Vol.2020-CSEC-90 No.24
Vol.2020-SPT-38 No.24
2020/7/21

のアラートの他，Linux OS が稼働するマシンの詳細な活動を示すシステムコールの情報を統合ログ管理環境に集約した．そして，侵入検知の際に悪性と考えられるプロセスとそれに関連する活動を視覚的に分析するための紐付け手法とその実装を検討し，評価を行った．

評価では実際に攻撃の再現を行い，そこから取得されるログを用いて活動の紐づけ及び視覚化を行った．評価の結果，プロセス間の親子関係による紐づけにより，攻撃の原因となるプロセスを把握できることが確認できた．

これにより，アラート発生時，詳細な調査を行う判断の段階で OS の挙動についての調査を行うことが可能となり，分析作業を簡略化し，解析が必要な攻撃の選別を支援することが期待できる．

以降本稿では，第 2 章で関連研究に関する調査と，本研究との相違点を述べる．第 3 章では，紐づけを行うためのプロトタイプシステムの検討内容について述べ，第 4 章で実装とその評価について述べている．第 5 章では，本研究の結論と明らかとなった課題について述べる．

2. 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，複数の研究がなされている．[2,3,4,5]　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[6].

Backtracker では以下の依存関係を取得して可視化を行う．

① プロセス同士
あるプロセスが別のプロセスの実行に直接影響するもの．プロセスの親子関係，同じファイルの読み書き，共有メモリなど

② プロセスとファイル
ファイルに関連したデータや属性に影響を与えたり受けたりするプロセスに関するもの．（write, read，execve，chown，chmod 等）

③ プロセスとファイル名
ファイル名オブジェクトに影響を与えたり受けたりするプロセスに関するもの．引数としてファイル名を利用するシステムコールが該当する．（open，creat，link，unlink，rename，mkdir，rmdir，and mount 等）

上記依存関係を基に可視化を行うと，依存関係が大きくなり過ぎる問題が発生する．BackTracker では特定のファイルやイベント，操作（read されたが write されていないもの）を除外するフィルタを設定することで対処している．

Backtracker で挙げられた依存関係の爆発に関しての解決策が複数検討されている．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）というスキームを提案している[7] ．また Ma らは Windows で利用できる Event Tracing for Windows(ETW)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提案をしている[8].

King らの研究では，本研究と同様に Linux 上でシステムコールのログ情報を取得し，関連付けの上，可視化するものであった．一方，Lee らや Ma らは，バイナリプログラムのログや Windows でのログの可視化を提案している．

3. プロトタイプシステムの検討

本章では，本研究にて検討したプロトタイプシステムの構成要素および紐付けアルゴリズムについて述べる．

3.1 ホストログ

LinuxOS が搭載する Audit フレームワークを用い，システムコールの情報を取得する．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，本研究で利用している主なシステムコールを以下の表に示す．

表 1　主なシステムコール

No	システムコール名	内容
1	open	指定したファイルを開く．CREATE オプションをつけることによって，ファイルを作成する際にも用いられる．
2	execve	指定したファイルを実行する．実行したコマンドなどの情報が分かる．
3	connect	指定した宛先と通信する．本研究では IPv4 アドレスの情報を用いる．
4	unlink	指定したファイルを閉じる．
5	clone	プロセスが子プロセスを生成する際に用いられる．親子関係の情報を取得するために利用．

またシステムコールには表 2 に示すような情報を含んでおり，システム内プロセス活動の詳細分析を可能とする．

表 2　Open システムコールに含まれる主な情報の一覧

No	項目	値（例）	内容
1	auditd/data.syscall	open	システムコール名
2	file.path	/home/wk/xx	ファイルパス
3	file.inode	8925028	inode 番号
4	file.mode	644	権限情報
5	process.exe	/usr/lib/firefox	実行ファイル
6	process.name	firefox	プロセス名
7	process.pid	3523	プロセス ID
8	process.ppid	1873	親プロセス ID
9	user.uid	1000	ユーザーID

