侵入検知に向けたシステム内悪性活動の紐付け及び可視化システムの開発

宮坂剛[1]　大窪巳祐[2]　末次信貴[2]　稲葉緑[2]　橋本正樹[2]

概要：近年の標的型攻撃対策においては，システムに残されたログから攻撃源を特定する作業の簡易化・効率化が課題となっている．標的型攻撃は，様々なプロセスやファイル等リソースが連携することで実現されることが多いため，膨大なログから攻撃に関係するプロセスやリソースを発見して正しく連結し，一連の悪性活動として組み上げて認識することが難しく，結果的に，真の攻撃源がどこにあるのかが不明瞭となる原因になっている．一般に，この作業は，システム内の活動に対する深い知識が必要となるため，専門の技術者が手作業で対応するが，非常にコストがかかる．本研究は，IDS/IPS 等で外部との不正な通信を検知し，その攻撃源を見つけるシナリオを想定し，システムコールログを半自動的に解析した上で，関連するプロセス及びファイルを一連の悪性活動として紐付け・可視化するシステムを提案するものである．本稿は，提案システムの概要と実装を説明し，その評価として，実際に提案システムを使用した利用者に対するアンケート調査を実施した結果を報告するものである．

キーワード：システムコール，可視化，紐づけ

Linking and Visualizing Malicious Activities for Intrusion Detection

Tsuyoshi Miyasaka[†1]　Miyu Okubo[†2]　Nobuki Suetsugu[†2]
Midori Inaba[†2]　Masaki Hashimoto[†2]

Abstract: In recent years, one of the major challenges for countermeasures against targeted cyber attacks has been to simplify and streamline the process of identifying the source from the logs stored in the system. Targeted attacks are often carried out by the coordination of various processes, files and other resources. This makes it difficult to discover and correctly link the processes and resources involved in an attack from a huge amount of logs, and to assemble and recognize them as a series of malicious activities. Therefore, it often remains unclear where the real source of the attack lies. Typically, this analysis work requires in-depth knowledge of the activities of the system and is therefore handled manually by specialist engineers, which is inefficient and very costly. In this study, we simulate a scenario in which an IDS/IPS detects unauthorized communication with the outside and try to identify the source of the attack. We also propose a system that semi-automatically analyzes the logs, and links related processes and files as a series of malicious activities, and visualizes them. This paper describes the outline and implementation of our proposed system. As an evaluation of the system, this paper also reports the results of a questionnaire survey from users who actually used the proposed system.

Keywords: Malicious Activity, System call, Visualization, Linking.

1. はじめに

サイバー攻撃による被害が後を絶たない．サイバー攻撃対策として，例えば侵入検知システムを導入していた場合であっても，一般に，攻撃者の侵入を完全に防ぐことは不可能であり，機密情報が漏えいした後に初めて攻撃に気づくケースも数多く存在する．その原因は様々であるが，例えば，標的型攻撃では特定の組織に向けてカスタマイズした手段が用いられるため，侵入検知システムを容易にすり抜けてしまうことや，日々発生する多くのアラートについて，攻撃の全体像を描く情報や分析能力が不足していること等が挙げられる．すなわち，侵入検知システムには，IDS や IPS を初めとしたネットワーク型侵入検知のものや，ウィルス対策ソフトに代表されるホスト型侵入検知が挙げられるが，これらは攻撃の片鱗が見られた際の動作をシグネチャや振る舞いから捕えるものであるため，単に攻撃全体の一片を捕捉した情報となっていることが多い．

IDS/IPS により攻撃が判明した際の一般的な対応としては，該当ホストをネットワークから隔離した後にログを取得し，フォレンジック技術を用いた侵入経路や被害状況の詳細調査が行われる．攻撃者の主たる侵入経路はメールや脆弱性を悪用するものが多いが，調査には分析のための特殊なソフトウェアや，OS・アプリケーション等を熟知した技術者による分析が必要となる．こうした調査には時間と労力がかかるため，様々な状況を想定し，一部または全体のシステム停止を実施する場合もあり，被害額が莫大なものとなるケースも存在する．また，十分なセキュリティ体制を自ら構築・維持することが難しい組織は SOC 等の有人セキュリティ監視サービスを用いて対策を行うなどしているが，SOC においても高度な分析を行うための情報や人材が不足しているという現状がある．

1 (株)アイネス総合研究所
INES Research Institute, Inc.
2 情報セキュリティ大学院大学
INSTITUTE of INFORMATION SECURITY

こうした現状を踏まえ，本研究では，IDS/IPS 等で外部との不正な通信を検知し，その攻撃源を見つけるシナリオを想定し, システムコールログを半自動的に解析した上で，関連するプロセス及びファイルを紐付け・可視化するシステムを開発する．すなわち，Linux を対象に，外部との不正な通信を検知した際の，システムコールの呼び出し履歴を集約して解析し，一連の悪性活動に関連するプロセス及びファイルを半自動的に紐付け, 可視化する. これにより，IDS/IPS のアラート発生時に，これまで専門知識に基づく手動分析に頼っていた攻撃源の分析作業を半自動化して効率化し，人間の負担の軽減を目指すものである．

