ログ調査のための LCS を用いたログイベント圧縮表現方法の提案

谷屋 直樹[1,*] 中野 心太[2] 関谷 信吾[2] 折田 彰[3]
岸本 頼紀[4] 早稲田 篤志[4] 花田 真樹[4]

概要：ログ調査ではイベントが膨大になるため視認性が悪くなる問題がある.そこで,LCS を用いたログ圧縮表現を提案する.ログ調査では異常ログの箇所を強調することが求められるが,LCS では繰り返し出現するパターンを圧縮するため,異常ログのような出現が稀な箇所は圧縮されずに残ると考えられる.本論文では,LCS を用いたログ圧縮の手法として,イベントを時間軸で２分割したものを対象とした LCS によるログ圧縮手法について提案し,本手法の適用によるログ調査の支援可能性について論じる.

キーワード：デジタルフォレンジック，ログ解析, LCS

A proposal of Log Event Compression Representation Method Using LCS for Log Analysis

Naoki Taniya[1,*] Shinta Nakano[2] Shingo Sekiya[2] Akira Orita[3]
Yorinori Kishimoto[4] Atsushi Waseda[4] Masaki Hanada[4]

Abstract: Log analysis has the problem of poor visibility due to the large number of events.Therefore, we propose a log compression representation using LCS.Log analysis requires emphasizing abnormal log sections.Since LCS compresses repeated patterns, it is thought that sections that rarely appear, such as abnormal logs, will remain uncompressed.In this paper, we propose a log compression method using LCS that targets events divided into two on the time axis, and discuss the possibility of applying this method to support log investigation.

Keywords: Digital Forensics, Log Analysis, LCS

1. はじめに

サイバー攻撃の調査において,ログは攻撃の痕跡を把握する上で重要な情報源の 1 つである.しかし,実際の運用環境では膨大なログが生成されるため,調査者がその全てを網羅的に調査・分析することは時間手的・労力的コストを伴う.特に,複数端末にわたるラテラルムーブメント型攻撃やランサムウェア等といった短時間で多くの危機に影響を及ぼす攻撃に対して調査する場合,ログは端末数の増加に比例し,可視化結果は膨大となる.これにより,攻撃痕跡の調査における,ログの視認性が低下することが課題となる.これに対して,ログを端末ごとに横並びに配置し,時系列を揃えることで,攻撃の動向を追跡しやすくする可視化手法が提案されている[1].この手法は,攻撃の全体像を見やすくする効果があるものの,１端末あたりのログの件数が多い場合,可視化の結果が膨大になってしまい,攻撃の痕跡が定常的なログに埋もれてしまう可能性がある.したがって,これに対処するには,ログを並べるだけでなく,ログ件数を整理または圧縮し,視認性を向上する必要がある.そのためには,攻撃痕跡等の重要な情報を残しつつ,ログを圧縮する必要がある.

本論文では,この課題に対して,最長共通部分文字列（Longest Common Substring, LCS）[2]を用いたログ圧縮手法を提案する.本手法では,記録されるログの中には,繰り返し出現する定常的な記録が残る点に着目し,ログを前半後半で出現する共通部分を圧縮することで,攻撃に関連する異常な記録が顕在化されつつ圧縮ができるのではないかと考える.これについて検証し,圧縮率と圧縮した状態で攻撃痕跡が残存するかを評価し,攻撃痕跡が含まれるログの圧縮への効果について論じる.

2. 先行研究

先ラテラルムーブメン攻撃等に対応する痕跡の可視化において,端末ごとにログを並べて可視化することで,攻撃者の動向を追いやすくする手法が提案されている[1].この手法

1 東京情報大学大学院 総合情報学研究科
Graduate School of Informatics, Tokyo University of Information Sciences.
2 株式会社日立システムズ サイバーセキュリティリサーチセンタ
Hitachi Systems, Ltd.Cyber Security Research Center.
3 日立システムズ セキュリティ・コアバリュー本部
Hitachi Systems, Ltd.Security Core Value Office
4 東京情報大学
Faculty of Informatics, Tokyo University of Information Sciences
* 2008tn@edu.tuis.ac.jp

では，異なる形式で記録されたログのフォーマットを統一し，HTML として出力することで，利用者が一つの画面上で多様なログを参照可能にしている.この仕組みにより，調査者は端末横断的な攻撃痕跡を直感的に把握しやすくなる.

