Vol.2022-CSEC-98 No.14
Vol.2022-SPT-48 No.14
2022/7/19

ランサムウェアに対する CPU 命令実行抑止機構の提案と評価

榎本 秀平[1]　葛野 弘樹[2]　山田 浩史[1]　白石 善明[2]　森井 昌克[2]

概要：ランサムウェアによる意図しないファイル暗号化の被害は増加の一途を辿っており，オペレーティングシステム (OS) の種類や，システムの構成環境を問わず，様々なランサムウェアファミリが確認されている．それらのランサムウェアから情報資産を保護するためには，アンチウイルスソフトウェアの導入が有効である．しかし，既存のアンチウイルスソフトウェアはシグネチャや既知のランサムウェアの振舞いを用いたパターンベースでの検出と防御を行うため，新たなランサムウェアファミリに対し，既存のパターンデータでは検出が困難な場合がある．本研究では，既存のアンチウイルスソフトウェアに新たなパターンデータが配信されるまでの間，未知のランサムウェアによる攻撃を緩和させる新たな手法を提案する．提案手法では，ランサムウェアの実行時に AES-NI 命令セットが利用される点に着目した．AES-NI 命令セットが含まれる暗号化処理の実行時に OS にてスキップさせることで，暗号化を透過的に無効化する．提案手法を Linux 5.7.15 に実装し，評価実験を行った．評価結果より，既存のランサムウェアによる攻撃を緩和可能であること，ならびに多くの環境において低オーバヘッドを実現可能であることを確認した．

キーワード：ランサムウェア，オペレーティングシステム，システムセキュリティ

1. はじめに

ユーザの意図に反して不正に動作するソフトウェア（マルウェア）の一種としてランサムウェアがある．ランサムウェアは標的とするホストのユーザ操作や情報資産の閲覧を不可能にし，これを復旧する代償としてユーザに対して身代金の支払いを要求する．ユーザ操作を不可能にする方法については，画面のロックのみを行う種類 [1] と，画面ロックに加えてオペレーティングシステム (OS) のファイルシステムにおいてユーザが所有するファイルの暗号化を行う種類 [2,3] が存在し，近年被害が拡大するランサムウェアは後者に分類される．また，標的とする環境は多岐に渡り，PC 上の Windows を対象とする種類 [1–3] をはじめ，ハイパーバイザや IoT 機器上の Linux ホストを対象とする種類 [4,5] などが確認されている．

既存のランサムウェアからファイルを保護するために，アンチウイルスソフトウェア（以降，アンチウイルス）の導入が有効である．アンチウイルスは，既存のマルウェアの表層的な特徴情報（シグネチャ）や，動作時の振る舞いを用いてマルウェア検出を行う．アンチウイルスはいずれもあらかじめ定義された既存のパターンデータ（以降，パターン）を用いた検出手法であり，実行ファイルのスキャンやユーザプロセス（以降，プロセス）を監視し，パターンに一致したファイルの削除やプロセスの停止を実施する．

アンチウイルスでは，新たなランサムウェアの出現に対し，既存パターンでは検出が困難な場合がある．そのため，アンチウイルスはベンダから配信される新たなパターンを適用する必要がある [6]．適用には以下の課題が存在する．

- **ランサムウェア検出にかかる時間の課題**
  アンチウイルスベンダがアンチウイルスに対してパターンを配信し，ランサムウェアを検出可能とするまでには一定の時間を要する．例として，SilentCrypt ランサムウェアファミリにおいては，VirusTotal 上のアンチウイルスエンジンにより，陽性と判定されるまでに 5 日程度の時間を要したことが報告されている [7]．そのため，パターン配信までに新たなランサムウェアファミリが拡散された場合，対応は困難といえる．また，新たなパターンの適用により，ランサムウェアが停止された場合においても，停止までの間に暗号化されたファイルの復号は困難である．

ランサムウェア対策の研究として，既存研究では，ランサムウェアによる暗号化前のファイル探索の挙動や，OS

[1] 東京農工大学
Tokyo University of Agriculture and Technology
[2] 神戸大学 大学院工学研究科
Graduate School of Engineering, Kobe University

