拡張子変更に注目したファイルサーバにおける ランサムウェアの暗号化攻撃からの回復手法の検討

永野 凜太郎[1]　稲村 浩[2]　石田 繁巳[2]

概要：感染した端末からアクセス可能なファイルを暗号化して使用不能にし，その復号と引き換えに金銭を要求する暗号化型ランサムウェアが企業を中心に甚大な被害を出している．既存の端末上の暗号化攻撃を受けたファイルを自動復元する手法では，各端末に対策機構を構築しており管理にコストがかかる．それに対し，データをファイルサーバだけで扱う仕組みにすれば，各端末ごとにデータを保存していた今までと異なり暗号化攻撃の対策対象はファイルサーバのみにできるため，管理コストが減少する．そこで本研究では，拡張子を特有のものに変更する既知の検体の振る舞いに限定して，ファイルサーバ上のファイルに対する暗号化攻撃のファイル操作をユーザの操作を契機に偽陽性なく短時間かつ自動で特定し，ロールバックする手法を提案する．本稿では暗号化の際にファイル拡張子を特有のものに変更する，ランサムウェアの典型的な挙動に則って実装した疑似ランサムウェアを用いて，提案システムに暗号化攻撃した．暗号化攻撃を受けたファイルを自動で復元したところ，疑似ランサムウェアが行ったファイル操作を短時間かつ自動で特定し，ロールバックできた．

1. はじめに

感染した端末からアクセス可能なファイルを暗号化することで使用不能にして，その復号と引き換えに金銭を要求する暗号化型ランサムウェアが流行している [1]．ランサムウェアには暗号化型と画面ロック型があるが，本研究では主に暗号化型ランサムウェアをランサムウェアと呼称する．ランサムウェアによるファイルの暗号化を暗号化攻撃と呼ぶ．暗号化攻撃により暗号化されたファイルを復号することは非常に難しいため，ランサムウェアは脅威である．

ランサムウェアによる暗号化攻撃は企業を中心に甚大な被害を出している [1,2]．例えば，大阪急性期・総合医療センターにて，電子カルテに関連するデータを暗号化され，サービス復旧に 60 日かかり，被害総額が十数億円以上となった事例がある [3]．このように，復旧期間が長くなることによって被害総額が増大する可能性が高くなる．企業などの組織が暗号化攻撃を受けた際，ファイルを短時間で復元し，事業を再開できれば被害額が減少する．

既存の端末上の暗号化攻撃を受けたファイルを自動復元する手法 [4] では，ユーザの正当な操作をランサムウェアによる操作と誤検知する偽陽性の低い検知と高速な自動復元をしているが，各端末に対策機構を構築すると管理にコストがかかる．端末ごとに暗号化攻撃を対策する場合，導入や更新が負担になる．企業などの組織においては情報技術の習熟の度合いにばらつきがあることが想定され，構成員自ら複雑な対策機構を管理することを期待することは難しい．十分な情報技術を持つ他社に各端末のセットアップや更新を依頼する際には多大なコストがかかる．

それに対し，データをファイルサーバだけで扱う仕組みにすれば，端末ごとにデータを保存していた今までと異なり暗号化攻撃の対策対象はファイルサーバのみに限定できるため，管理コストが減少する．すなわち，ファイルサーバで暗号化攻撃を対策することにより，ソフトウェアの更新や個別環境への対応を行う対象がサーバ 1 つに集約される．

そこで本研究では，拡張子を特有のものに変更する既知の検体の振る舞いに限定して，ファイルサーバ上のファイルに対する暗号化攻撃のファイル操作をユーザの操作を契機に偽陽性なく短時間かつ自動で特定し，ロールバックする機構を提案する．管理コストの減少が目的であるため，暗号化攻撃対策にはファイルサーバ上で得られる情報のみを使用する．ランサムウェアの中には，暗号化されていることを分かりやすくし被害者をより動揺させるため，暗号化したファイルの拡張子を特有の拡張子に変更するものが存在する [5,6]．

ファイルサーバ上でランサムウェアに対抗するため，ファ

[1] 公立はこだて未来大学大学院 システム情報科学研究科
[2] 公立はこだて未来大学 システム情報科学部

イル拡張子を特有のものに変更する暗号化攻撃から自動でファイルを復元する．暗号化の際にファイル拡張子を特有のものに変更する，ランサムウェアの典型的な挙動に則って実装した疑似ランサムウェアを用いて，提案システムに暗号化攻撃して自動で暗号化攻撃を受けたファイルを復元する．本研究では，暗号化の際にファイル拡張子を特有のものに変更する疑似ランサムウェアを実装し，評価した．

本研究の貢献は以下の通りである：

- 管理コストを低減するため，ファイルサーバ上でユーザの操作を契機に暗号化攻撃を受けたファイルを自動的に復元する手法を提案し，プロトタイプ実装を行った
- 暗号化攻撃とそれ以外のファイル操作が同時並行する状況において，暗号化攻撃からの回復が高速に行われることを示した

