T. B.: An evaluation of Twitter river and Logstash performances as elasticsearch inputs for social media analysis of Twitter, *2015 International Conference on Information Communication Technology and Systems (ICTS)*, pp. 181–186 (online), DOI: 10.1109/ICTS.2015.7379895 (2015).

[21] Wei, B., Dai, J., Deng, L. and Huang, H.: An Optimization Method for Elasticsearch Index Shard Number, *2020 16th International Conference on Computational Intelligence and Security (CIS)*, pp. 191–195 (online), DOI: 10.1109/CIS52066.2020.00048 (2020).

[22] Bajer, M.: Building an IoT Data Hub with Elasticsearch, Logstash and Kibana, *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 63–68 (online), DOI: 10.1109/FiCloudW.2017.101 (2017).

[23] Ahmed, F., Jahangir, U., Rahim, H., Ali, K. and Agha, D.-e.-S.: Centralized Log Management Using Elasticsearch, Logstash and Kibana, *2020 International Conference on Information Science and Communication Technology (ICISCT)*, pp. 1–7 (online), DOI: 10.1109/ICISCT49550.2020.9080053 (2020).

[24] Bhatnagar, D., SubaLakshmi, R. J. and Vanmathi, C.: Twitter Sentiment Analysis Using Elasticsearch, LOGSTASH And KIBANA, *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1–5 (online), DOI: 10.1109/ic-ETITE47903.2020.351 (2020).

[25] Divya, P. J., George, R. S., Madhusudhan, G. and Padmasree, S.: Organization-wide IOC Monitoring and Security Compliance in Endpoints using Open Source Tools, *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, pp. 1–6 (online), DOI: 10.1109/GCAT55367.2022.9971999 (2022).

[26] Lertwuthikarn, T., Barroso, V. C. and Akkarajitsakul, K.: Resource Optimization for Log Shipper and Preprocessing Pipeline in a Large-Scale Logging System, *2022 IEEE 5th International Conference on Knowledge Innovation and Invention (ICKII)*, pp. 196–200 (online), DOI: 10.1109/ICKII55100.2022.9983590 (2022).

[27] Yu, J., Chen, R., Xu, L. and Wang, D.: Concept extraction for structured text using entropy weight method, *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6 (online), DOI: 10.1109/ISCC47284.2019.8969759 (2019).

[28] Stepanenko, D. V., Stoychin, K. L. and Shevchenko, D. V.: Analysis of Operating System Event Logs when Investigating Information Security Incidents, *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 313–315 (online), DOI: 10.1109/USBEREIT58508.2023.10158875 (2023).

[29] Zhang, J. and Wang, F.: Research on the construction of log parsing system based on regular expression, *2022 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, pp. 627–630 (online), DOI: 10.1109/ICITBS55627.2022.00137 (2022).

[30] Hržić, F., Jansky, V., Sušanj, D., Gulan, G., Kožar, I. and Jeričević, D. c.: Information entropy measures and clustering improve edge detection in medical X-ray images, *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 0164–0166 (online), DOI: 10.23919/MIPRO.2018.8400032 (2018).