により提供される暗号化 API の呼出しに着目した防御手法 [8, 9] が提案されている．しかし，これらの手法を回避するランサムウェアが報告されている．例として，ランサムウェアファミリの一種である LockBit [5] では，高速な暗号化のために OS による暗号化 API を利用せず，ランサムウェア内部にて暗号化実装を備えている．

本研究では，アンチウイルスベンダによるパターン配信までの間，ランサムウェアによる攻撃を緩和させるための新たな手法である CryptoWarp を提案する．CryptoWarp は，x86 が持つ AES-NI 命令 [10] に着目し，実行時に AES-NI を含む暗号化処理を透過的にスキップすることで，不正なファイル暗号化を無効化する．これにより，近年の多くのランサムウェアが利用する AES-NI 命令によるファイル暗号化の防止を実現する．CryptoWarp は OS カーネル内で動作するソフトウェアコンポーネントであり，幅広い環境に対し適用可能である．本研究の貢献を以下に示す．

(1) ランサムウェアによるファイル暗号化を緩和させる機構として CryptoWarp を提案した．CryptoWarp は既知および未知のランサムウェアに対し，暗号化を開始した段階からの早期的な緩和を実現する．
(2) 暗号化処理を透過的に禁止するための設計と実装を示した．CryptoWarp の実現にあたり，要求される既存の OS の実装変更は僅かである．アンチウイルスや既存研究の提案手法に対し，補完的に機能する．
(3) CryptoWarp を Linux 5.7.15 に実装し，セキュリティとパフォーマンスに関する評価を実施した．セキュリティ評価として，LockBit の実装を模した PoC を実行し，防御可能であることを実証した．パフォーマンス評価として，UnixBench [11] を実行し，26.84 % 程度のオーバーヘッドで実行可能であることを確認した．

2. 背景

2.1 ランサムウェアによる攻撃

ランサムウェアは，感染先ホスト上のユーザ操作を不可能にし，復旧の代償としてユーザから身代金を得ることを目的としている．操作を不可能にする手法としては，オペレーティングシステム (OS) におけるファイルシステム内のファイル暗号化が広く取り入れられている．また，ランサムウェアが標的とする環境は多岐にわたる．例として CryptoLocker [2] や WannaCry [3] では，PC 上の Windows を標的としている．また，ハイパーバイザの VMWare ESXi を標的とする LockBit [5] に加え，IoT 機器上の Linux を標的とする PaperW8 [4] が確認されている．

2.1.1 ファイル暗号化の手順

ランサムウェアが暗号化を行う際の流れを図 1 に示す．はじめに，ランサムウェアは C&C サーバにて生成された公開鍵暗号方式の鍵ペアのうち，公開鍵を受け取る．次に，感染ホストで共通鍵暗号方式の鍵を生成し，標的とするファイルを共通鍵で暗号化した上で，共通鍵を公開鍵で暗号化する．共通鍵暗号方式のアルゴリズムには Advanced Encryption Standard (AES) が用いられる．AES は固定長の鍵と平文から暗号文を生成するブロック暗号であり，ラウンドと呼ばれる計算操作を所定回数繰り返すことで暗号文生成を行う．x86 では，AES による暗号化および復号を高速化するための命令セットとして，AES-NI [10] が実装されている．例として，各ラウンド操作の計算処理である *SubBytes*, *ShifRows*, *MixColumns*, *AddRoundKey* を演算するための命令として aesenc，最終ラウンドの計算処理である *SubBytes*, *ShifRows*, *AddRoundKey* の命令として aesenclast がある．