以降本稿では，第 2 章において，提案システムの設計について述べる．その後，第 3 章で，提案システムのプロトタイプ実装について説明し，その後第 4 章で，提案システムの効果を測定するための評価とその結果について説明する．第 5 章では，関連研究について説明し，最後に， 第 6 章では，本稿のまとめと今後の課題を述べる．

2. 提案システムの設計

本章では，提案システムの設計として，分析対象とするシステムコールと，その収集・分解・集積方法について説明する．また，集積したシステムコールログを分析し，可視化する手法についても説明する．

2.1 分析対象システムコール

提案システムでは，Linux Audit Framework[1]を用いて収集した Linux のシステムコールを分析対象として利用する. システムコールは，プロセスが Linux カーネルの機能を利用するための API であり，例えばファイルを開く際には open システムコール, プログラムの実行には execve システムコール, 別のホストと TCP/IP 通信を開始する際は connect システムコールが利用される．システムコールの数や種類は Linux カーネルのバージョンによって異なるが，本研究では，多くのシステム内活動に共通してよく利用されるシステムコールとして，open, execve, connect, unlink, close を選出し，分析対象とする．これらの他にも，ファイルの読み書きを行う read, write をはじめとし，多数のシステムコールが存在するが, 分析対象が膨大な量となるため除外し，上記 5 つのシステムコールに分析対象を絞り込んだ．特に open システムコールについては，プロセスがファイルに対して read，write を実行する前段階で open システムコールを呼び出すため，後述の紐付けアルゴリズムによってプロセス同士の関係性を見出す際に非常に有用である．また，open システムコールには，ファイルパスや inode 番号，実行ファイル，親プロセス ID 等，プロセスやファイルの関連付けに有用な情報が多く含まれていることが多い．

2.2 システムコールログの収集・分解・集積

提案システムでは，システムコールログの収集に Auditbeat[2]を, その分解に Logstash[3]を利用する. Auditbeat は, Linux ホストで Audit フレームワークにより記録されたシステムコールログを収集して, Logstash に渡す．Logstash は，受け取ったシステムコールログを分解し，統合ログ管理基盤に集積する．システムコールログの集積と，その後の利用インターフェースには，Elasticsearch[4]を用いる．Elasticsearch は，Apache lucene[5]を基盤とするオープンソースの高速な検索エンジンであり，提案システムでは，これをシステムコールログの集積・管理・分析のための基盤として利用する．一般に，セキュリティに関連するログを管理するシステムを SIEM (Security Information and Event Management) と呼ぶが, 提案システムにおける Elasticsearch は，データの検索処理を実行するための API を備えた他の SIEM システムで代用することも可能である．

2.3 システムコールログの分析と可視化

提案システムでは，統合ログ管理基盤に集積したシステムコールログを分析し, ネットワーク図として可視化する．システムコールログの分析と可視化においては，統合ログ管理基盤から情報を取得する API を備えたものを利用する．

分析プログラムを記述するプログラミング言語として Python を利用する．統合ログ管理基盤から情報を収集する API は，Elasitcsearch ライブラリを利用した．なお，提案システムにおいては，収集したシステムコールデータを後から一覧で確認できるよう Pandas ライブラリによる表データ化も行う．これにより，CSV ファイルとして出力可能のため，利用者が後の調査で確認できる．

ネットワーク図の生成には，NetworkX[6]を利用する．生成したネットワークデータを描画する基盤は複数存在するが，可視化や属性表示に優れた Cytoscape[7]を利用する．

2.4 プロセス・ファイルの紐付け手法

提案システムでは，システムコールログを分析することで，一連の悪性活動を半自動的に紐付ける．提案システムにおける紐付けは，プロセスの親子関係による紐付けと，プロセスとファイルの関係による紐付けの 2 種類がある．

2.4.1 親子関係による紐付け

親子関係による紐付けは，親プロセス方向の紐付けと子プロセス方向の紐付けからなる(図 1).

親プロセス方向の紐付けにおいては，調査対象プロセスがシステムコールを呼び出すとその親プロセス ID も記録されるため，その情報を用いて親プロセスが呼び出しているシステムコールを検索することが可能である．これを繰り返すことで親プロセス名や ID，開いているファイル，実行しているファイル，通信先等の情報を得る．調査対象プ

ロセスと親プロセスの関係は 1 対 1 となる．

子プロセス方向の紐付けにおいては，調査対象プロセスの ID を親として持つプロセスを検索することで，子プロセスが呼び出しているシステムコールを検索することが可能である．これを繰り返すことで子プロセス名等の情報を得る．調査対象プロセスと子プロセスの関係は 1 対多となる．