3.2 ログの収集・保管・監視

前述したホストログの収取，送付，受信，分解を行うツールとして，Auditbeat と Logstash を利用する．Auditbeat はログの収集，送信を行い，Logstash により受信，分解を行いログの保管を行う．またログの保管は Elasticsearch[9]を統合ログ管理基盤として利用し，監視及びスクリプトの実行については，ElastAleart[10]を利用する．

3.3 プロトタイプシステムの概要と動作フロー

3.3.1 概要

プロトタイプシステムは前項までに述べたログを収集・保管する仕組みを用い，侵入検知システムのアラート検知をトリガーとしてシステム内活動の収集と紐付けを自動的に行うものである．また，ネットワーク図を用いた視覚的なデータを生成し，分析者がアラートの対象となるプロセスを中心として関係するプロセス，ファイル，通信等を確認し，攻撃の元となっているプロセスや悪性ファイルを特定することができる．

データとしてシステムコールの情報を利用するため，詳細かつ正確な情報が利用できる一方，情報量が非常に多いという特徴がある

3.3.2 パラメータ仕様

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 **インデックスの指定**

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．インデックスはホスト単位で作成され，システムコールの情報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の範囲を指定する．デフォルトで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**

攻撃活動として検知された通信先 IP を指定する．この IP を基に 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．

3.3.3 動作フロー

① **侵入検知システムによるアラート検知**

IDS/IPS 等によって攻撃やその疑いのある通信・挙動が検知され，統合ログ管理基盤にアラートが送付される．

② **攻撃活動の調査**

アラートの発生をトリガーとし，攻撃活動の調査が開始される．悪性通信に関するアラートを例にとると，アラートに記載された送信元 IP からホストを特定し，宛先 IP からプロセス（通信に用いる connect システムコールを利用しているもの）を特定し，調査を開始する．

③ **紐付け・ネットワーク図生成・描画**

特定したプロセスに関連する別の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，後述の紐付けアルゴリズムに従って紐付けする．紐付けした結果はネットワーク図として生成・描画し， 参考情報として CSV ファイルを出力する．

④ **通知**

アラートの発生と紐付け処理完了を利用者に通知する．

3.4 紐づけアルゴリズム

紐付けのアルゴリズムは以下の 3 通りに大別される．

3.4.1 親子関係に基づく紐付け

① 親プロセス方向

Linux におけるプロセスには親子関係が存在する．調査対象となる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，通信先等の情報を取得する．調査対象プロセスと親プロセスの関係は 1 対 1 となる．

② 子プロセス方向

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が得られる．調査対象プロセスと子プロセスの関係は 1 対多となる．

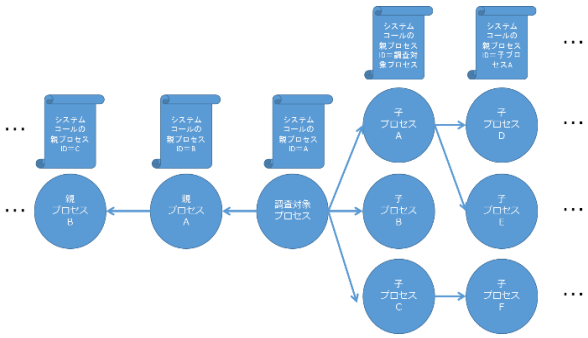

図 1　親子関係に基づく紐付け

3.4.2 ファイルとの関係に基づく紐付け

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，そのファイルに対して同じく open システムコール（CREATE オプション：ファイルを作成する）を呼び出しているプロセスを検索し紐付ける．関係図を以下に示す．

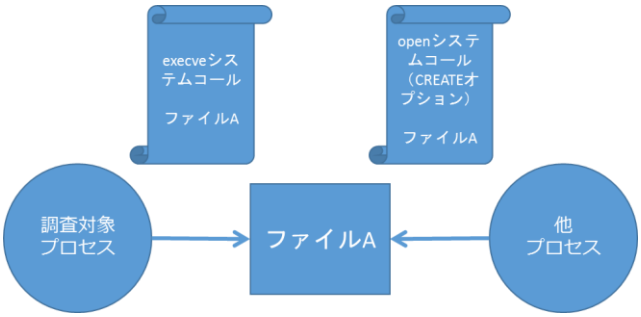

図 2　ファイルとの関係に基づく紐付け

上記で示した関係性の他，以下表の組み合わせを用いてプロセスを紐付ける

表 3　ファイルに対して呼び出しているシステムコールによるプロセスの紐付けの組み合わせ

No	システムコール名	内容
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)