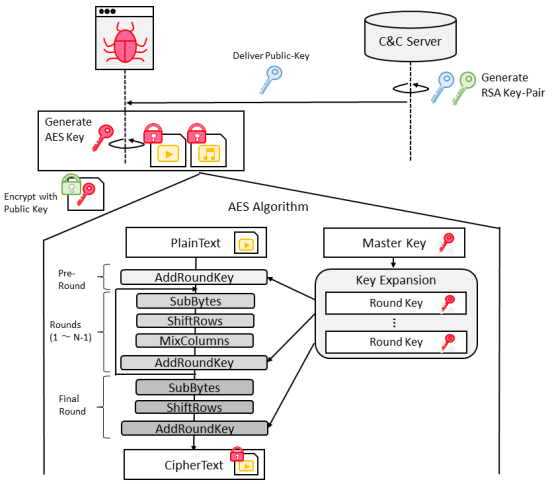

図 1 ランサムウェアにおけるファイル暗号化の流れ

復号では，支払いが確認された段階で C&C サーバが秘密鍵の配信を行い，ランサムウェアは秘密鍵を用いて共通鍵を復号，共通鍵を用いてファイルの復号を行う．

2.2 アンチウイルスによる検出

既存のアンチウイルスでは，シグネチャや振る舞い等の事前に定義されたパターンを用いてマルウェアを検出する [5]．シグネチャとしては，実行ファイルのハッシュ値等の表層情報や，実行ファイル内のセクション数等のメタデータが該当する．アンチウイルスはファイルシステム内の定期的なスキャンにより，シグネチャに一致するファイルを検索し，一致したファイルの削除を行う．マルウェアかの判定に用いる振る舞いとして，実行時に動的に得られる情報が該当する．例として，ネットワーク通信先となるドメインがある．プロセスを監視し，マルウェアの振る舞いと類似するプロセスや，閾値を超過するプロセスを検出した場合には，停止や実行ファイルの削除を行う．

3. 脅威モデル

本研究の脅威モデルとして，攻撃者は盗取されたユーザの認証情報を用いて標的とするホストにアクセスし，ランサムウェアを用いてファイルの暗号化を行うことを想定す

る．このため，OS におけるユーザ空間を一般ユーザ権限で操作可能であり，ランサムウェアをはじめとする任意のアプリケーションのインストールや，プロセスの生成が可能である．また，OS にはアンチウイルスがインストールされており，ランサムウェアによるアンチウイルスの停止やパターンの改竄等は本研究における対象外の攻撃とする．

4. 提案手法

本研究では，ランサムウェアによる不正なファイル暗号化を緩和するための新たな手法である CryptoWarp を提案する．提案手法が達成する要件を以下に示す．

要件 1：ランサムウェア以外のアプリケーションを低オーバーヘッドに実行可能とする．

要件 2：幅広い環境に対して適用可能とする．

要件 1 を満たすため，CryptoWarp は不正なファイル暗号化の緩和機能が OS に与える性能劣化を最小とするように設計し，CryptoWarp 未適用時の OS の実行状態に近い，アプリケーションの実行性能を達成する．

要件 2 を満たすため，CryptoWarp は特定のハードウェア機能に依存しないようなソフトウェアコンポーネントとして設計を行い，PC やクラウドサーバといった幅広い環境に対する適用性を達成する．

4.1 アプローチ

攻撃者は一般ユーザ権限によりランサムウェアをインストール後，ユーザプロセスとして実行し，ファイルシステム内における任意のファイルの暗号化を試みる．

CryptoWarp では，ランサムウェアが実行時に暗号化アルゴリズム用の専用命令セットを発行する点に着目する．命令セットの発行を OS にてトラップし，暗号化処理に関する命令列をスキップ（以降，暗号化スキップ）することで，暗号化をアプリケーションから透過的に無効化する．

CryptoWarp による暗号化スキップの概観を図 2 に示す．CryptoWarp が適用された OS においてはランサムウェアが暗号化を実施したファイルコンテンツは平文のコンテンツと同一となる．

4.2 デザインチャレンジ

4.1 節にて示したアプローチを満たすにあたり，要求されるデザインチャレンジを以下に示す．

- **特定のランサムウェアの実装に依存することなく，幅広いファミリに対して暗号化スキップを適用する**
  ランサムウェアはファミリやファミリ内の亜種により，プログラムの実装や実行ファイル生成に使用されたコンパイラは異なる．このため，暗号化処理の内容やコードサイズについても異なる可能性がある．そのため，暗号化スキップを行うコードサイズをランサムウェアの検体毎に動的に決定可能な仕組みが必要となる．
- **ランサムウェアでない，通常のアプリケーションによる暗号化に悪影響を与えない**
  提案手法が適用された OS では，ランサムウェアではない通常のアプリケーションについても動作しており，暗号化処理が実装されている場合がある．それらのアプリケーションの実行時の振舞いを示すセマンティクスに悪影響を与えないような設計が必要となる．