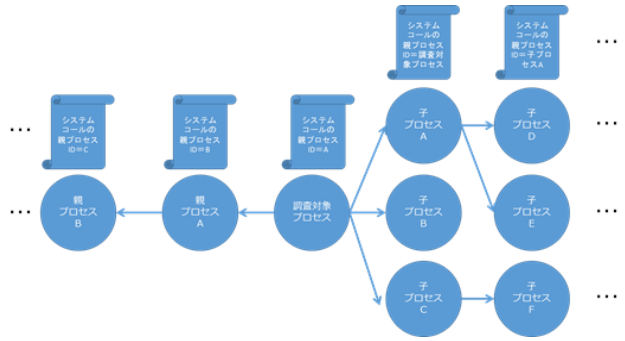

図 1.　親子関係による紐付け

2.4.2 プロセスとファイルの関係による紐付け

調査対象プロセスが関連しているファイルに対し，他プロセスとの関連を紐付ける．具体的には，調査対象プロセスが execve システムコールで実行しているファイルに対し，open システムコール (CREATE オプション) を呼び出しているプロセスを検索し紐付ける．これにより，調査対象プロセスが実行しているファイルと，そのファイルを作成したプロセスを紐付ける(図 2)．また，execve - open (create) の他にも表 1 に示す組み合わせについても，関連があるものとして紐付ける．

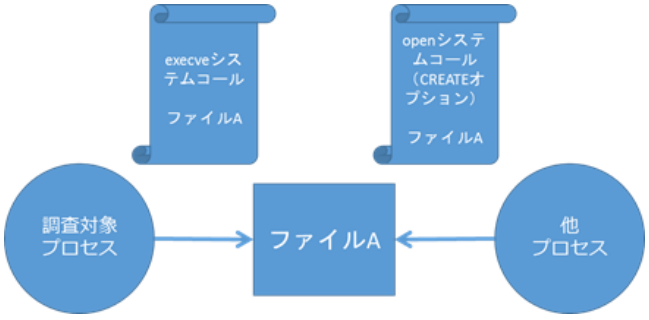

図 2. ファイルとの関係に基づく紐付け

表 1 ファイルに関する紐付けの組合せ

No	調査対象プロセス	他プロセス
1	open	open(オプションなし)
2	open	open(CREATE オプション)
3	execve	open(オプションなし)
4	execve	open(CREATE オプション)

3.　プロトタイプ実装

本章では，IDS/IPS 等で外部との不正な通信を検知し，その攻撃源を見つけるシナリオを想定した提案システムのプロトタイプ実装について，その概要と動作手順を説明する．

3.1　プロトタイプシステムの概要

プロトタイプシステムは，第 2 章で述べた設計により，システム内活動のシステムコールログ収集・分析と，紐付けによる悪性活動の可視化を半自動的に行うものである．

プロトタイプシステムでは，ネットワーク図を用いた視覚的なデータを生成するので，分析者は，アラートの対象となるプロセスを中心に，関係するプロセスやファイル，通信等を視認し，攻撃の根元となったプロセスや悪性ファイルを特定することができる．

プロトタイプシステムは以下の情報を基に攻撃活動の分析を行う．

① 分析対象データ
分析対象データとして，Elasticsearch がログを保管する単位であるインデックスを指定する．インデックスは通常ホスト単位で作成されるもので，プロトタイプシステムにおいては，システムコールログを保持するデータベースである．
② 検索時間範囲
攻撃活動検知時，システムコールログを検索する時間範囲を指定する．プロトタイプシステムにおいては，攻撃アラート検知より 20 分前までのシステムコールログを対象とする．
③ IP アドレス
IDS/IPS 等により検知された不正な通信の通信先を想定し，特定の IP アドレスを指定する．プロトタイプシステムでは，この IP アドレスを起点に connect システムコールを検索し，プロセス ID や関連ファイルの検索を行う．

3.2　プロトタイプシステムの動作手順

プロトタイプシステムは以下の手順により，IDS/IPS 等によるアラートの検知から，可視化，分析者への通知を半自動的に行う．

① 侵入検知システムによるアラート検知
IDS/IPS 等によって外部との不正な通信が検知され,アラートが発生する．
② 悪性活動の分析開始
アラートの発生をトリガーとし，悪性活動の調査が開始される．
③ 不正通信元プロセスの特定

アラートに記載された送信元・送信先 IP アドレス情報を利用して connect システムコールを分析し，不正通信元のプロセスを特定する．
④プロセスとファイルの紐付け
不正通信元プロセスを起点に，2.4 で説明した手法により，関連プロセスやファイル情報を統合ログ管理基盤から収集し，紐付ける．
⑤紐付け結果の可視化
紐付け結果をネットワーク図として可視化し，参考情報として CSV ファイルを出力する．
⑥ 分析者への通知
アラートの発生と紐付け処理完了を分析者に通知する．

