1W-03

4KiBブロックごとの類似ハッシュの検出性能の評価

都築夏樹† 平野学†
豊田工業高等専門学校 専攻科 情報科学専攻†

1. はじめに

現在、情報漏洩や不正アクセスなどのコンピュータ犯罪が増加しており、これらのセキュリティインシデントへの対応が重要になっている。セキュリティインシデント対応のディスク解析ではハッシュ値が用いられている[1]。あらかじめ検索対象のファイルのハッシュ値を計算してディスク内のファイルのハッシュ値と比較することにより効率的に検索を行う手法である。しかし、ハッシュ値を用いると数ビットしか違わない似て非なる内容を持つファイルを検索できないという問題がある。例えば一部のテキストが削除または挿入されたファイルは、変更が少量であっても検出できない。この問題を解決するために類似ハッシュアルゴリズムが提案されている。類似ハッシュアルゴリズムは、ファイルの中の特徴的なビットパターンを用いてハッシュ値を生成することで類似したファイルを検出できるようになっている。本稿では、類似ハッシュアルゴリズムの一つである sdhash [2]をディスクイメージの 4KiB ブロックごとに適用した比較結果を示し、セキュリティインシデント対応に用いる方法を検討する。

2. 類似ハッシュアルゴリズム[2]

類似ハッシュアルゴリズムはビットパターンが似ているファイルを検索する。類似ハッシュアルゴリズムによって生成されたハッシュ値は、生成元の特徴を保持している。よって、二つのファイルからハッシュ値を生成し、比較を行うとファイルの類似度を調べることができる。本研究では、類似ハッシュアルゴリズムのとして sdhash を用いる。そして sdhash を用いてハッシュ値を生成し、比較を行うと 0 から 100 までのスコアを返す。スコアは２つのファイルのビットパターンの似ている度合いを表しており 100 が最大値となる。

sdhash のハッシュ値の生成方法は、以下の通りである。まず、先頭から 1 バイトずつシフトしながら特徴的な 64 バイトを探索する。そして、特徴的なビットパターンの 64 バイトを見つけると SHA1 ハッシュを生成する。生成されたハッシュ値は、ブルームフィルタに格納される。ブルームフィルタにはハッシュ値が存在しているかが 0 と 1 で格納されており、限界まで格納された場合には、新たにブルームフィルタを作成する。特徴的なビットパターンの探索を先頭から最後まで行い、得られたブルームフィルタをすべて連結してハッシュ値とする。ハッシュ値を比較する場合は、ブルームフィルタをそれぞれ比較して得られた最小ハミング距離の平均をスコアとする。

3. 実験方法

ディスクイメージに対して、類似ハッシュアルゴリズムである sdhash を適用した。利用したディスクイメージは、Windows 8.1、MacOS X 10.9、CentOS 6.5（全て初期インストール状態のもの）の 3 種類である。まず、ディスクイメージを 4KiB のブロックに分割する。そして、各ブロックに sdhash を適用する。このときブロックの中身がすべて ”0” のブロックを除外する。1%の抽出はかたよりのなくなるように先頭から 100 ブロックごとに 1 ブロックを抽出した。そして、全ブロックの 1%を抽出してハッシュ値を生成する。さらに生成されたハッシュ値を総当たりしてディスクイメージのすべてのブロックの比較結果を得る。上記の手順を 3 つのディスクイメージに対して行い、得られた結果を類似度ごとに分類した。

4. システムの実装

3 節で示した実験を行うシステムを並列分散処理フレームワークの Hadoop を用いて実装した。実験に用いた Hadoop クラスタの構成を表 1 に示す。用いた sdhash のバージョンは 4.0 である。C 言語で実装されている sdhash を Hadoop で実行するために Hadoop Streaming を用いた。Hadoop のバージョンは 2.7.1 である。ディスクイメージから sdhash で類似ハッシュアルゴリズムのハッシュ値を計算させる処理を MapReduce で開発した。

Evaluating Detection Performance of Similarity Hashes in 4KiB Blocks
†Natsuki Tsuzuki, Manabu Hirano, Computer Science Course, Advanced Engineering Course for Bachelor's Degree, National Institute of Technology, Toyota College