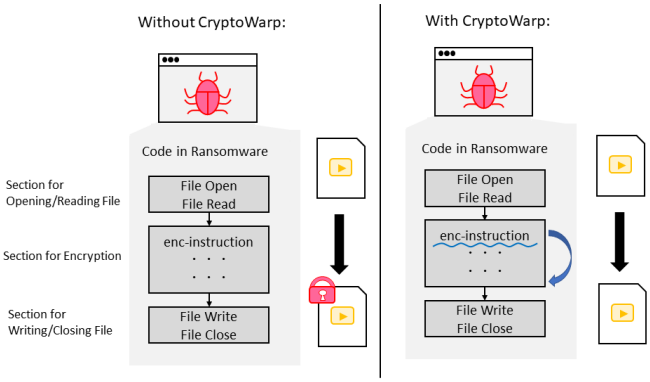

図 2 CryptoWarp による暗号化スキップの概観

5. 設計

5.1 暗号化スキップ

プロセスから透過的に暗号化処理の無効化を行うため，CryptoWarp では暗号化開始時の命令をトラップし，暗号化終了後の命令まで実行をスキップする．図 3 に，暗号化スキップの実現に必要となる暗号化処理のコードサイズの特定，暗号化開始時のプロセスの実行中断，ならびにコードサイズ分の暗号化命令のジャンプの処理の流れを示す．

5.1.1 コードサイズの特定

プロセスロード時に実行可能ページをスキャンし，暗号化処理開始にあたる命令と終了にあたる命令を探索する．また，開始命令と終了命令の仮想アドレスにおける差を取ることで，暗号化処理のコードサイズを特定する．

LockBit におけるコードサイズ特定例

リスト 1 に，LockBit における AES 暗号化処理の x86 アセンブリを示す．暗号化処理開始にあたる命令は *AddRoundKey* 操作を行う pxor 命令である．暗号化処理終了にあたる操作は *SubBytes*，*ShifRows*，*AddRoundKey* であり，aesenclast 命令が対応する．aesenclast 命令の直後の movups 命令により，暗号化結果をレジスタからメモリ内にコピーするため，暗号化処理終了の命令は movups である．これらの開始命令と終了命令の仮想アドレスの差分を算出することで，コードサイズを特定する．

aesenclast 命令後に暗号化結果をレジスタからメモリ内にコピーする命令については，ランサムウェアにより種類が異なる．例として，LockBit は movups 命令を用いるが，Solaso [12] では movdqu 命令を使用する．ランサムウェアファミリによる命令の違いを吸収するため，CryptoWarp

では 128 ビット mov 命令の種類をバイナリから識別するためのパーサを導入する．パーサを用いてコピーに伴う命令のサイズを特定し，最終的なコードサイズを決定する．

5.1.2 プロセスの実行中断とコードジャンプ

暗号化開始時にプロセス実行を中断させるため，CPU のハードウェアブレークポイント機能であるデバッグレジスタを使用する．デバッグレジスタには，ページスキャン時に特定した暗号化開始命令の仮想アドレスを登録する．

デバッグレジスタへの登録により，プロセスによる該当の仮想アドレスに格納される暗号化命令の実行時には，ハードウェアデバッグ割り込みが発生する．OS において割り込みはカーネル内に存在するハンドラにより処理される．プロセスによる暗号化実行開始時に，CryptoWarp によるハンドリングを可能とするため，CryptoWarp はハードウェアデバッグ割り込みハンドラをフックする．

ハンドリング時，CryptoWarp が監視対象とするプロセスかチェックを行う．対象の場合，ページスキャン時に算出したコードサイズをプロセスのユーザ空間におけるプログラムカウンタに加算する．この操作により，暗号化開始時の命令から終了後の命令までコードのジャンプされ，暗号化スキップを実現する．

5.2 信頼可能なアプリケーションへの対応

ランサムウェアでない通常のアプリケーションが暗号化処理を実行する場合，暗号化スキップによりプログラムのセマンティクスに影響を与える可能性がある．このため，CryptoWarp では信頼可能なユーザにより実行されるアプリケーションのページスキャンを回避する仕組みとして，UID Policy を提供する．