4. 提案システムの評価

本章は，プロトタイプシステムを用いた提案システムの評価について説明する．具体的には，i) 具体的な攻撃シナリオを再現し，悪性活動の紐付け・可視化ができることの確認，ii) i)の紐付け・可視化が人間による分析の支援となることの確認，iii) 様々な現実の攻撃シナリオで悪性活動の紐付け・可視化ができることの確認，により，提案システムの効果を実証し，有効性を示す．

4.1 攻撃シナリオの再現と紐付け・可視化

初めに，バックドアによる不正通信からその攻撃源までを一連の悪性活動として紐付け・可視化できることを確認する．本実験で再現した攻撃シナリオは以下の通りである．なお，本実験においては，メーラとしては一般的な Thunderbird を，バックドアファイルとして pupy[8]を利用した．

① 攻撃者は，被害者の PC 上のメールアカウントへバックドアファイルの付いたメールを送付する．
② 被害者が，メーラ上でメールを開封し，バックドアファイルをダウンロードして実行する．
③ C&C サーバへのコールバックが実行される．

プロトタイプシステムによる本攻撃シナリオの紐付け・可視化結果を図 3 に示す．

図中の A は，バックドアプロセスを示しており，このプロセスが Python でコーディングされているため「Python」と表示されている．図中の B は，バックドアファイル「inode8926665」を示しており，図中 C は，バックドアファイルを生成した Thunderbird について，プロセス「pid18605 thunderbird」を示している．

A，B，C は，プロセスとファイルの関係による紐付けにより，バックドアファイル「inode8926665」とそれを作成したプロセス「pid18605 thunderbird」，バックドアファイル

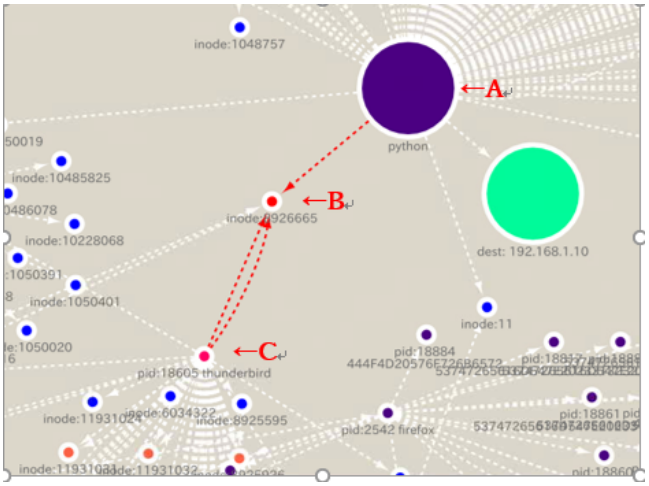

図 3. メールソフトを経由したバックドアの配送

を実行した結果生成されたプロセス「python」が関連付けられて視覚化されている．この結果から，バックドアの実体となるファイルについて，thunderbird 経由で作成された，つまり，メールからの攻撃であるという関係性を把握することができる．なお，同様の実験をブラウザを利用する Web メールにおいても実施したが，メーラが Web ブラウザに置き換わる形で，ほぼ同様の結果が得られた．

4.2 分析支援効果の評価

プロトタイプシステムの分析支援効果をアンケートによって評価する。評価実験への参加者には、4.1 の攻撃シナリオについて，攻撃源の解析を手動とプロトタイプシステム利用の双方行ってもらい、終了後にアンケートへの回答を依頼した。

4.2.1 参加者の特性

本実験では３名の協力を得た。３名とも技術系の社会人であり、コンピュータ技術者として十分な知識と経験を保有している。

4.2.2 実験方法

以下の手順で実験を実施した。

① 画面に次の設問を提示した：「社員の端末より不審な外部サーバに対して通信が発生している事を検知しました。外部サーバの IP アドレスが「xxx.xxx.xxx.xxx」であることが分かっています。社員の端末及び通信の発生時刻は特定できており、解析に必要なログも確保済です。与えられたログより、この通信がどのような経緯で発生したかを解析して下さい。」
② 手作業による解析を依頼した。外部サーバとの通信が発生した時間帯のログ（システムコール、CSV 形式）を入力として、通信が発生した経緯を解析してもらった。

③ プロトタイプシステムを利用した解析を依頼した。まず、3.1 で説明したネットワークデータを Cytoscape に読み込ませるように伝えた。続いて手順書に従ってネットワーク図を整形し（現時点では、完全な自動化ができておらず、Cytoscape に表示された初期状態から一定量の手作業が必要)、そこから通信が発生した経緯を読み取るように依頼した。

④ アンケート回答

プロトタイプシステムを使用して解析する場合と、手作業で解析する場合とを比較して、どちらの方が望ましいか回答してもらった。アンケートは質問項目 6 つから構成された。各質問項目は次のとおりである:「解析を速く行える」、「解析を簡単に行える」、「解析を正確に行える」、「解析結果を分かりやすく説明できる」、「解析で得た情報を共有しやすい」、「解析で得た情報を蓄積しやすい」。質問に対する評価は 5 段階であった。「どちらでもない」を評価値の中央（3）とし、「手作業の方が良い」（1）から「プロトタイプシステムの方が良い」（5）までの間の整数値での評価を依頼した。