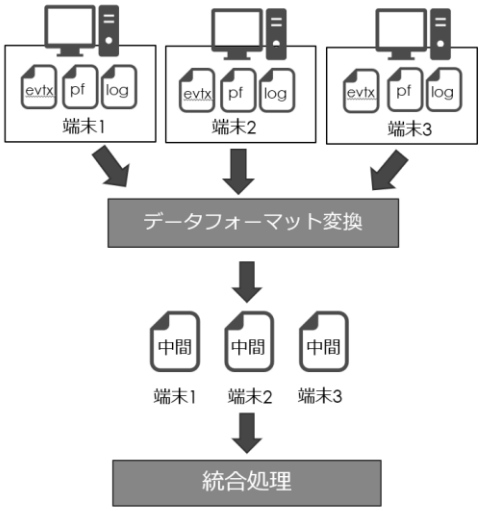

図 1.可視化手法の構成図

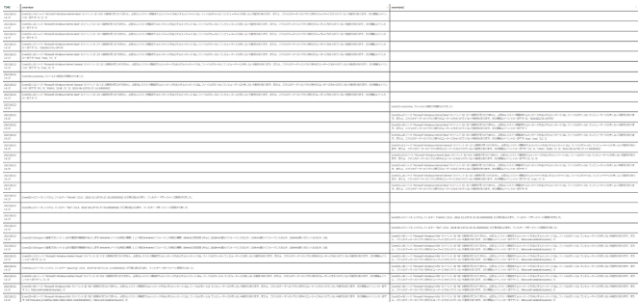

図 2.可視化システムの実行結果

TIME	overview	overview2
2023/8/22 16:34 23	FileSize=45538 FileName=RAT_CLIENT.EXE	
2023/8/22 16:35 42	FileSize=10672 FileName=CMD.EXE	
2023/8/22 16:36 44	FileSize=29340 FileName=PSEXEC.EXE	
2023/8/22 16:39 54		192.168.10.5"GET /webshell2.php?cmd=ipconfig HTTP/1.1" 200 360
2023/8/22 16:39 54		EventID=1000障害が発生しているアプリケーション名: httpd.exe、バージョン: 2.4.41.0障害が発生しているモジュール名: php7ts.dll、バージョン: 7.4.0.0
2023/8/22 16:40 00		FileSize=28200 FileName=CONHOST.EXE
2023/8/22 16:40 02		FileSize=9560 FileName=CMD.EXE
2023/8/22 16:40 12		FileSize=8932 FileName=IPCONFIG.EXE

図 3.可視化システムの実行結果の拡大図

しかしこの手法では,可視化のためのフォーマット統一に重点が置かれており，ログ量そのものを圧縮して視認性を高める点については十分に検討されていない.そのため,情報が過剰になることで重要な痕跡が埋没し,重要な情報を抽出することが困難になる可能性がある.

そこで本研究では,岩崎らのシステムの可視化的アプローチを踏まえつつ，LCS を用いた端末単位のログの縦軸方向に圧縮する手法を提案する.LCS を用いることで定常的に繰り返されるログをまとめ,異常ログは圧縮せずに残すことで,攻撃痕跡を強調し,膨大なログを対象とする調査において視認性を高めることを目指す.

3. 考え方

先行研究における可視化手法は,複数の端末を横並びに配置し,時系列に沿って攻撃者の行動を追跡できる点に特徴がある.しかし,端末ごとのログは膨大であるため,可視化結果も大量になり,視認性の低下を招く可能性がある.そこで本研究では,1 端末ごとのログを圧縮し,可視化における情報の見やすさを高めることを試みる.ただし,単純に圧縮してしまうと,攻撃の痕跡となるログまでまとめて消失してしまい,フォレンジックの観点から分析が困難になる.したがって,定常的に記録されるログは圧縮しつつ,攻撃に関連する異常なログは残す手法が必要となる.この点に着目し,本論文では,定常ログが一定の並びとして繰り返し出現する性質を利用し,最長共通部分文字列（Longest Common Substring, LCS）を用いる.LCS により共通部分が圧縮される一方,定常とは異なる記録は圧縮されずに残るため,異常痕跡が顕在化することが期待される.ただし,LCS による共通部分抽出には 2 系列の比較が必要となるため,対象データとして Windows イベントログを半分に分割し,ログの前半後半の 2 系列を用いて比較を行う.

本研究では,この手法を用いてログの圧縮率と,攻撃が実行された時間帯に対応するログが圧縮されずに残るかを調査し,サイバー攻撃を受けた端末のログ圧縮に対する有効性を検証する.