本論文の構成は以下の通りである．第 1 節にて背景と目的を示した．第 2 節では，関連研究を示し，既存の暗号化攻撃への対策手法に注目し，ファイルサーバ上での対策手法に有用な情報を示す．第 3 節では，拡張子変更に注目したファイルサーバにおける暗号化攻撃からの回復手法を述べる．第 4 節では，提案手法の復元速度について評価方法を説明し，その結果について議論する．最後に，第 5 節にてまとめとする．

2. 関連研究

ファイルサーバ上でランサムウェアによる暗号化攻撃の対策機構を実現するために，関連研究を確認することで，本研究の立ち位置と目的の達成に向けて有効な情報を明らかにする．

ランサムウェアによる暗号化攻撃の対策機構を構築できるのは，「クライアント端末」，「クライアント端末がマウントしているファイルサーバ」，「ファイルサーバとクライアント端末が構成するネットワーク」の 3 箇所である．この 3 箇所のうち，「クライアント端末がマウントしているファイルサーバ」で対策を行う研究は著者らの知る限り報告されていない．

本節では，クライアント端末上で暗号化攻撃の検知・停止，および暗号化されたファイルの自動復元を行った研究と，ファイルサーバとクライアント端末間の通信トラフィックを用いて暗号化攻撃を検知・停止する研究の 2 種類について述べ，ユーザの正当な操作をランサムウェアによる操作と誤検知する偽陽性を低くすることの重要性を述べる．そのうえで，これまでランサムウェアを検知する手掛かりとして用いられていた情報を述べ，ファイルサーバ上での対策機構実現に有効な情報を明らかにする．

2.1 クライアント端末上での暗号化攻撃の検知・停止，および自動復元手法

クライアント端末における既存手法では，機械学習などの方法で暗号化攻撃や脅迫文の作成などの操作が示す特徴を用いて高い精度で暗号化攻撃を検知・停止，および自動復元ができることが明らかとなっている．

文献 [7] ではハイパーバイザを用いて，ストレージへのアクセスパターンを収集することでランサムウェアの検知を実現している．文献 [8] では，暗号化攻撃の振る舞いに関する指標を使用して，暗号化攻撃を検知するカーネルドライバを作ることで，ユーザのデータが大量に失われる前に低い偽陽性でランサムウェアを停止することを実現している．しかしながら，ハイパーバイザやカーネルドライバは導入や維持更新に関わる管理コストが高いため，これらの手法を組織へ導入することは難しい．例えば，組織でハイパーバイザなどの複雑な仕組みを管理するためには高度な知識が必要となり，情報技術に習熟していない組織の構成員が各自でこれらの検知機構を導入することは困難である．つまり，既存の手法を導入するためには，他組織に端末のセットアップや更新作業を依頼する必要があり，時間と費用がかかる．さらに，これらの手法は自動でファイルを復元できず，ファイル復元にかかる時間が別途必要であり，事業の復旧に時間がかかるため被害が拡大する可能性があることが問題である．

文献 [4] では，Windows ネイティブファイルシステム向けにランサムウェアの暗号化攻撃を検知し，データの自動復元を行うアドオンドライバを開発した．IRPLogger (I/O Request Packet Logger) という低レベル I/O ファイルシステム要求を収集するためのモジュールを実装し，ユーザとランサムウェアのファイルアクセスを収集した．収集したデータを用いて教師あり学習を行い生成した分類モデルで，良性プロセスとランサムウェアプロセスを 97.7%の精度で分類できることを示した．この手法では，プロセスのファイル初回アクセス時にファイルのコピーをとり，プロセスがランサムウェアであると判断された場合はコピーを用いてファイル復元を実現している．しかしながら，この手法では検知に機械学習を使用しているため，後に再評価を行った際には検知率が大きく低下している．文献 [9] で作られたテストケースでは，分類精度が 65.7%まで低下したことが報告されている．

以上のように，クライアント端末向けの手法は「管理コストが高い」，「ランサムウェアの停止のみ行い自動で復元できていないものがある」，「機械学習で検知しているため再学習をしなければ偽陽性が上がってしまう」という 3 点が問題である．

2.2 通信トラフィックを用いた暗号化攻撃の検知・停止

通信トラフィックを用いた暗号化攻撃の検知・停止を目的に，文献 [10] ではパケットベースの通信を分析可能なメタデータ形式に変換するシステムでトラフィックデータを集め，暗号化攻撃特有の動作を検知し，クライアント端末との通信を遮断するシステムを開発した．この手法では，ファイルサーバが暗号化攻撃された際でも攻撃を停止可能であることが確認されている．しかしながら，通信トラフィックを用いた暗号化攻撃の検知・停止では，ファイルの自動復元まで行うことができていない．