4.2.3 アンケート結果

結果、総じてプロトタイプシステムを使用することについて有効であるとの評価を得た。特に、「解析結果を分かりやすく説明できる」については、3 名中 2 名が評価値 5（プロトタイプシステムを使用した方が良い）と回答した。また、質問「解析を速く行える」、「解析で得た情報を共有しやすい」、「解析で得た情報を蓄積しやすい」についても、プロトタイプシステムを使用した場合の方が良い（評価値 5)、あるいは、やや良い（評価値 4）と回答した参加者は 3 名中 2 名であった。

ただし、いずれの質問に対しても「どちらでもない」（評価値 3）あるいは「手作業の方がやや良い」（評価値 2）の回答が得られた。これについては、プロトタイプシステムの完全な自動化ができておらず、Cytoscape に表示された初期状態から一定量の手作業が必要であるために、その有用性が十分に認識されなかった可能性がある。プロトタイプシステムの有用性をユーザに認識させるためには、このような課題を解消する必要があると考えられる。

4.3 現実の攻撃シナリオにおける紐付け・可視化

4.3.1 ShellShock による CGI プログラム実行

ShellShock と呼ばれる攻撃シナリオをプロトタイプシステムで分析・紐付け・可視化した結果を図 4 に示す．この攻撃シナリオでは，bash の脆弱性（CVE-2014-6271）により，任意のコマンドが実行される．

図 4 中の A~E は，各々以下を示している．

A. バックドアプロセス「UQCBT」
B. バックドアファイル「inode 11479」
C. バックドアファイルを作成したプロセス「pid25691 bash」
D. E から生成されたプロセス「pid25690 bash」
E. プロセス「pid23509 apache2」から生成されたプロセス「pid25687 test.cgi」

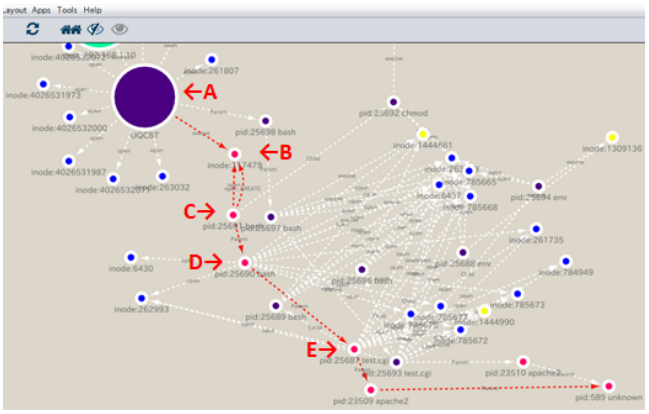

図 4. ShellShock の可視化

A，B，C は，プロセスとファイルの関係による紐付けにより，バックドアファイル「inode 11479」とそれを作成したプロセス「pid25691 bash」，バックドアファイルを実行した結果生成されたプロセス「UQCBT」が関連付けられて視覚化されている．また，D，E は親子関係により紐付けられている．

この可視化結果から，バックドアの実体となるファイルについて，Apache2 プロセスを起点として生成されたものであることが可視化されており，分析者はより正確な状況を把握することができるといえる．

4.3.2 Drupalgedon

プロトタイプシステムにおいて，CMS である Drupal の脆弱性（CVE-2014-3704）[10]を利用した攻撃を分析・紐付け・可視化した結果を図 5，図 6，図 7 に示す．この攻撃シナリオでは，SQL インジェクションを実行することで管理者権限の Drupal ユーザを作成し，任意のコードを実行することが可能となる．

図 5 中の A はバックドアプロセス「perl」を，B は A の親プロセス「pid3750 perl」を各々示す．また，図 5 に続く紐付け・可視化結果を図 6 に示す．図 6 中の C は，B の親プロセス「pid 3749 sh」を示している．

図 6 に続く紐付け・可視化結果を図 7 に示す．図 7 中の D は C の親プロセス「pid23543 apache2」を示している．

これらをまとめると，A，B，C，Dは，親子関係により紐づけられていることがわかる．すなわち，この可視化結果においては，apache2 プロセスからシェルスクリプトが起動し，最終的に perl によるリバースシェルが実行され