UID Policy はページスキャンを除外するユーザ ID (UID) を管理するためのポリシーリストである．ページスキャン開始時に対象となるプロセスの UID が UID Policy と一致するかチェックを行う．一致する場合にはページスキャンを行わない．UID Policy により，信頼可能なユーザにより実行開始されるプロセスは CryptoWarp の適用対象外として実行する．UID Policy の読み込みと書き込みは特権ユーザのみ許可し，攻撃者による操作を困難とする．

6. 実装

本稿における CryptoWarp の実現環境は x86 アーキテクチャの Linux を想定している．本章では，CryptoWarp の実装内容について述べる．

6.1 ページスキャン

ページスキャンは，ユーザプロセス内に実行可能ページが生成されたタイミングで開始される．タイミングは以下の三種類に分類することができる．

(1) exec により，実行可能な仮想アドレス領域がマップされ，実行可能ページが生成されたタイミング

(2) mmap により，実行可能な仮想アドレス領域がマップされ，デマンドページングにより実行可能ページが生成されたタイミング

(3) mprotect により，既存のページに実行可能権限が付与されたタイミング

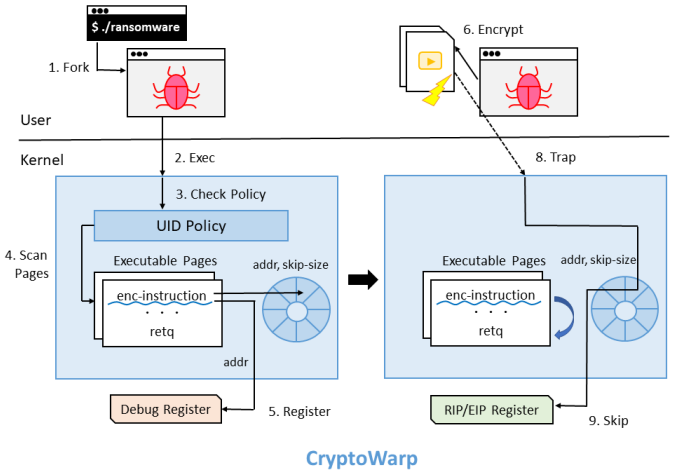

図 3 CryptoWarp の設計の概観

Listing 1 LockBit における暗号化処理

```
1     nop
2     lea     0x10(%eax),%eax
3     movups  -0x10(%esi,%eax,1),%xmm0
4     pxor    %xmm0,%xmm1
5     pxor    (%ecx),%xmm1
6     aesenc  0x10(%ecx),%xmm1
7     aesenc  0x20(%ecx),%xmm1
8     aesenc  0x30(%ecx),%xmm1
9     aesenc  0x40(%ecx),%xmm1
10    aesenc  0x50(%ecx),%xmm1
11    aesenc  0x60(%ecx),%xmm1
12    aesenc  0x70(%ecx),%xmm1
13    aesenc  0x80(%ecx),%xmm1
14    aesenc  0x90(%ecx),%xmm1
15    aesenclast 0xa0(%ecx),%xmm1
16    movups  %xmm1,-0x10(%eax)
17    sub     $0x1,%edx
18    jne     0x43d8e0
19    pop     %esi
20    mov     %ebp,%esp
21    pop     %ebp
22    mov     %ebx,%esp
23    pop     %ebx
24    ret
```

6.1.1 exec におけるページスキャン

exec では do_exec 関数の呼び出しをトリガーとし，CryptoWarp の持つ cw_exec 関数を呼び出す．cw_exec 関数では，コードがマップされている仮想アドレスの領域を特定し，ページスキャン用の関数である cw_start_scanning 関数に対して領域の範囲を渡す．このため，実行中スレッドの vm_area_struct 構造体を取得し，VM_EXEC フラグの有効な仮想アドレス領域を検索する．これにより，VM_EXEC が有効な領域の vm_start および vm_end を取得し，暗号化命令の含まれる仮想アドレスの範囲を特定する．

