ランサムウェア対策及び情報漏洩防止のためにファイルアクセスログに基づくファイルアクセス制御システム

이한수 [O1] 김동주 [1] 이혁준 [1] 황동혁 [2]

[1] 광운대학교 컴퓨터정보공학부, [2] 테르텐

lhs1438@gmail.com, gggg8657@gmail.com, hlee@kw.ac.kr, mask@teruten.com

A File Access Control System Based on File Access Logs for Ransomware Response and Data Loss Prevention System

Hansu Lee [O1] Dongju Kim[1] Hyukjoon Lee[1] Donghyuk Hwang[2]

[1]Department of Computer and Information Engineering, Kwangwoon University

[2]Teruten

あらすじ

本論文ではエンドポイントで実行されたプロセスのファイルアクセスを制御することでランサムウェア攻撃からユーザーを保護し情報漏洩を遮断するシステムを提案する。多数のPCでファイルアクセスログを集め、これらを使ってファイルアクセスホワイトリストを作ることでファイルアクセスを制御する。ファイルアクセスログの収集と制御のためファイルアクセスAPIをフックする。ファイルアクセスホワイトリストはABACポリシー形式で実装する。

1. まえがき

コロナウイルスの影響でリモートワーク環境が拡散したことで全世界的にサイバーセキュリティの重要性が高まっている。2020年のグローバル企業の情報漏洩事故の主な原因は発生数が多い順に悪意的な攻撃（52%）、システムの欠陥（25%）、ヒューマンエラー（23%）などの順番であって[1]、最近はランサムウェア攻撃と情報漏洩が一緒に生じることが増えている[2]。また、2021年は全世界の大企業と中小企業共にネットワーク及びエンドポイント保安、識別・接近管理部門の市場がさらに成長することに予想している点で確認できる[3]。プロセスのファイルI/Oを制御しランサムウェアを含めたすべてのプロセスのファイルアクセスを確認することができる。このことに基づいて短時間内に1つのプロセスからファイルI/O要請を頻繁に送ると不正なファイルアクセスと判断しブロックしたり[4]、 重要なファイルだけファイルの読みとりまたは書き込みの権限を制限したりする方法でランサムウェアからファイルを保護する研究がある[5] 。しかし、これらの研究はI/O interval基盤で誤探知の可能性があり、 保護するファイルを自ら選ばなければいけない限界がある。ランサムウェアが情報漏洩を伴う時は完全に対処できないことも限界である。

本論文ではプロセスからファイルアクセスログを収集してABACポリシー形式のファイルアクセスホワイトリストを作る。これを使ってプロセスのファイルアクセスを制御することでランサムウェア等からの情報漏洩を保護するシステムを提案する。システムは１）ログ収集サブシステム、２）ファイルアクセス分別サブシステム、３）ファイルアクセス制御サブシステムの3つで構成されて、ファイルアクセス制御サブシステムのファイルアクセス環境をログ収集サブシステムと似たように作って多数のエンドポイントPCでファイルアクセスを制御する。ABACポリシーを用いるためI/O intervalが頻繁な一般プロセスを誤探知しない故に、保護するファイルを選ぶこともない。また、ランサムウェアだけでなく認可されていないプロセスの情報漏洩と一般プロセスの不正なファイルアクセスもブロックできる。

2. ファイルアクセスログ基盤ファイルアクセス制御システムの構造

2.1 全体システムの構造

本論文で提案するファイルアクセス制御システムは図1と同様にログ収集サブシステム、ファイルアクセス分別サブシステム、ファイルアクセス制御サブシステムの3つのサブシステムで構成されている。ログ収集サブシステムは安全性が確認された複数のPCで構成されていて、このサブシステムにあるPC上で実行されたプロセスからファイルアクセスログを収集する。こうして収集したファイルアクセスはすべて安全なものだと見なす。ファイルアクセス分別サブシステムではログ収集サブシステムとファイルアクセス制御サブシステムから収集したブロックログは必要な場合管理者の判断で許可できるようにする。

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학 지원사업의 연구결과로 수행되었음 (2017-0-00096).