たことが確認できる．また，リバースシェルのプロセスで実行した whoami などのコマンドが子プロセスとして実行されていること等も視認できる．

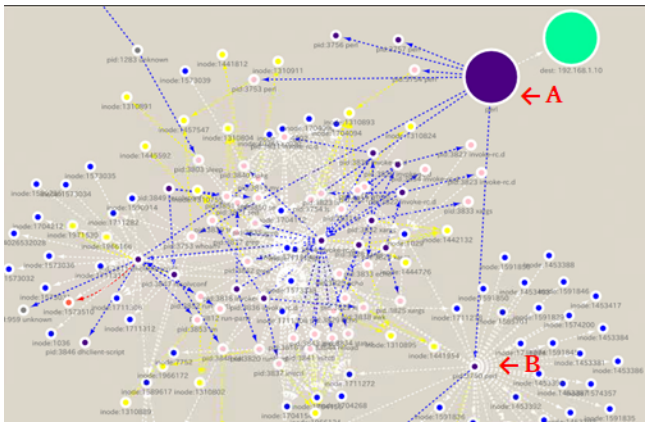

図 5. Drupalgedon の可視化①

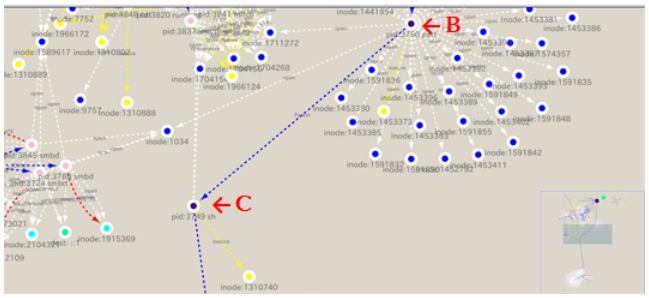

図 6. Drupalgedon②

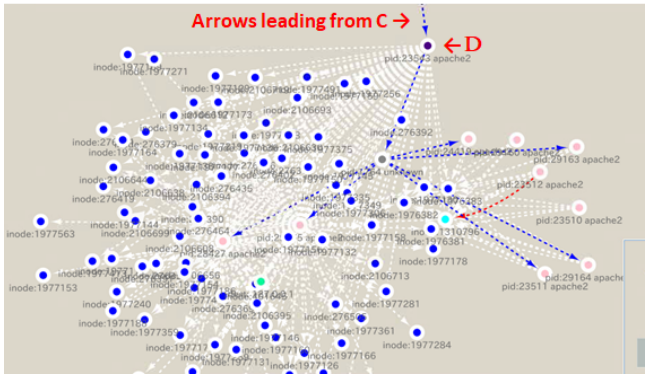

図 7. Drupalgedon③

4.3.3 overlayfs の脆弱性を悪用した権限昇格

特定バージョンの Linux に搭載されたオーバーレイファイルシステムの権限チェックに関するバグにより，通常のユーザ権限から特権ユーザへ権限昇格が可能となる脆弱性（CVE-2015-1328）[11]を利用した攻撃について，プロトタイプシステムを用いて分析・紐付け・可視化した結果を図 8，図 9 に示す．

図 8 中の A，B，C，D，E，F，G，H，I は，各々以下を示している．

A. proftpd デーモンの脆弱性（CVE-2015-3306）を利用してバックドアとして起動したプロセス「pid553 perl」（一般ユーザ権限）
B. A が起動した「pid564 curl」
C. B によるネットワーク接続先の IP アドレス「192.124.249.8」
D. B により作成された「inode1180288」
E. A が起動した gcc コマンド「pid:572 gcc」．F と G を子プロセスとして起動している．
F. コンパイルを行う「pid:573 cc1」．D を open している．
G. アセンブラとリンカの中間に位置する「pid575: collect2」．
H. リンカである「pid576:ld」．
I. コンパイルの結果生成されたファイル「inode:1180327」．これは D をコンパイルした結果の実行ファイルである．

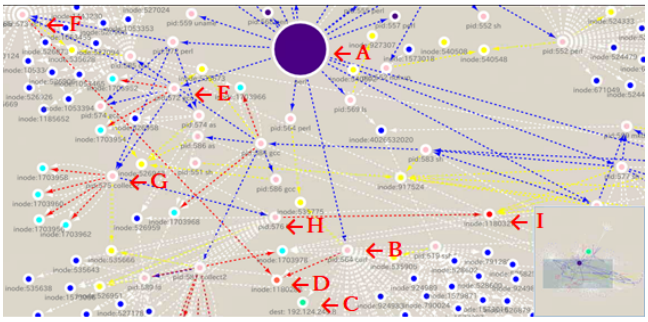

図 8. overlayfs の脆弱性に対する攻撃の可視化①

なお，「inode1180288」は，overlayfs（CVE-2015-1328）の脆弱性を攻撃するための攻撃コード本体（コンパイル前の C 言語によるソースコード）である．

また，図 8 に続く紐付け・可視化結果を図 9 に示す．図 9 中の J，K，L は，各々以下を示す．

J. 「pid:577 cve-2015-1238」．I を実行し，バックドアが生成されている．
K. 「pid:577 sh」． J が同じ pid で変化したもの．
L. 「pid:577 su」．一連の攻撃コードが実行された後に実行される su コマンド．この実行され，root 権限のシェルが奪取される．図では省略されているが，バックドアから「pid:577 /bin/dash」が clone され，以降のコマンドは pid:577 から実行される．