2.3 ファイルサーバ上で暗号化攻撃を受けたファイルを自動復元する際の課題

ファイルサーバ上で暗号化攻撃を受けたファイルの復元を試みる際，管理コストの問題は解決できるが，「短時間での復元」と「偽陽性」の 2 つが大きな課題となる．

企業などの組織で実運用する際，暗号化攻撃を受けても素早く事業を復旧できることが経済的な被害を抑えることにつながる．復旧期間の長期化による被害総額の増大を抑止するために，暗号化攻撃の停止だけでなく自動で復元も行うことで高速なファイル復元が必要である．

これまで見てきた既存の手法では，いずれも偽陽性が重要な尺度となっている．暗号化攻撃の特定の際ユーザの正当な操作をランサムウェアによる操作と誤検知することを偽陽性と呼ぶ．暗号化攻撃の検知・停止，暗号化攻撃を受けたファイルの自動復元は，偽陽性が起こってしまうと，ユーザの正当な変更を停止したり，ロールバックしたりしてしまう．

2.4 暗号化攻撃の特性

前節で述べた 2 つの課題の解決に有効な情報として，ランサムウェアの拡張子変更が挙げられる．暗号化攻撃の際，暗号化するファイルの拡張子を変更するランサムウェアが複数報告されている [5,6]．ファイルサーバ上でファイル名の取得は可能であることから，ファイル拡張子はファイルサーバ側のみで取得できる情報である．

文献 [8,11] において，以下 4 つの典型的な暗号化攻撃の挙動が述べられている．

(1) 暗号化対象となるファイルのデータを読み込み，暗号化を行ったデータを同一のファイルに上書きする
(2) 暗号化対象となるファイルのデータを読み込み，暗号化を行ったデータを別ファイルに保存した後，元のファイルを削除する
(3) 暗号化対象となるファイルのデータを読み込み，暗号化を行ったデータを別ファイルに保存した後，元のファイルを別のデータで上書きし破壊する
(4) ユーザのドキュメントディレクトリから一時ディレクトリなどにファイルを移動させた後ファイルを読み込み，暗号化した内容を書き込んで元の位置に戻す．

ランサムウェアがファイル拡張子を特有の拡張子に変更する場合，ファイル拡張子の変更操作から，ランサムウェアの暗号化攻撃によるファイル操作を特定できる可能性がある．ファイル操作のうち，操作対象ファイルのファイル名に特有の拡張子を含んでいるものは暗号化攻撃に関連する操作だと分かる．例えば，このうち，2 の挙動を示すものは暗号化データを保存するファイルを作成する際，拡張子を変更したファイルを作成していると推測される．暗号化攻撃に関する操作を特定できれば，暗号化攻撃による操作を元に戻す操作を実行することで暗号化されたファイルを復元できる．

3. 提案手法

本研究では，拡張子を特有のものに変更する既知の検体の振る舞いに限定して，ファイルサーバ上のファイルに対する暗号化攻撃のファイル操作を偽陽性なく短時間かつ自動で特定し，ロールバックする手法を提案する．ファイルサーバ上でランサムウェアの検知を行うために暗号化攻撃に伴うファイル拡張子の変更操作を利用する．ファイルサーバ上で書き込み処理が起きるたびに書き込み前のファイル内容をバックアップした上で，クライアント端末で実行されたファイル操作を時系列順に並べた「ファイル操作列」から暗号化される前のバックアップデータを特定する．本研究では拡張子を判断基準として，ランサムウェアの操作を確定できるため，偽陽性をなくせる利点がある．

ファイルを復元する際に機械学習などの処理時間のかかる仕組みを使わず，高速に判別できる拡張子を用いる．ランサムウェアのファイル削除操作・ファイル作成操作を拡張子を含むファイル名から特定し，短時間で暗号化攻撃によるファイル操作のロールバックを試みる．

自動ファイル復元機構では偽陽性があるとユーザの正当な変更をロールバックしてしまうため，偽陽性を無くすことは重要な指標となる．ランサムウェアが脅迫効果をより高めるため暗号化したファイルの拡張子を特有のものに変更するという特徴を用いて，その特有の拡張子を基準にした検知を行うことで偽陽性を減らすことが可能になる．

3.1 提案システムの概要

ランサムウェアによるファイル破壊から，提案システムによるファイル復元までの流れを**図 1** に示す．

提案手法は，ファイル操作列を出力する「ファイル操作列出力機能」，ファイル操作列から疑似ランサムウェアの暗号化攻撃における特有の拡張子のファイル作成操作・ファイル削除操作を特定する「暗号化攻撃特定機能」，特定された暗号化攻撃のファイル作成操作・ファイル削除操作を