4. 実験

4.1 概要

本実験では,Windows のイベントログ（Application，Security，System）を対象とし，仮想環境下で作成した攻撃ログを埋め込むことで，攻撃痕跡を含む検証用データを準備した.これに対し,最長共通部分文字列（Longest Common Substring, LCS）を用いて,ログを圧縮.LCS で共通するイベント ID の並びを抽出し,それらをまとめてトークンで置換することで圧縮を行った.ここで, LCS は 2 系列間の比較を前提とするため，Windows イベントログを前半と後半に分割して 2 系列を生成し，これらを比較対象として処理した.本実験では，圧縮処理によるログの圧縮率を評価するとともに，攻撃が行われた時間帯の痕跡ログが圧縮されずに残存するかを検証することで，本手法が痕跡顕在化に有効であるかを確認する.

4.2 実験環境と攻撃シナリオ

実験の実施にあたり構築した環境を以下に示す.

4.2.1 ログ圧縮処理環境

ログ圧縮に用いた実験環境を以下に示す.

環　　境：Windows11, Jupyter
言　　語：Python3.11.8
ライブラリ：pandas, difflib-SequenceMatcher
アルゴリズム：Longest Common Substring

4.2.2 攻撃痕跡作成環境

攻撃痕跡は，VirtualBox 7.1.12 上に構築した Active Directory 環境において作成した.攻撃端末には Kali Linux を使用し，被害環境には複数の Windows 端末を配置した.攻撃手法としては，代表的な認証情報窃取攻撃である Pass-The-Hash (PTH)と Pass-The-Ticket (PTT)[3] を選択した.

・Pass-The-Hash

環　　境：Virtual Box7.1.12
攻 撃 端 末：Kali Linux
被 害 環 境：AD 環境
被害端末 1：Windows10
- ログイン情報：ユーザ A, ユーザ B

被害端末 2：Windows10
- ログイン情報：ユーザ B, Administrator

被害端末 3：Windows2019
- ログイン情報：Administrator

・Pass-The-Ticket

環　　境：Virtual Box7.1.12
攻 撃 端 末：Kali Linux
被 害 環 境：AD 環境
被害端末 1：Windows10
被害端末 2：Windows2019

4.2.3 攻撃シナリオの詳細手順

各攻撃シナリオにおける攻撃手順を以下に示す.

・Pass-The-Hash

1. nmap でポートスキャン,ターゲットを特定.
2. nmblookup でターゲットの NetBIOS 名やドメイン情報を確認.
3. Metasploit で psexec を使用し,ターゲット(被害端末 1)にユーザ A で接続.
4. 被害端末 1 の PowerShell を起動,Kali linux から被害端末 1 に mimikatz.exe と PxEcxec64.exe を配布.
5. 被害端末 1 で mimikatz.exe を実行,ユーザネーム,NTML 等情報を取得.
6. Pass-the-Hash 攻撃.
7. 取得したハッシュを使い,ユーザ B で被害端末 1 に再侵入.
8. PsExec.exe を実行し,ユーザ B で被害端末 2 に mimikatz.exe を配布,実行（失敗

・Pass-The-Ticket

1. WindowsServer2019 で C 直下にテキストファイル test.txt を作成
2. 被害端末Aにてリアルタイム保護を切断
3. 被害端末Aにて bayload.exe, mimikatz.exe ダウンロード,保存
4. meterpreter と被害端末Aのセッション開始
5. 被害端末Aの SYSTEM 権限取得
6. 被害端末Aの powershell 遠隔起動,mimikatz 起動
7. 被害端末Aのに保存されている ticket 情報確認
8. PassTheTicket 攻撃

4.2.4 実験対象データ

Windows イベントログ(Application, Security, System)の日付時刻と ID のカラムを対象とする.

・PTH 攻撃を含むログ件数

Application ログ：24768（うち攻撃ログ 1 件
Security ログ：22995（うち攻撃ログ 30 件
System ログ：22447（うち攻撃ログ 3 件

・PTT 攻撃を含むログ件数

Application ログ：17274（うち攻撃ログ 14 件
Security ログ：22799（うち攻撃ログ 445 件
System ログ：24369（うち攻撃ログ 12 件

4.3 ログ圧縮と評価手順

ログ圧縮と評価手順を以下に示す.