3.4.3 通信対象に基づく紐付け

調査対象プロセスが通信している宛先 IP アドレスに対し，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 connect システムコールで通信している宛先 IP と同じ IP を宛先として connect システムコールを呼び出しているプロセスの情報を紐付ける．関係図を以下に示す．

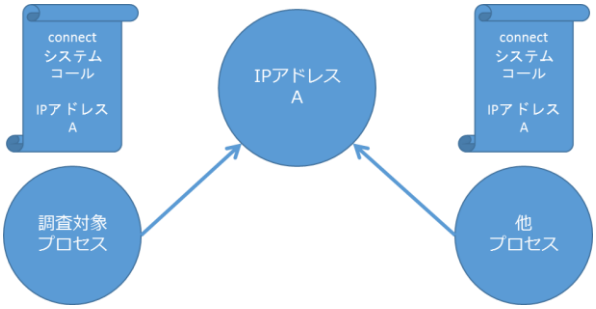

図 3　同じ宛先へ通信する関係に基づく紐付け

なお，この紐付けアルゴリズムに関しては設計のみで実装はしていない．

4. 評価と考察

本章では，前述で述べたシステムと攻撃活動分析プログラムを用いて，攻撃の再現と分析を行った結果について述べる．

4.1 実験環境

前章で検討したプロトタイプシステムや実験環境を以下のような環境で構築し，評価を実施した．

- 統合ログ管理サーバー：Ubuntu16.04．4vCPU,18GB メモリ，700GB HDD を割り当て，プロトタイプシステムを実装した．
- 被害端末：Ubuntu16.04．4vCPU,6GB メモリ，180GB HDD を割り当て，攻撃活動を再現し，分析用のログデータを収集するためのマシン
- 攻撃端末：KaliLinux(Kali GNU/Linux Rolling 2019.2)．2vCPU，4GB メモリ，300GB を割り当て，被害端末に対して攻撃の再現を行うためのマシン．

4.2 実験内容

攻撃端末を疑似的な C&C サーバーとして準備し，被害端末上のメーラーとして Thunderbird，バックドアファイルとして Python で作成されたリモートアクセスツールの Pupy を利用した．

攻撃シナリオは，攻撃端末から被害端末のメールアカウントへバックドアファイルを送付し，そのファイルをダウンロード及び実行し，攻撃端末へコールバックが実行される状況を再現した．分析は IPS がコールバックを検知したことをトリガーとして行った．

実験を通して攻撃の再現を行った際に得られたシステ

ムコールの情報を，攻撃活動分析プログラムにて分析を行った．

4.3 結果及び評価

分析した結果の攻撃活動をネットワーク図で示したものを以下に示す．

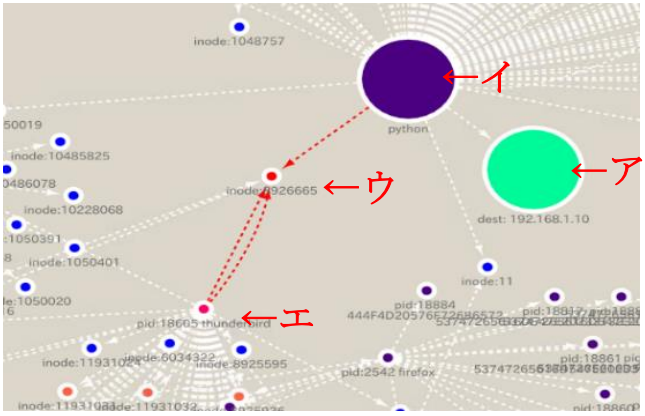

図 4　メールソフトを経由したバックドアの配送

作成されたネットワーク図から以下のことがわかる．

(ア) C&C サーバーへ通信がバックドアプロセス「Python」によって行われたことを表す．
(イ) バックドアプロセス「Python」（リモートアクセスツールが Python でコーディングされているため）を表す．
(ウ) バックドアのファイル「inode 8926665」を表す．
(エ) バックドアファイルの実体を生成したプロセス「pid18605 thunderbird」を表す．