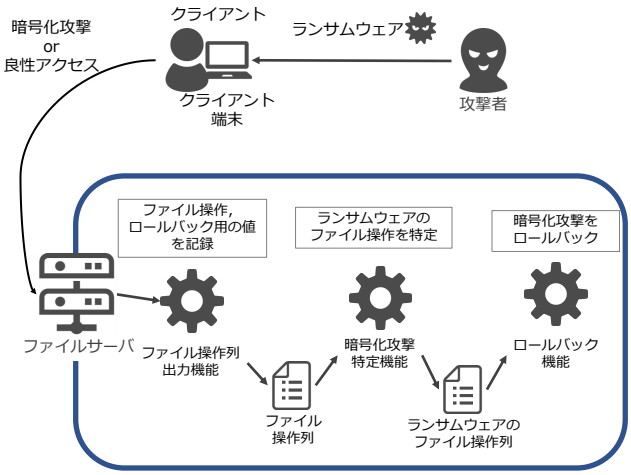

図 1　ファイル破壊から復元までの流れ

ロールバックする「ロールバック機能」の 3 つの要素で構成される．ファイル操作列出力機能が出力したファイル操作列を暗号化攻撃特定機能に入力し，特定した暗号化攻撃のファイル操作をロールバック機能に入力して自動ファイル復元を実現する．

3.2　プロトタイプ実装に用いるファイルサーバ

本研究では，オープンソースのファイルサーバである UNFS3 [12] を拡張する形で実装し，提案システムを開発した．UNFS3 は，Network File System ver3.0(NFS ver3.0) [13] プロトコルを実現したファイルサーバ実装であり，ユーザスペースで動作する．提案システムは，NFS ver3.0 プロトコルで定義される手続きを実現したものを記録，ロールバックする．提案システムを構築する際に使用した UNFS3 のリリースバージョンは 0.9.22 である．

3.3　ファイル操作関数

提案システムを構成するにあたり，ファイルサーバ上のファイルを操作した際に実行される関数に注目する．本研究では，この関数のことを「ファイル操作関数」と呼ぶ．NFS ファイルサーバをマウントしたクライアントでファイル操作を行うと，ファイルサーバ上では ONC RPC [14,15] によってファイル操作関数が実行される．ONC RPC とは，サンマイクロシステムズ社が開発した，ネットワークを経由して関数を別の PC で実行する仕組みである．この仕組みに基づき，クライアントプログラムが関数を呼んだとき，クライアント端末からファイルサーバにリクエストが送信される．ファイルサーバはリクエストを受け取り，対応する処理を実行し，クライアント端末に処理結果を返却する．このようにファイルサーバは，クライアント端末がファイルを操作する度に，対応するファイル操作関数を実行している．

実行されるファイル操作関数は，OS のシステムコールとほぼ 1 対 1 で対応するものである．例えば，書き込みの時は write，読み出しの時は read というように対応する関数が呼ばれている．NFS ver3.0 においては，文献 [13] の中で述べられている Server Procedures のうちの一部がファイル操作関数に当たる．NFS ver3.0 の Server Procedures を**表 1** に示す．

表 1　NFS ver3.0 - Server Procedures

Server Procedures	動作
NULL	何もしない (レスポンステスト用操作)
GETATTR	ファイル属性を取得する
SETATTR	ファイル属性を指定する
LOOKUP	ファイル名を検索する
ACCESS	アクセス権限を確認する
READLINK	シンボリックリンクを読みだす
READ	ファイルを読みだす
WRITE	ファイルへの書き込み
CREATE	ファイルを作成する
MKDIR	ディレクトリを作成する
SYMLINK	シンボリックリンクを作成する
MKNOD	名前付きパイプ，デバイスファイルの作成
REMOVE	ファイルを削除する
RMDIR	ディレクトリを削除する
RENAME	ファイル名を変更する
LINK	ハードリンクを作成する
READDIR	ディレクトリから読みだす
READDIRPLUS	拡張した情報をディレクトリから読みだす
FSSTAT	動的なファイルシステム情報を取得する
FSINFO	静的なファイルシステム情報を取得する
PATHCONF	ファイル名等の POSIX 基準適合状態を取得する
COMMIT	サーバにキャッシュしたデータをストレージに反映する

実行されたファイル操作関数は，クライアント端末に依存することなく，ファイルサーバ側で記録することができる．そのため，ファイル操作関数はファイルサーバ側のみで自動ファイル復元を実現することを目指している提案システムに使用することができる．

3.4　ファイル操作列出力機能

ファイル操作列出力機能は，ファイル操作列を出力する機能である．この機能はファイル操作関数が実行されるたびにファイル操作列の 1 レコードを記録する．1 レコードに含まれるデータは，実行タイムスタンプ，実行されたファイル操作関数，ファイル操作関数の引数，操作対象の変更前のデータである．ファイル操作列出力機能による処理の全体像を**図 2** に示す．

ファイル操作列出力機能の実装について実装主要部分を**図 3** に示す．ここで，`file_op_proc` はファイル操作関数，`ts` はファイル操作の実行タイムスタンプ，`op_target` は `file_op_proc(args)` の呼び出しによって変更される対象