ファイルアクセス制御サブシステムもまた多数のPCで構成して、これらのPCではホワイトリストからファイルアクセスを許可またはブロックすることを決める。ブロックが行われた場合、ブロックログを記録する。ログ収集サブシステムのブロックログの中から管理者が許可したログに基づいてホワイトリストを作るため、ログ収集サブシステムのファイルアクセス環境とファイルアクセス制御サブシステムのファイルアクセス環境は似てくる。各サブシステムの役割がそれぞれ異なって、各サブシステムを構成するPCの数を必要に応じて流動的に整えるため分けた。各サブシステムの間でのファイル転送はネットワークを介して行われる。

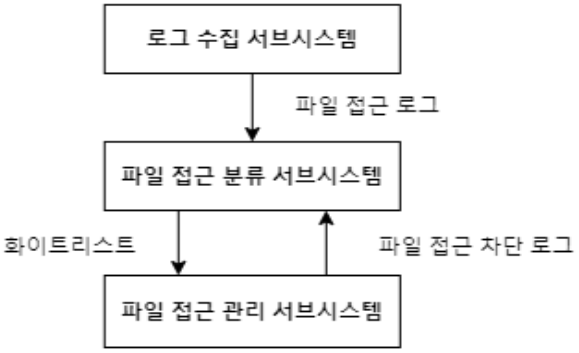

図 1. 全体システムの構造

2.2 ログ収集サブシステム

ログ収集サブシステムは図2のようにファイルアクセスをロギングするDLLファイル、プロセスの実行を探知しDLLインジェクションをするプロセスモニター、ファイルアクセスログファイルを周期的に自動転送するスクリプトの3つで構成されている。まずファイルアクセスをロギングするDLLはntdll.dll内部のZwReadFile等のファイルアクセスAPIをフックし、ロギングする。こういったログにはプロセスのプログラム名、ファイルアクセス関数の種類、アクセスしたいファイルの拡張子などの情報が存在する。ログはJSON形式とする。プロセスモニターはOSが始まった時点で自動に実行され、OSから実行されたプロセスを探知してDLLインジェクションをする。最後に、収集したファイルアクセスログはタスクスケジューラを通じて周期的にファイルアクセス分別サブシステムへ送る。ログはzipで圧縮されネットワークを介して送る。

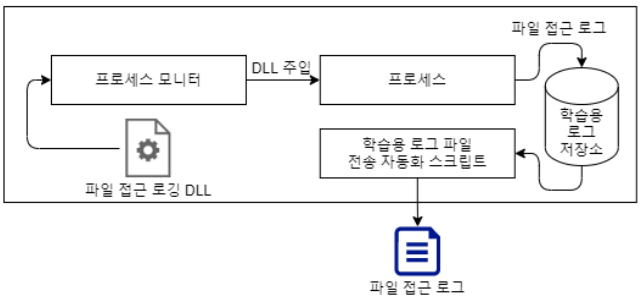

図 2. ログ収集サブシステムの構造

プロセスモニターではプロセスをランタイム中メモリーレベルでフックする。この論文ではオープンソースのDetoursを用いてフッキングを成し遂げた[6]。DLLインジェクションした後は図3のようにメモリーが修正されフック関数が呼び出される。

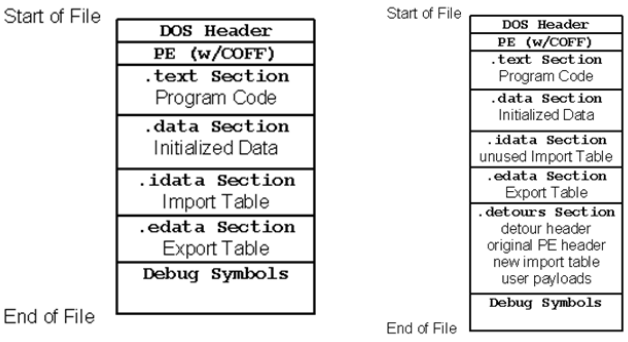

図 3. DLLインジェクション前後のプロセスの変化 [6]