表1　Hadoopクラスタの構成

	マスターサーバ	スレーブサーバ
CPU (Core)	XeonE5-2630v3 x2(16 core)	Core i7 5820K (6 core)
CPU キャッシュ	20MiB	15MiB
RAM	DDR4 64MiB	DDR4 64MiB
NIC	10GBASE-T Intel 540-T2	10GBASE-T Intel 540-T2
ストレージ	SATA3 SSD 512GB	SATA SSD 512GB
マシン数	1	3

ブロックごとに総当たりで類似度のスコアを算出する処理は Hadoop の分散処理ではなく、マスターサーバで C 言語を用いて実装した。

5. 実験結果

各ディスクを 4KiB に分割し、すべてが ”0” のブロックを除外した結果、Windows 8.1 のブロック数は 3,139,531 個、CentOS 6.5 のブロック数は 3,311,809 個、MacOS X 10.9 のブロック数は 2,499,362 個となった。その後、全ブロックの 1%を抽出すると Windows 8.1 のブロック数は 31,395 個、CentOS 6.5 のブロック数は 33,118 個、MacOS X 10.9 のブロック数は 24,993 個となった。次に各ブロックに sdhash を適用しハッシュ値の比較を総当たりで行った。得られた類似度をスコアごとにまとめた結果を表 2 に示す。

6. 考察

表 2 の結果からディスクイメージの 4KiB ブロックへ類似ハッシュアルゴリズムを適用した際の有効性を考察する。類似ハッシュアルゴリズムは、ビットパターンが似ているファイルを検出するアルゴリズムである。そして、比較結果のスコアが高いほど類似したファイルといえる。今回の実験ではファイル単位ではなく 4KiB ブロックに対して sdhash を適用した。類似ハッシュアルゴリズムをディスクイメージに適用するにあたって、ディスクイメージの中にスコアが高いブロックが多く存在すると検索結果に False Positive（偽陽性）が多くなるという問題がある。表 2 より Windows8.1 では 0 から 19 のスコアの値が合計の 99.997%を占めている。CentOS 6.5 や MacOS X 10.9 の場合も同様で CentOS 6.5 では全体の 99.998%、MacOS X 10.9 では 99.997%と大部分を占めている。よって、sdhash の False Positive（偽陽性）の割合は低いと考えられる。

表2　ハッシュ値を総当たりしたスコアの分布（ペアの個数）

スコア	Windows 8.1	CentOS 6.5	MacOS X 10.9
100	167	142	82
90-99	127	182	841
80-89	420	490	723
70-79	533	559	588
60-69	1,005	721	847
50-59	1,515	997	639
40-49	1,986	1,349	632
30-39	2,122	1,475	856
20-29	6,100	3,015	1,296
0-19	492,793,340	548,375,428	312,306,024
合計	492,807,315	548,384,403	312,312,528

表 2 のスコアが 20 から 100 の範囲を見ると、MacOS X 10.9 を除く 2 つの OS でスコアが最大値の 100 に近づくにつれて、検出されたハッシュ値の比較結果のペアの個数が少なくなっていることがわかった。スコアが高いファイルは類似度が高いため検出するスコアの下限を上げることで誤検出が減少し、False Positive（偽陽性）が減少すると考えられる。今後は適切な閾値を検討していく必要がある。

7. まとめ

本稿では類似ハッシュアルゴリズムの sdhash を代表的な 3 種類の OS のディスクイメージに適用した。そしてハッシュ値の比較を総当たりで実行した結果、すべての OS でスコアが 0 から 19 の小さい領域に 99.99%のペアが分布していることがわかった。そして検出するスコアの下限を設定することで検出精度を調整できることがわかった。本稿の実験により、類似ハッシュアルゴリズムがフォレンジックに対してある程度有効であることがわかったが、今後さらなる調査が必要である。

参考文献

[1] K. Scarfone, K.Kent, and B. Kim: 米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, SP800-61, 2008.

[2] Vassil Roussev: Data fingerprinting with similarity digests, IFIP International Conference on Digital Forensics, Springer Berlin Heidelberg, pp. 207-226, 2010.