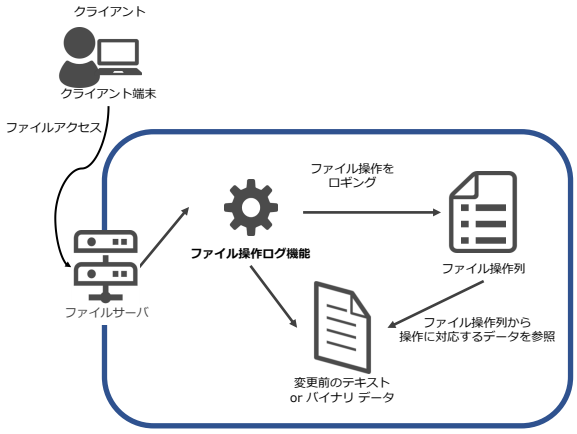

図 2　ファイル操作列出力機能

```
func file_op_proc (args){
    time ts = getTS();
    string name = get_proc_name();
    Data op_target = get_target_data();

    if (name == "WRITE"){
        fp = fopen(backup_path);
        write(fp, op_target);
        fclose(fp);

        printRecord(ts, name, args, backup_path);
    }
     // 本来の処理
}
```

図 3　ファイル操作列出力機能の実装主要部分

データの変更前の値，is_logging はファイル操作列出力機能を起動，停止するための真理値である．各ファイル操作関数の初めに，このコード例で示した処理が行われている．

ファイル操作列出力機能にかかわる処理の流れを，コード例に沿って述べる．クライアント端末からファイル操作が行われた際，対応するファイル操作関数が実行される．ファイル操作関数が実行されると，実行タイムスタンプ，ファイル操作関数名，ファイル操作関数の対象データの変更前の値を取得する．ファイルへの書き込み処理の場合は，更新前のファイル内容はファイル操作列に直接書き込むにはデータサイズが大きく，のちにロールバック機能や暗号化攻撃特定機能で扱いづらくなるためバックアップデータを保管用のファイルパスに書き込む．そして，最後にタイムスタンプ，ファイル操作関数名，引数などを用いてファイル操作列の 1 レコードを出力して，ファイル操作関数本来の処理に戻るという流れになる．

この機能により，記録されたファイル操作列は図 4 のようになる．時系列順に，実行タイムスタンプ，実行時間，実行されたファイル操作関数，ファイル操作関数の実行対象となるファイルやディレクトリの絶対パス，ファイル操作関数の変更前情報を出力している．しかし，WRITE が呼ばれた際は，この形に当てはまらず，ファイル操作列には変更前の情報を保存したファイルパスを記録している．

```
Tue Jan 17 19:20:58 2023 NFSPROC3_GETATTR
/home/inamura/PublicUnfs3/dir3/c
Tue Jan 17 19:20:58 2023 NFSPROC3_ACCESS
Tue Jan 17 19:20:58 2023 NFSPROC3_WRITE
/home/inamura/PublicUnfs3/dir3/c
/home/nagano/unfs3_log/2023-01-17-19:20:58.287009
-before_write
Tue Jan 17 19:21:30 2023 NFSPROC3_LOOKUP
/home/inamura/PublicUnfs3/dir3/log_stop
```

図 4　実際のファイル操作列

表 2　ファイル操作関数の意味とロールバック手順

ファイル操作関数	操作	ロールバック操作
WRITE	ファイルへの書き込み	書き込みが行われる前のバックアップを保存しておきファイル内容を復元する
SETATTR	ファイル属性の指定	変更前のファイル属性を記録しておき，変更する．
MKDIR	ディレクトリの作成	作成したディレクトリの名前を記録しておき，作成したディレクトリを削除する
SYMLINK	シンボリックリンクの作成	作成したシンボリックリンクの名前を記録しておき，作成したシンボリックリンクを削除する
MKNOD	特殊ファイル (名前付きパイプ，デバイスファイル) の作成	作成した特殊ファイルの名前を記録しておき，作成した特殊ファイルを削除する
REMOVE	ファイルの削除	削除したファイルのバックアップを保存しておきバックアップをもとにファイルを再生成する
RMDIR	ディレクトリの削除	削除されたディレクトリの名前を記録しておき，ディレクトリを再生成する．
RENAME	ファイル名の変更	変更される前のファイル名を保存しておき，変更されたファイル名を変更される前のファイル名に再変更する
LINK	ハードリンクの作成	生成したハードリンクの名前を記録しておき，ハードリンクを削除する