2.3 ファイルアクセス分別サブシステム

本システムではファイルアクセスホワイトリストのためABAC（Attribute Based Access Control）ポリシーを使う。ABACとは、主体と主体の捜査を受ける、そして周りの環境の属性情報に基づいて作られたポリシーに従ってアクセスを許可またはブロックするアクセス制御方法である[7]。ファイルアクセス分別サブシステムは2つの異なるサブシステムから収集したファイルアクセスログに基づいてABACポリシー形式のホワイトリストを作る。主体はファイルアクセスを要請するNPE（Non-Person Entity）であるプロセスで対象はプロセスがアクセスするファイルだ。本サブシステムではまずファイルアクセスログを読み込んでプログラム名とファイルアクセス関数の名前、アクセスしたファイルの拡張子などの属性情報のタプルを作る。そして、このタプルを持っているホワイトリストから探す。タプルがホワイトリストの中にある場合は次のログを読み込む。もし探せなかったら新しいルールで追加する。

2.4 ファイルアクセス制御サブシステム

ファイルアクセス制御サブシステムはログ収集サブシステムと似てファイルアクセスをブロックするかどうかを決めるDLLファイル、プロセスモニター、ブロックログを周期的に送る自動化スクリプトが含まれるまた、ファイルアクセスをブロックした場合ユーザーに表示されるポップアップUIまで全部4つのコンポーネントで構成する。ファイルアクセス許可やブロックはプロセスモニターによりインジェクションされたDLLで決めた。図4を参考するとプロセスの仕組みが分かる。 OS上でプロセスが実行されたらプロセスモニターがこれを探知してDLLインジェクションをする。プロセスがファイルアクセスAPIを呼び出したらDLLで定義しておいたフック関数を代わりに呼び出す。フック関数ではホワイトリストからこのプロセス

が要求するアクセスを許可するか確認する。ホワイトリストが許可したアクセスの場合、実際API関数を呼び出してその結果を返す。許可したアクセスではない場合、ブロックログを記録しながらユーザーにファイルアクセスがブロックされたことを知らせるポップアップUIを出す。その後API関数呼び出し失敗の結果を返す。