上記の内容から，C&C サーバーへの通信を行ったプロセスを辿っていくことにより，どのような経緯よって悪性活動が引き起こされるかを把握することができる．

上記のイ，ウ，エにおける関係性はファイルとの関係に基づいた紐付けとなっており，バックドアファイルの実体「inode 8926665」とそれを作成したプロセス「pid18605 thunderbird」，バックドアファイルを実行したプロセス「python」との関係性が表されている．このことから Thunderbird によりバックドアファイルが作成され，その作成されたバックドアが Python によって実行され，最終的に C&C サーバーへの通信が発生したことがわかる．このことからシステム内での悪性活動について，各プロセス間の紐付けがされており，かつネットワーク図による可視化を行うことができた．さらには可視化を通じてメールを経由した攻撃であるという，一連の悪性活動を把握することが可能となった．

4.4 実験結果の考察

前述した実験以外にも ShellShock を利用したバックドアファイルの作成や，CMS である Drupal の脆弱性（CVE-2014-3704）を利用した任意コードの実行など，いくつかの攻撃シナリオの再現し，分析の際には親子関係による紐付けから攻撃の原因となるプロセスを把握できることがわかった．また，メールを介してバックドアが保存・実行される場合や，不正に作成されたファイルを介し，別のプロセスから攻撃が行われる場合など，ファイルとの関係に基づく紐付けによって分析をする場合においては，ネットワーク図の特性が活かされており，ファイルとそれに関係するプロセスが視覚的に理解し易いかたちで表現出来ていると考える．

関連研究比較すると取得するシステムコールの種類が 5 つと少なく，その結果，取得するログ容量も依然として改善は必要であるが、抑えられることができた．

5. まとめと課題

5.1 まとめ

本研究では，システム内悪性活動の紐付け手法と可視化を検討した．検討には Linux において監査ログを取得する Audit フレームワークより，特定のシステムコールの情報を用いた．侵入検知手法に関しては既存の環境にある IDS/IPS による検知をトリガーとし，特定ホストの悪性活動の紐付けを行う想定で行った．紐付けにはプロセスの親子関係，ファイルとの関係を用いることで，必要な範囲のログを収集した．いくつかの攻撃シナリオを再現した結果，攻撃活動の分析に必要な情報の紐付け，可視化が可能であることを確認した．特にファイルとの関係を用いた紐付けにより，通常は関連付けることに手間がかかる，親子関係の無いプロセス同士を結び付け，1 つの活動として分析するにあたって必要な情報として取得できることを示した．

本研究で提示したプロトタイプを SOC の監視員が活用することで，IDS/IPS の情報に加え，より詳細なシステム内の活動を特別な作業を行わずに分析することが可能になると考えられる．特にネットワークのログだけを監視している環境において誤検知が頻繁に発生する状況であっても，本当に対応が必要な分析につながると考えられる．

また，インシデント発生時の攻撃活動調査をする際にはシステムを停止し，OS に精通した技術者による調査を行うことがある．プロトタイプで利用しているシステムコールは得られる情報の粒度が細かく，全てのシステム内活動を漏れなく取得し，それらをニアリアルタイムで解析可能な仕組みとなっているため．予め詳細に調査する必要のあるプロセス，ファイルを一覧化できる等，攻撃調査を簡略化することにも繋がると考えられる．

5.2 課題

5.2.1 依存関係の爆発

本研究における紐付けと可視化手法では，本来分析に不

要な情報が多く含まれることがある.関連研究においては,「依存関係の爆発」という課題として認識されており，ループ処理によって発生するログの除外や，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．本研究ではこの課題に対する明確な答えが見出せていないが，取得した情報を用いて悪性の活動を定義することで，必要な情報だけを取得することに繋がっていくと考える．そのために，システムコールから得られる情報をプロセス ID やユーザーID で括り, 活動として定義する方法を検討するなどの更なる工夫が必要になると考えられる.