3.5　暗号化攻撃特定機能

暗号化攻撃特定機能は，ランサムウェア特有のファイル拡張子を用いて，暗号化攻撃に伴うファイル削除操作とファイル作成操作を特定する機能である．この機能は，本システムの管理者が任意のタイミングで実行する．ファイル削除操作，ファイル作成操作を行う際には対応するファイル操作関数に操作対象ファイルのパス名が引数として渡される．それらのパス名はファイル操作列に出力することができる．それらのパス名がランサムウェア特有の拡張子を含んでいれば，そのファイル操作はランサムウェアのファイル操作列である．最終的に，ロールバック機能に入力するために特定したファイル操作を出力する．

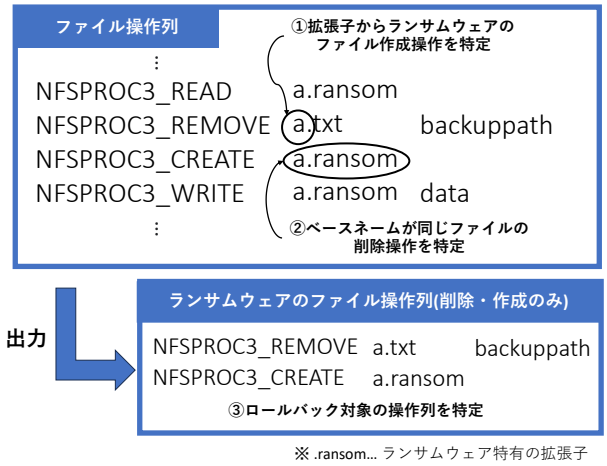

図 5　暗号化攻撃特定機能の操作

暗号化攻撃特定機能の操作を図 5 に示した．まず，暗号化攻撃の特定を行っている間に新たなファイル操作が実行されないように，提案システムにおいてクライアントからの要求の受け付けを一時的に停止する．その後，ファイル操作列を読み出し，リスト構造に格納する．そして，リストの各要素ごとにファイル作成操作の作成対象のファイル名を探索し，ランサムウェア特有の拡張子を含んでいた場合には，ランサムウェアの操作として特定する．その後，特定したファイル作成操作により作成されたファイルとベースネームが一致するファイルの削除操作をランサムウェアの操作として特定する．ここで，ベースネームとはファイル名のうち，拡張子以外の部分のことを示す．最後に特定した操作列を出力する．このように特有の拡張子を用いてファイル操作列から，ランサムウェアのファイル操作列のみを抜き出す．

3.6　ロールバック機能

ロールバック機能は指定したファイル操作を元に戻すものである．前節の暗号化攻撃特定機能により得られた暗号化攻撃のファイル操作列に含まれるファイル操作をロールバックする．ファイル操作列出力機能により保存されたファイル操作列を用いて，行われたファイル操作と相反する操作を行う．例えば，ファイル作成操作を元に戻したい場合はファイル削除操作を行う．ファイルへの書き込み操作を元に戻したい場合は，書き込みが起こる前の状態のバックアップデータからファイルを復元する．実行された操作とロールバック操作の対応を表 2 に示す．

ロールバック機能の流れを述べる．まず，暗号化攻撃の特定を行っている間に新たなファイル操作が実行されないように，提案システムにおいてクライアントからの要求の受け付けを一時的に停止する．次に，ファイル操作列を読み出し，リスト構造に格納する．そして，ファイル操作列をタイムスタンプの逆順にソートする．最後にファイル操作列のそれぞれのレコードに対し，実行されたファイル操作関数と逆の操作を行う手続きを実行する．

4.　評価

第 3 節で示した提案システム上のファイルに対して，独自に実装した疑似ランサムウェアの暗号化攻撃から復元機構を実行した後に自動で選択的にファイルを復元した実行時間を計測した．この評価において，暗号化攻撃と並行して他のファイル操作が行われた場合でも暗号化攻撃のファイル操作のみを選択的にロールバックできることを示す．

4.1　疑似ランサムウェア

本研究では，評価の際に予想外の挙動を示さず評価実験がやりやすいため，独自に実装した疑似ランサムウェアを評価に用いている．疑似ランサムウェアは文献 [8,11] で述べられている典型的なランサムウェアの挙動を参考に実装されている．

実験で用いた疑似ランサムウェアは次に示すような流れで暗号化攻撃を行うものである．

(1) 暗号化対象ファイルの読み出し
(2) 暗号化データを書き込むファイル作成
(3) 暗号化データの書き込みと拡張子変更
(4) 暗号化対象ファイルの削除

Go 言語で実装された疑似ランサムウェアのコードにおける主要部を図 6 に示す．疑似ランサムウェアの流れを，疑似コードに沿って述べる．まず，暗号化の鍵を作成する．次に，暗号化対象のディレクトリを読み出し，ディレクトリにあるそれぞれのファイルに対して暗号化処理を行う．本来，暗号化対象のディレクトリは/Downloads などである．その後読み出し元のファイルを削除し，CFB モードで AES 暗号化したテキストを作成する．最後に元ファイルのファイル名において拡張子を特有のものに変更したファイルを作成する．作成したファイルに暗号化したデータを書き込む．この処理をディレクトリ内のそれぞれのファイルに対して繰り返す．