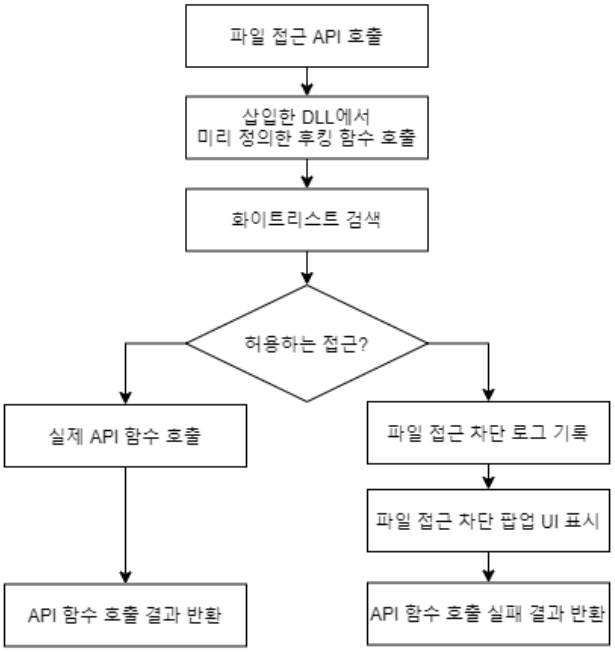

図 4. ファイルアクセス許可またはブロックの流れ

3. システム開発及びパフォーマンス分析

本システムはWindows 10の環境で開発した。プロセスモニターでプロセスの実行を探知するためWMI（Windows Management Instrumentation）を使って、ログをJSON形式に記録するためオープンソースのJSON for Modern C++[8]を使った。ログファイルを圧縮するためtarを使って、ファイルアクセス分別サブシステムにログを送るためscpを使った。ホワイトリストはsqlite3を使った。図5はプロセスモニターを通じて実行したプロセスにDLLインジェクションをした結果だ。図6はログ収集サブシステムで実行されたプロセスから収集したログで、図7は収集したファイルアクセスログに基づいて作られたホワイトリストを使って音楽再生ソフトfoobar2000でflacファイル音源の再生がブロックされたことだ。

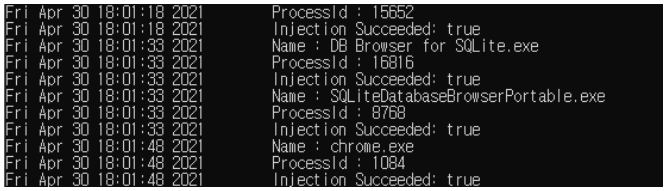

図 5. プロセスモニターのDLLインジェクション

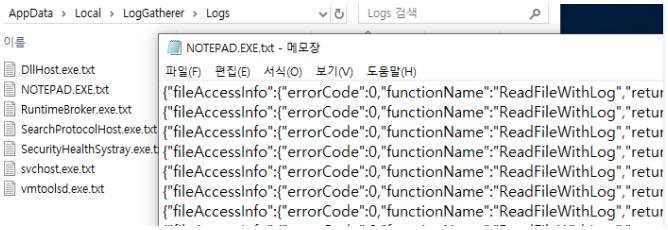

図 6. 収集したファイルアクセスログ

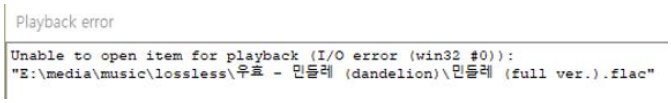

図 7. foobar2000でflac音源の再生がブロックされる

4. むすび

本論文ではファイルアクセスログを収集し、ファイルアクセスホワイトリストを作ってファイルアクセスを制御するシステムを提案する。これを通じてプロセスの認可されてないファイルアクセスを制御することでランサムウェア攻撃及び情報漏洩を防止する効果が期待できる。今後はより様々な属層情報を使用してファイルアクセスを制御することで遮断方式の正確度と安全性を高める必要があるし、ホワイトリストではなくブラックリストも含めるポリシーで発展する必要がある。また、管理者が直接判断し許可する部分も自動化する必要がある。我々はファイルアクセス分類サブシステムにマシーントレーニングアルゴリズムを適用することで先並べた問題を改善する研究を進める予定だ。

参考文献

[1] 한국인터넷진흥원, “IBM 2020 글로벌 기업 데이터 유출 현황 주요 내용 분석,” 2020.
[2] Risk Based Security, “2020 Year End Data Breach QuickView,” 2021.
[3] 한국인터넷진흥원, “2020 글로벌 정보보호 산업시장 동향조사,” 2021.
[4] 윤정무, 조제경, 류재철, “파일 I/O Interval을 이용한 랜섬웨어 공격 차단 방법론,” 정보보호학회논문지, 26(3), 645-653, 2016.
[5] 김재홍, 나중찬, "파일의 읽기/쓰기 권한 제한을 통한 암호화 랜섬웨어로부터 선택적 파일보호 연구," 한국정보처리학회 학술대회논문집, vol. 24, 234-237, 2017.
[6] “Detours,” https://github.com/microsoft/Detours.
[7] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, et al., “Guide to Attribute Based Access Control (ABAC) Definition and Considerations,” NIST, vol. 800, no. 162, 2013.
[8] “JSON for Modern C++,”https://json.nlohmann.me.