6.1.2 mmap におけるページスキャン

実行可能領域をマップする mmap においてもページス

キャンを行う必要がある．mmap 呼び出し時，物理ページの生成は行われない．対象コードを実行する際にページフォールト発生し，物理ページが生成されるため，ページフォールト後の物理ページの生成タイミングにおいてページスキャンを行う．

最初に，ksys_mmap_pgoff 関数の呼び出しをトリガーとし，引数 prot が PROT_EXEC ビットを含む場合に，仮想アドレスとサイズを記録する．記録は CryptoWarp の持つ双方向リスト対して要素を追加することで行う．次に，do_page_fault 関数をトリガーとし， CryptoWarp の持つ cw_mmap 関数を呼び出し，ページフォールトを引き起こした仮想アドレスがリストに存在するかチェックする．存在する場合，cw_start_scanning 関数を呼び出す．

6.1.3 mprotect におけるページスキャン

mprotect では do_mprotect_pkey 関数の呼び出しをトリガーとし，引数 prot が PROT_EXEC ビットを含む場合，CryptoWarp の持つ cw_mprotect 関数を呼び出す．cw_mprotect 関数の引数には，mprotect システムコール時の仮想アドレスとサイズが渡され，対象の仮想アドレスに対応する物理ページが存在するかチェックする．存在する場合， cw_start_scanning 関数を呼び出す．存在しない場合，mmap と同様に双方向リストを用いた仮想アドレスとサイズの記録を行う．

6.1.4 コードサイズの計測

cw_start_scanning 関数では，暗号化処理のコードサイズの計測，および開始命令の仮想アドレスをデバッグレジスタに対して登録する．AES-NI を用いた暗号化演算では，128 ビット長レジスタの xmm レジスタが使用される．

コードサイズの計測として，開始命令について， *AddRoundKey* 操作にあたる，xmm を用いた 128 ビット XOR 演算である pxor 命令を検索する．次に，同一ページ内に aesenclast 命令が存在するかチェックする．aesenclast は可変長命令であるため，先頭 4 バイトのオペコード部分を用い，一致するバイト列パターンを特定する．

バイト列パターンの特定後，5.1.1 節 にて述べた mov 命令用のパーサを用いて，終了命令にあたる仮想アドレスを特定する．パーサは，対応する mov 命令が一致した場合に命令サイズを返す．aesenclast 命令の仮想アドレスに対して，aesenclast 命令のサイズ，およびパーサを用いて特定した mov 命令のサイズを加算した値が終了命令の仮想アドレスとなる．

一連の処理より，開始命令と終了命令の仮想アドレスを特定し，両仮想アドレスの差を取りコードサイズを決定する．その後，開始命令の仮想アドレスとコードサイズの組を CryptoWarp の持つリングバッファに登録し，同時に開始命令の仮想アドレスをデバッグレジスタへ登録する．

6.2 デバッグ割り込みの設定とハンドリング

6.2.1 デバッグレジスタへの登録

CryptoWarp におけるデバッグレジスタの設定では，Linux の持つ register_user_hw_breakpoint 関数を用いて，対象となる仮想アドレスをデバッグレジスタに登録する．これにより，対象のスレッドがデバッグレジスタに登録した仮想アドレスの命令の実行を試みた場合，ハードウェアデバッグ割り込みが発生する．

CryptoWarp では，task_struct 構造体に識別用フラグとして cw_target_dr エントリを追加し，本エントリを有効にすることでハードウェアデバッグ割り込み時に，対象スレッドが CryptoWarp のハンドリング対象か識別する．

6.2.2 割り込みのハンドリング

ハードウェアデバッグ割り込み時に呼び出される do_debug 関数をトリガーとし，割り込みを引き起こしたスレッドの cw_target_dr が有効であるかチェックする．有効である場合，CryptoWarp の対象スレッドと判断し，暗号化命令スキップを行う．スキップ時のサイズは CryptoWarp の持つリングバッファを参照し，暗号化スキップに必要なコードサイズを取得する．

6.3 UID Policy

現在の CryptoWarp の実装では，信頼可能なユーザ設定の仕組みを実現するため，procfs を使用している．`/proc/cryptowarp/scanning_policy` に対して信頼可能なユーザの UID を書き込むことで，CryptoWarp に対して通知が行われる．CryptoWarp では，cw_start_scanning 開始時に current_uid 関数を発行し，通知済の UID と一致する場合には，ページのスキャンを回避する．なお，`scanning_policy` は root ユーザのみ読み書きを可能とするファイルパーミッションが設定されており，攻撃者によるファイルの読み書きは困難である．