4.2　暗号化攻撃と並行して行われるファイル操作

提案システムを実際に使用する際には，暗号化攻撃と並行して他にファイルが操作されることが予想される．実環境では並行してユーザの作業が行われているはずである．

この際，並行して行われる操作は，複雑なものであればあるほどファイル操作の特定が難しくなる．実環境で行われる操作は Word，Excel，その他のツールによる操作と思われるが，これらのアプリケーションソフトウェアによるファイル操作は散発的でクライアント当り少量と考えられる．大量のファイル操作を行うものの例としてはプログラムのビルドがあげられる．本実験では，疑似ランサムウェ

```
// Generate a key
key = make([]byte, 32)
rand.Read(key)
for folderPath = range foldersPath {
  files = ReadDir(folderPath)
  for _, file := range files {
    filePath := filepath.Join(folderPath, file.Name())
    if !HasSuffix(file.Name(), ".ransom") {
      // Encrypt the file
      plainText = ReadFile(filePath)

      // Delete the original file
      Remove(filePath)

      // encrypt plain text
      encrypt(key, plainText, cipherText, "CFB")

      // Write the encrypted file
      destinationPath := filepath.Join(folderPath,
                    changeExt(file.Name()+".ransom"))
      err := WriteFile(destinationPath, cipherText)
    }
  }
}
```

図 6　疑似ランサムウェアの実装主要部分

アの暗号化攻撃と並行してファイルの読み出しや書き込みが集中して行われるソフトウェアのソースコードからのビルドを行い，複数のファイル操作の中から暗号化攻撃を受けたファイルのみを自動的に復元できることを評価する．対象のソフトウェアには，nginx のリリースバージョン 1.23.3 を用いた．

4.3　暗号化攻撃の選択的ロールバック評価

ユーザの操作を契機に，暗号化攻撃によるファイル操作を暗号化攻撃特定機能を用いて特定し，選択的にロールバックすることで暗号化攻撃を受けたファイルを復元し，復元の実行時間を測定する．この処理の実行時間がファイルの復元処理におけるダウンタイムである．攻撃無効化処理の実行時間は攻撃対象ファイルの数やサイズで変化することが想定されるため，ここでは攻撃対象ファイルの数を変化させ実行時間の増分を評価する．今回は暗号化対象のファイル個数を 1〜20 個に変更し，設置した個数ごとに計 20 回実験を行い，ファイル復元にかかる時間を計測した．

4.3.1　評価環境

実験環境を**図 7** に示した．サーバ上でファイルサーバを起動した上で，公開したディレクトリを自己マウントし，マウントしたディレクトリ内部のディレクトリを暗号化攻撃している．暗号化攻撃の際は，共有ディレクトリに nginx のディレクトリと暗号化対象ディレクトリを置き，nginx のビルドを実行している．

文献 [16] を参考に，暗号化対象となるファイルのファイルサイズは，実環境で Windows ユーザが使用するファイルのファイルサイズの中央値である 80kB とした．この文献では，2000 年の上半期に一ヶ月間，犯罪研究所の取締役とその上司，州の警察責任者，理系の大学院生などの多様な人物が，多様な業務でファイルを使用した際のデータを収集している．この実験の中で，Windows NT ユーザが使用したファイルの中央値が 80KB ほどであり，実際に使用されたファイル全体でも約 20%が 80kB 前後と示されている．

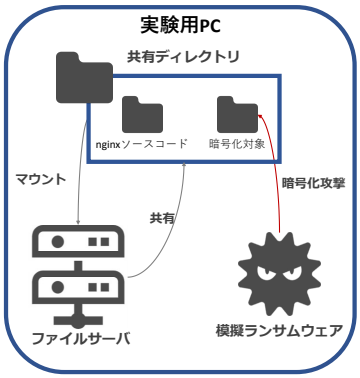

図 7　実験環境

製品名	Odyssey Blue J4105
CPU	Intel®Celeron®J4105
メモリ	8GB
OS	Debian 10 (buster)

表 3　実験に用いた小型 PC のスペック

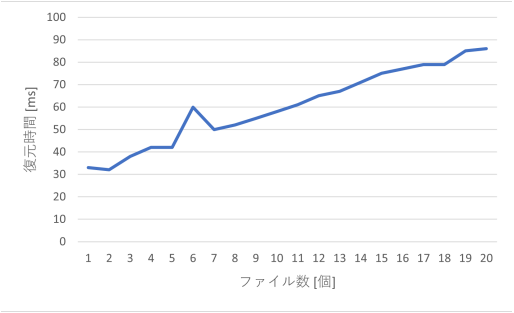

図 8　復元時間とファイル数の関係

実験で用いた機材は**表 4.3.1** に示した．

4.3.2　復元時間の評価結果

ファイル操作を並行して行いつつ，疑似ランサムウェアによる暗号化攻撃を実行し，疑似ランサムウェアによるファイル削除・作成操作を特定してファイルを復元した結果を**図 8** に示す．この図は縦軸がファイルの復元にかかる実行時間を，横軸が復元したファイル数を表している．この際，並行して実行したファイル操作は 4.2 で述べたように nginx のビルドである．結果として，最もファイル数の多い 20 個のファイルに対して暗号化攻撃を行った場合でも，86ms で自動でファイルを復元できた．グラフがほぼ一直線になっていることから復旧時間は失われたファイル