1.Windows イベントログ（Application, Security, System）を取得.
2.攻撃ログを作成.手順 1 で取得したログに埋め込む.
3.ログのカラムを「日付時刻」「EventID」のみに整理.
4.ログを前半・後半に分割し,LCS の対象とする.
5.LCS を用いて共通部分を抽出し,[COMMON_x] 形式のトークンに置換（圧縮）.
6.最長共通部分文字列が 2 件になるまで圧縮する.
7.圧縮後ログについて以下を評価する.
- i 圧縮成功件数,圧縮失敗件数,圧縮率
- ii 置換（圧縮）回数
- iii 攻撃ログが残存した件数と総数

4.4 実験結果

4.4.1 攻撃シナリオ：Pass-The-Hash

表 1 に攻撃シナリオ: Pass-The-Hash のログ圧縮結果を示す.Application ログでは 75.03% の圧縮率を示したが，攻撃痕跡は残存しなかった.Security ログでは 63.08% の圧縮率を示し，痕跡は 30/30 件すべて残存した.System ログでは 84.77% の高圧縮率を示しつつ，痕跡もすべて残存した.

表.1　攻撃シナリオ:Pass-The-Hash

ログ種類	圧縮成功	圧縮失敗	圧縮率	圧縮回数	残存攻撃ログ	残存率
Application	18583	6185	75.03%	830	0/1	0%
Security	14506	8489	63.08%	123	30/30	100%
System	19028	3419	84.77%	170	3/3	100%

4.4.2 攻撃シナリオ：Pass-The-Ticket

表 2 に攻撃シナリオ: Pass-The-Hash のログ圧縮結果を示す.Application ログでは 67.05% の圧縮率を示しつつ，痕跡の残存率は 7.14% であった.Security ログでは 90.71% の圧縮率を示したが，残存率は 1.80% であった.一方で System ログでは 69.93% の圧縮率を示し，痕跡は 100% 残存した.

表.2　攻撃シナリオ:Pass-The-Ticket

ログ種類	圧縮成功	圧縮失敗	圧縮率	圧縮回数	残存攻撃ログ	残存率
Application	11582	5692	67.05%	746	1/14	7.14%
Security	20680	2119	90.71%	288	8/445	1.80%
System	17042	7327	69.93%	247	12/12	100%

5. 考察

本実験では，圧縮率を維持しながら攻撃痕跡を残存させることができた例が確認された.しかし一方で，攻撃ログが圧縮されて消失する事例も見られた.攻撃痕跡が圧縮されてしまう要因の一つは，攻撃に関連するイベントが定常的な記録列に混在し，共通部分として誤って圧縮対象となるためであると考えられる.一方で，定常ログとは異なる挙動を示す攻撃イベントについては圧縮されずに残存し，痕跡が顕在化する場合も確認された.このことは，本手法が必ずしも痕跡を消失させるのではなく，特定の条件下では攻撃ログを強調できる可能性を示しており，痕跡抽出の有効な補助手段となり得ることを示唆している.

6. 結論

本研究では，サイバー攻撃調査におけるログの膨大さと視認性の低下という課題に対し，最長共通部分文字列（LCS）を用いたログ圧縮手法を提案した.実験の結果，攻撃ログが定常的な記録列に混在する場合には痕跡が圧縮されて消失する事例が見られた一方，定常ログとは異なる挙動を示す攻撃イベントについては圧縮されずに残存し，痕跡が強調される場合も確認された.これにより，本手法は必ずしも痕跡を消失させるものではなく，条件次第では攻撃痕跡を顕在化させる有効な補助手段となり得ることが示された.

今後の展望としては，異なる攻撃シナリオやより大規模なログ環境での適用検証を進めるとともに，LCS 以外の圧縮アルゴリズムとの比較や組み合わせを検討することが課題である.

参考文献

[1] 岩崎晃大, 折田彰, 関谷信吾, 中野心太, 花田真樹, 布広永示, 岸本頼紀, “機械学習を用いた痕跡情報自動収集システムの試作”, “コンピュータセキュリティシンポジウム 2023 論文集”, pp-916-918,2023-10-23.

[2] Oracle (R) Enterprise Data Quality オンライン・ヘルプ バージョン 8.1, “比較: Longest Common Substring”, https://docs.oracle.com/cd/B72202_01/Content/processor_library/matching/comparisons/longest_common_substring.htm

[3] IIJ,” Mimikatz 実行痕跡の発見手法”, https://www.iij.ad.jp/dev/tech/techweek/pdf/171108_02.pdf