この可視化結果より，脆弱性を実行するコードが外部からダウンロードされ，コンパイルされた後実行されていることが確認でき，また，その後の攻撃コードの動きと，su コマンドによって root 権限が奪取される状況も確認できる．

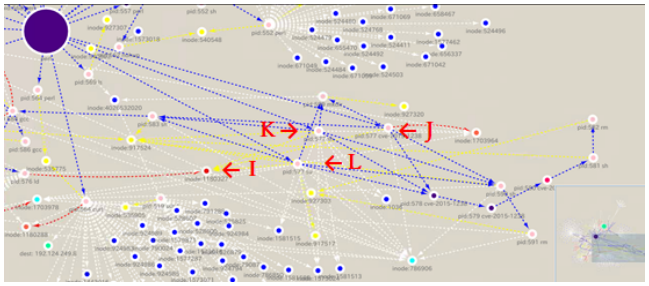

図 9. overlayfs の脆弱性に対する攻撃の可視化②

5. 関連研究

システムコールの情報から攻撃活動に関するログを関連付けする研究に関して, 多くの研究がなされている. King らは攻撃の検知点を起点としてプロセスやファイルの依存関係をグラフによって可視化し，侵入経路調査を支援するツールである BackTracker を提案している[12]. Backtracker では，プロセス同士，プロセスとファイル，プロセスを基に可視化を行うが，依存関係の爆発する問題が発生した．そのため，BackTracker では，特定のファイルやイベント，操作を除外するフィルタを設定することでこの問題に対処した.

ログを関連付ける際の依存関係の爆発問題に関しては，様々な研究においてその解決が試みられている．Lee らはシステムコールのログを利用したフォレンジック分析における依存関係の爆発を解決するため，バイナリプログラムのロギングを行う BEEP（Binary-based ExEcution Partition）を提案している[13]. BEEP はアプリケーションをリバースエンジニアリングすることにより，繰り返し実行されるコードを訓練し，ループと判断し除外している．例えば被害が発生した際に特定のファイルを起点として read/write しているプロセスなどの分析を行う場合，BEEP によって繰り返しの部分が除外され，純粋に攻撃に関連した情報だけが抽出できるとしている．また，Ma らは Windows で利用できる ETW (Event Tracing for Windows) の情報を用いて，オーバーヘッドを抑えつつ自動的にイベントループを見つけ出す手法を提案をしている[14]. Ma らの手法では，ログ収集，Prefix（接頭辞）分析，プログラム分析，unit の認識，因果関係の分析，洗練化の 6 手順により，イベントのモデル化と不必要な情報を省いたグラフ生成を可能とした.

6. まとめと今後の課題

6.1 まとめ

本稿は，侵入検知に向けたシステム内悪性活動の紐付け・可視化手法について説明し，そのプロトタイプ実装を用いた評価結果を報告するものである．提案システムは，5 つの Linux システムコールに着目し，この集積と分析，可視化を実施するものであり, プロセス-プロセスの紐付けとプロセス-ファイルの紐付けを半自動的に行うことで, 一連の悪性活動を分析者にわかりやすく提示することを目指すものである.

プロトタイプ実装を用いた評価は，主に具体的な攻撃シナリオを再現し, 悪性活動の紐付けができることの確認と，本システムが人間による分析を支援でき得ることを確認するためのアンケート調査により実施した．これらの評価により，提案システムが，様々な現実の攻撃シナリオを再現した上で半自動の紐付け・可視化が可能であることが実証され，また，人間にとって，プロトタイプシステムの利用が有効であることが認められた.

6.2 今後の課題

本研究の今後の課題は以下の 5 項目である.

① 依存関係の爆発問題への対処

一般に，情報システム内のプロセスやファイル間には膨大な依存関係が存在し得る．既存研究においては，依存関係の爆発問題として認識されており，ループ処理に起因するログの除外や，タグ付けによる情報付加等の工夫による対処が検討されている.

これに対し本研究では、解析対象とするシステムコールを限定することで、依存関係の爆発問題に対処することを試みた。しかしながら、評価実験により，様々な攻撃シナリオを紐付けできることを実証したが，この理論的な裏付けは今後の課題である．また，攻撃に関連する紐付けのみを効率的に抽出するための工夫として，良性活動に関する依存関係については，事前に定義したホワイトリストによって分析対象から除外する等の処理を追加することを検討している.

② 必要十分な分析期間の設定方法

プロトタイプシステムでは，分析に利用するシステムコールログを検索する際, 攻撃アラート検知時点から 20 分前までのログを対象と想定したが，それでは不十分である可能性がある．特に攻撃者の侵入から実際の攻撃までが数日をかけて行われる事例も存在することから，分析に利用するファイルの最終アクセス時刻を利用するなど，なんらかの指標を検討する必要がある．その際には，課題①で示した依存関係の爆発問題も対処できている必要があるものと思われる.