量に比例し，1 ファイルあたり 3ms 弱ほどで復元できていることが分かる．

復元にかかる時間は比較的短く，事業の継続には影響が少ないことが期待できる．本実験は企業などの組織がランサムウェアに侵入されファイルサーバに暗号化攻撃をうけた想定であり，事業の復旧の観点から復元にかかる時間が重要となる．提案手法は暗号化攻撃の際，元ファイルを削除して，拡張子を変更したファイル名のファイルを作成するランサムウェアから短時間のうちに自動でファイルを復元できるということが分かった．

5. おわりに

本研究では，管理コストの減少を目的として，ファイルサーバ上のファイルに対するファイル拡張子を特有のものに変更する暗号化攻撃のファイル操作を，ユーザの操作を契機に偽陽性なく自動かつ短時間で特定し，ロールバックする機構を提案した．提案システム上のファイルに対して，独自に実装した疑似ランサムウェアによる暗号化攻撃から自動でファイルを復元した結果を評価した．この評価において，暗号化攻撃と並行して他のファイル操作が起きた場合でも提案手法は有効なことを示した．

参考文献

[1] 独立行政法人情報処理推進機構（IPA）：情報セキュリティ白書 2022，独立行政法人情報処理推進機構（IPA）(2022).

[2] 警察庁：令和 3 年におけるサイバー空間をめぐる脅威の情勢等について，警察庁 (2022).

[3] 地方独立行政法人大阪府立病院機構 大阪急性期・総合医療センター情報セキュリティインシデント調査委員会：調査報告書, https://www.gh.opho.jp/pdf/report_v01.pdf (参照 2023-5-23).

[4] Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S. and Maggi, F.: ShieldFS: A Self-healing, Ransomware-aware Filesystem, *Proceedings of the 32nd Annual Computer Security Applications Conference*, ACM.

[5] McAfee: 脅威増大中の Babuk ランサムウェアを分析（技術レポート）, https://ascii.jp/elem/000/004/064/4064324/ (参照 2023-5-22).

[6] GROUP, Y. T.: Conti Ransomware source code: a well-designed COTS ransomware, https://yoroi.company/research/conti-ransomware-source-code-a-well-designed-cots-ransomware/ (参照 2023-5-22).

[7] 程田凌羽，今泉大慈郎，平野学，小林良太郎ほか：ストレージアクセスパターンに着目した機械学習及び深層学習によるランサムウェアの検知手法の検討，研究報告マルチメディア通信と分散処理 (DPS)， Vol. 2020, No. 19, pp. 1–8 (2020).

[8] Scaife, N., Carter, H., Traynor, P. and Butler, K. R. B.: CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data, *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 303–312 (2016).

[9] Gupta, A., Prakash, A. and Scaife, N.: Prognosis Negative: Evaluating Real-Time Behavioral Ransomware Detectors, *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–368 (online), DOI: 10.1109/EuroSP51992.2021.00032 (2021).

[10] Morato, D., Berrueta, E., Magaña, E. and Izal, M.: Ransomware early detection by the analysis of file sharing traffic, *Journal of Network and Computer Applications*, Vol. 124, pp. 14–32 (online), DOI: https://doi.org/10.1016/j.jnca.2018.09.013 (2018).

[11] 荻原拓海，小林良太郎，加藤雅彦ほか：デコイファイルを用いた暗号化型ランサムウェアの検知とプロセス特定に関する検討，研究報告マルチメディア通信と分散処理 (DPS)， Vol. 2021, No. 50, pp. 1–8 (2021).

[12] Schmidt, P.: UNFS3 Homepage, https://unfs3.sourceforge.net/ (参照 2022-12-22).

[13] Staubach, P., Pawlowski, B. and Callaghan, B.: NFS Version 3 Protocol Specification, RFC 1813 (1995).

[14] Sun Microsystems, I.: RPC: Remote Procedure Call Protocol specification: Version 2, RFC 1057 (1988).

[15] Srinivasan, R.: XDR: External Data Representation Standard, RFC 1832 (1995).

[16] Roselli, D. S., Lorch, J. R., Anderson, T. E. et al.: A Comparison of File System Workloads., *USENIX annual technical conference, general track*, pp. 41–54 (2000).