1W-02

ブロック単位でのディスクへの書き込み履歴から異常検出する手法の評価

吉田　光輝†　　平野　学†　　池田征士郎‡　　原田滉史‡

豊田工業高等専門学校　専攻科　情報科学専攻†　　豊田工業高等専門学校　情報工学科‡

1. はじめに

近年、コンピュータやインターネットの普及にともない不正侵入などの技術が進歩し、デジタルデータに対する犯罪が増加している。さらに、コンピュータ犯罪に関する訴訟も増加している。このような情報セキュリティを脅かす事件や事故をセキュリティインシデントという。ウイルスやトロイの木馬等を用いた攻撃や個人情報の流出に対する懸念からインシデント対応の重要性が増している。米国の国立標準技術研究所によるコンピュータセキュリティインシデント対応ガイド[1]ではインシデント対応プロセスを図 1 のように定義している。本研究では特に検知と分析を支援するシステムを提案する。

平野らの研究[2]では、仮想計算機モニタを用いてブロックストレージに対する時系列の変更履歴をブロック単位で記録してゆき、インシデント発生時に過去のディスクを復元できる監視システムを開発した。このシステムは仮想マシンで動いているゲスト OS がブロックストレージへ書き込んだブロック毎のデータを時刻情報とともに保存してゆくため、ファイルシステムの種類に関わらず監視できる利点がある。

著者らはこれまでの研究で先行システム[2]から得られた時系列のブロックデータから時刻範囲、キーワード、ファイルのブロックハッシュで検索を行い、結果をテキスト形式で出力するシステムを開発している。これにより、インシデントの証拠となるデータが書き込まれた時刻、データに含まれるキーワード、事件に関するファイルが分かっている場合には、その書き込み時刻を特定できるようになった。しかし、先行システムではインシデントの原因と考えられる異常な値を大量のデータから効率良く発見するのが困難である問題があった。そこで、本研究では過去の監視記録である時系列データをグラフで可視化し、そのグラフから GUI を用いて解析するためのシステムを開発する。

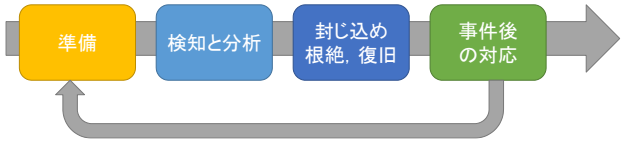

図 1　インシデント対応プロセス

2. 提案システム

提案システムの構成を図 2 に示す。提案システムではウェブブラウザから時刻範囲、キーワード、ファイルを指定して検索を行うと、Node.js のプログラムが動作する。Node.js のプログラムは指定した時刻範囲、キーワード、ファイルに対応する Hadoop の MapReduce プログラムを動作させ、先行研究のシステム[1]から検索を行い、該当する時刻、ブロック番号、データサイズを取得する。監視データはバイナリデータであるため Hadoop の SequenceFile 形式に変換しておく。また、ハッシュ検索を行う際に用いるインデックスも作成しておく。提案システムでのハッシュ検索とは、指定したファイルの 512 バイトのブロックごとにハッシュ値を計算し、検索するものである。検索結果は Hadoop 分散ファイルシステム（HDFS）からマスターサーバにコピーし、最後にグラフ描画ライブラリの D3.js を用いてウェブブラウザに表示する。

3. 評価方法

本研究では検索対象となる監視データのサイズを変化させて検索にかかる時間を計測し、開発したシステムがセキュリティインシデントの解析に有用であるかを検証した。実験で用いた検索対象データサイズは 3.3GB である。監視データは 1 分おきに Govdocs の一部のデータ（530MB）、画像データ（220KB）、テキストデータ（58 バイト）を書き込み、それを 6 回繰り返したものである。Govdocs とは米国政府サイトから集めたフォレンジック研究向けのデータセットである。Hadoop 分散ファイルシステムと MapReduce の実行には表 1 の環境を用いた。

Evaluation of an Anomaly Detection Method for Preserved Blocks on Data Storage

† Koki Yoshida, Manabu Hirano, Computer Science Course, Advanced Engineering Course for Bachelor's Degree, National Institute of Technology, Toyota College

‡ Seishiro Ikeda, Koshi Harada, Department of Information and Computer Engineering, National Institute of Technology, Toyota College