③ 時系列情報の表現

プロトタイプシステムにおいては，分析期間中のプロセス-プロセス間，プロセス-ファイル間の紐付け・可視化は実現しているものの，現状では，時系列情報を利用してい

ない．しかしながら，時系列情報を付加して紐付け・可視化を行うことで，提案システムの有効性・効率性を向上できる可能性があると考えている．そのためには，ネットワーク図を拡張し，時系列を理解できる表現方法を実現する必要がある．

④ 紐付け及び可視化の完全自動化

4.2 で示した通り，プロトタイプシステムを利用した人間による攻撃の分析作業では，現状，紐付けされた様々なシステム内活動から悪性活動を抽出して視認するために，一定量の手作業が必要である．手作業を最小限に止める又は完全自動化のために，課題①~③の解決を含む，分析・紐付けプログラムの改良が必要であり，これは今後の課題である．

⑤ 分析支援効果についての評価の継続

分析支援効果の評価に関する今後の課題の 1 つ目は，より厳密な実験計画によって，改めて提案システムを評価することである．参加者 3 名全員には，1 種類のシナリオを手作業で解析した後，提案システムを利用して同じシナリオを解析するよう依頼した．先の手作業による解析の結果を知った上で，同じ内容を再度解析すれば，参加者は 2 回目の解析を容易に感じやすいと考えられる．今回の結果からは，提案システムに対する肯定的な評価が，提案システム自体に対する評価なのか，あるいは，2 回目の解析に対する容易さを反映したものなのかは判断することが難しい．

2 つ目の課題は，評価人数を増やすとともに，特に，経験が浅い技術者による提案システムへの評価結果も得ることである．悪性活動の解釈には多くの知識や経験を要し，これらを十分に備えた技術者への負担を軽減するためには熟達していない技術者でも解析作業を担い，かつ，これらの技術者が悪性活動を解釈するための知識等を効率的に習得できる環境を整えることが望ましい．このような目的を到達するために提案システムが有効であるか，引き続き検証する価値があると考える．

参考文献

[1] L. Zeng, Y. Xiao and H. Chen, "Linux auditing: Overhead and adaptation," 2015 IEEE International Conference on Communications (ICC), 2015, pp. 7168-7173, doi: 10.1109/ICC.2015.7249470.

[2] Auditbeat: Lightweight Shipper for Audit Data | Elastic. “Auditbeat: Lightweight Shipper for Audit Data | Elastic”.url:https://www.elastic.co/products/beats/auditbeat (visited on June 25, 2021).

[3] Logstash: Collect, Parse, Transform Logs | Elastic. “Logstash: Collect, Parse, Transform Logs | Elastic”.url:https://www.elastic.co/logstash (visited on June 25, 2021).

[4] Gormley, C. and Zachary Tong. “Elasticsearch: The Definitive Guide.” (2015).

[5] Białecki, Andrzej, et al. "Apache lucene 4." SIGIR 2012 workshop on open source information retrieval. 2012.

[6] NetworkX - NetworkX documentation. “NetworkX - NetworkX documentation”.url:https://networkx.org/ (visited on June 25, 2021).

[7] Cytoscape: An Open Source Platform for Complex Network Analysis and Visualization. “Cytoscape: An Open Source Platform for Complex Network Analysis and Visualization”.url:https://cytoscape.org/ (visited on June 25, 2021).

[8] GitHub - n1nj4sec/pupy: Pupy is an opensource, cross-platform (Windows, Linux, OSX, Android) remote administration and post-exploitation tool mainly written in python. “GitHub - n1nj4sec/pupy: Pupy is an opensource, cross-platform (Windows, Linux, OSX, Android) remote administration and post-exploitation tool mainly written in python”.url:https://github.com/n1nj4sec/pupy (visited on June 25, 2021).

[9] NVD - CVE-2014-3704. “NVD - CVE-2014-3704”.https://nvd.nist.gov/vuln/detail/CVE-2014-3704 (visited on June 25, 2021).

[10] NVD - CVE-2015-1328. “NVD - CVE-2015-1328”.https://nvd.nist.gov/vuln/detail/CVE-2015-1328 (visited on June 25, 2021).

[11] Samuel T. King and Peter M. Chen. 2003. Backtracking intrusions. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). Association for Computing Machinery, New York, NY, USA, 223–236. DOI:https://doi.org/10.1145/945445.945467

[12] Lee, K., Zhang, X., and Xu, D. High accuracy attack provenance via binary-based execution partition. In Proceedings of Network and Distributed System Security Symposium (NDSS), 2013.

[13] Shiqing Ma, Kyu Hyung Lee, Chung Hwan Kim, Junghwan Rhee, Xiangyu Zhang, and Dongyan Xu. 2015. Accurate, Low Cost and Instrumentation-Free Security Audit Logging for Windows. In Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015). Association for Computing Machinery, New York, NY, USA, 401–410. DOI:https://doi.org/10.1145/2818000.2818039