5.2.2 ログ容量

分析に利用するシステムコールログの容量が膨大であることも課題として挙げられる．実験では，特別なタスクの処理を行わないサーバーであっても，1 日あたり 1GB のログを記録することもあった．実際に定期的な処理やサーバーとしての機能を提供する実システム環境におけるサーバーでログを取得する場合は更に多くのシステムコールログが発生すると考えられるため，取得するシステムコールを厳選し，ログ管理サーバーでは，必要な情報だけを保持するチューニング等が必要になってくると考えられる.

5.2.3 分析期間

プロトタイプでは分析に利用するシステムコールのログを検索する際，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ているが，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．その際に依存関係の爆発も考慮されていることが望ましい.

5.2.4 時系列情報の表現

分析期間中のプロセスとファイルの関係性の可視化は実現しているものの，ネットワーク図に時系列情報が存在しないため，攻撃者の行動についての詳細な分析を行う際には，直接システムコールのログを参照する必要がある．ネットワーク図の特性上，実現が難しいと考えられるが，時間軸を追加する等，時系列を理解できる図を実現することができれば，より効率的な攻撃調査が可能になると考えられる.

5.3 今後について

上述の課題があるものの，本研究ではプロセスとファイルの関係性について，ネットワーク図を用いて俯瞰的に把握できることを示した.

研究の本来の目的として，OS 内活動の抽象化を目指しているため，引き続きその点での検討を進めていく必要がある.

今後は，ネットワーク図を用いた分析の課題として挙げた時系列情報を表現しつつ，そのグラフ上で OS 内活動を要約することを検討していく．例えば特定期間中に取得したシステムコールログについて, ユーザーID とプロセス ID でまとめたものを「活動」として要約していき，それら要約をさらにユーザーID で束ねて要約していくこと等を目指していく.

参考文献

[1] 総務省. “我が国のサイバーセキュリティ人材の現状について” https://www.soumu.go.jp/main_content/000591470.pdf (参照　2 2020-05-12)
[2] A. Goel, W.-C. Feng,D. Maier, and W.-C. Feng,J. Walpole. Forensix: A Robust, High-Performance. 25th IEEE International Conference on Distributed Computing Systems Workshops, 2005, page144-151
[3] Md Nahid Hossain1, Sadegh M. Milajerdi2, Junao Wang1, Birhanu Eshete2, Rigel Gjomemo2,R. Sekar1, Scott D. Stoller1, and V.N. Venkatakrishnan2. SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data. 26th USENIX Security Symposium, 2016, page487-504
[4] Wajih Ul Hassan, Mark Lemay, Nuraini Aguse, Adam Bates and Thomas Moyer. Towards Scalable Cluster Auditing through Grammatical Inference over Provenance Graphs.NDSS2018, 2018.
[5] Devin J. Pohly, Stephen McLaughlin, Patrick McDaniel, and Kevin Butler.Hi-Fi: Collecting High-Fidelity Whole-System Provenance. ACSAC '12: Proceedings of the 28th Annual Computer Security Applications Conference, 2012, Page 259–268
[6] Samuel T. King, Peter M Chen. Backtracking Intrusions ACM SIGOPS Operating Systems Review. 2003, Volume 37, Issue 5
[7] K. H. Lee, X. Zhang, and D. Xu. High Accuracy Attack Provenance via Binary-based Execution Partition. NDSS Symposium 2013. 2013.
[8] Shiqing Ma et al. Accurate, Low Cost and Instrumentation-Free Security Audit Logging for Windows. Proceedings of the 31st Annual Computer Security Applications ConferenceDecember, 2015, page401-410.
[9] Elasticsearch https://www.elastic.co/jp/products/elasticsearch
[10] ElastAlert - Easy & Flexible Alerting With Elasticsearch https://elastalert.readthedocs.io/en/latest/
[11] Thunderbird https://www.thunderbird.net/ja/
[12] pupy https://github.com/n1nj4sec/pupy
[13] drupal https://www.drupal.org/