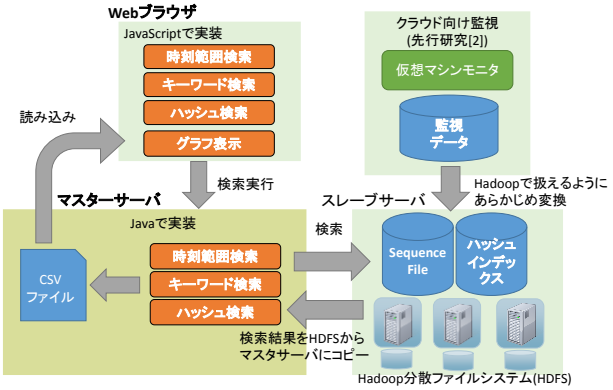

図 2　提案システムの構成図

表 1　実行環境

サーバ数	6台
OS	Linux version 2.6.32-358.el6.i686
CPU	Intel Core 2 Duo（マスタサーバ） Intel(R)Pentium(R) M processor 1.73GB（スレーブサーバ）
メモリ	4GB（マスタサーバ） 2GB（スレーブサーバ）
ハードディスク	256GB
利用プログラム	Hadoop-2.7.1,Node.js v4.4.5,D3.js v3

4. 結果

実装したシステムの検索画面を図 3 に示す。図 4 には、キーワード検索から得られた 2 つグラフを示す。いずれも横軸は時刻である。単体の検索、AND 検索を 7 通り実行した。検索はそれぞれ 3 回ずつ行い、かかった時間の平均を表 2 に示した。時刻範囲検索とキーワード検索を行うための SequenceFile の作成には 118 秒かかり、サイズは 2.03GB となった。ハッシュ検索を行う際に用いるインデックスの作成には 125 秒かかり、サイズは 239.28MB となった。

図 4 のグラフには時刻毎に書き込まれたデータ量が示されている。棒グラフの右から 2 本目が赤く表示されているのは、棒グラフごとの書き込まれたデータ量の平均値から分散の 3 倍以上離れているためである。大量のデータが書き込まれた時刻を視覚的に確認することができた。表 2 を見ると、ハッシュ検索だけ実行時間が短いことが分かる。これはあらかじめインデックスを作成しているためである。時刻範囲、キーワード、時刻範囲とキーワードの AND 検索の実行時間がほとんど同じなのは 1 つの MapReduce プログラムで同時に処理しているためである。同様にキーワードとハッシュ、時刻範囲とハッシュ、時刻範囲とキーワードとハッシュの AND 検索の実行時間がほとんど同じなのも同じ理由からである。

5. 考察

実際のインシデント対応のことを考えると、検索対象のデータはもっと大きくなる。例えば

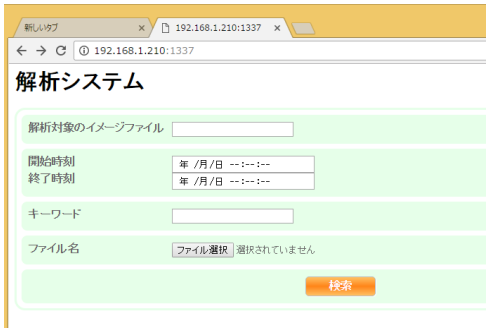

図 3　検索画面

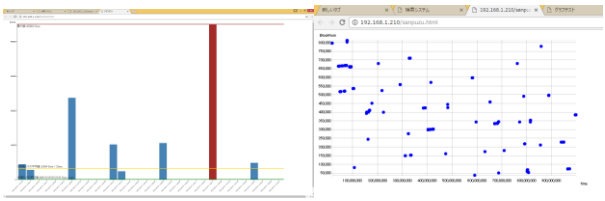

図 4　データ量の棒グラフとブロック番号の散布図

表 2　検索ごとの実行時間

検索名	実行時間[s]
時刻範囲	47.2
キーワード	49.2
ハッシュ	28.9
時刻範囲とキーワードのAND検索	41.4
キーワードとハッシュのAND検索	72.2
時刻範囲とハッシュのAND検索	68.2
時刻範囲とキーワードとハッシュのAND検索	71.0

ハードディスクのサイズが 500GB だとすると、今回の実験データの約 150 倍のサイズである。実行時間は約 3 時間になると考えられる。セキュリティインシデント対応は被害を最小限に抑えるために迅速な対応が求められるため、解析に時間がかかりすぎている。現在のアルゴリズムは逐次検索なので、現実的なデータサイズに対応するには、今後 B-tree のような効率的なアルゴリズムの採用が必要である。

6. おわりに

本研究では、先行研究のシステム[2]によって得られた監視データを可視化し、解析するシステムを開発した。その結果、定常状態からかけ離れたディスクへの書き込みを自動的に検出し、提示することができた。

参考文献

[1] K. Scarfone, K. Kent, and B. Kim: 国立標準技術研究所コンピュータセキュリティインシデント対応ガイド, NIST SP800-61, 2008.

[2] M. Hirano and H. Ogawa: A Log-structured Block Preservation and Restoration System for Proactive Forensic Data Collection in the Cloud, Proc. of the 11th International Conference on Availability, Reliability and Security (ARES 2016), pp. 355-364, 2016.