7. 評価

提案手法の評価実験として，ランサムウェアを模した PoC コードによる暗号化の攻撃検証，およびパフォーマンスに与えるオーバーヘッドの測定を行う．評価の目的と内容を以下に示す．

- **攻撃の検証**
  CryptoWarp が適用された OS において，AES-NI を用いてファイル暗号化を実施する PoC を実行し，暗号化を防止可能か評価した．
- **パフォーマンスの計測**
  CryptoWarp が適用および未適用の OS において，マイクロベンチマークならびに実アプリケーションに近いワークロードを動作させ，パフォーマンスを計測する．パフォーマンスの比較を行うことで，CryptoWarp が性能に与える影響を定量的に特定した．

表 1　実験環境

CPU	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz，12 Core
Memory	49067528 KB

表 2　攻撃結果の比較（✓ 暗号化防止成功; – 暗号化成功）

概要	提案手法未適用	提案手法適用
LockBit の実装を模した PoC を実行	–	✓

実験環境を表 1 に示す．CryptoWarp は Linux 5.7.15 に対して実装を行い，1435 行を追加することで実現した．

7.1　攻撃の検証

CryptoWarp 適用および未適用の OS にて，ランサムウェアの動作を模した PoC を動作させ，暗号化の攻撃が成立するか否かを確認した．PoC は LockBit の AES-NI の命令列を参考に実装した．入力として与えられたファイルパスが指し示すファイルについて，AES-NI を用いて暗号化を行い，暗号化されたコンテンツを拡張子 .enc のファイルとして出力する．攻撃可否の判断基準については，暗号化前と暗号化後のファイルの SHA256 ハッシュ値を比較し，ハッシュ値に差異がある場合には暗号化がなされており，攻撃成功と見なす．また，ハッシュ値が同一の場合には，暗号化防止が成功していると見なす．

PoC の実行結果を表 2 に示す．CryptoWarp が適用された OS では，ファイル暗号化前と暗号化後のハッシュ値は一致し，暗号化は無効化されていることを確認した．

7.2　パフォーマンスの計測

マイクロベンチマーク

CryptoWarp を構成する各コンポーネントが性能に与える影響について調査した．ベンチマークについては，専用の計測プログラムおよび OpenSSL Speed を使用した．

専用の計測プログラムは AES-NI による暗号化を一回行う C 言語実装によるプログラムとした．16 バイト分の配列に格納された文字列を 10 命令分の aesenc ならびに 1 命令分の aesenclast を用いて暗号化する．このとき，計測プログラムをプロセスとして実行を開始し，終了するまでの実行時間計測を行った．

OpenSSL Speed [13] は，OpenSSL ライブラリによる暗号化速度の性能を計測するためのコマンドであり，1 秒間に暗号化を行うことが可能なバイト数を算出し，実験環境におけるスループットを計測した．本ベンチマークでは，OpenSSL 1.1.1o において，EVP API と呼ばれる AES-NI 命令を使用するためのモードを有効にした上で実行した．

以上のベンチマークを以下の 3 種類の OS 環境で実行し，結果を比較した．

(1) CryptoWarp 未適用のネイティブ OS
(2) CryptoWarp を構成するコンポーネントのうち，ページスキャンのみが有効な状態の OS

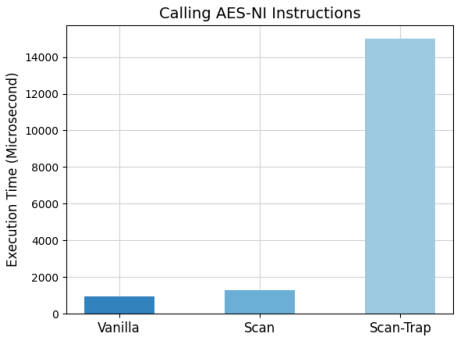

図 4　AES-NI を用いて 16 バイト分の暗号化を行うベンチマークにおける実行時間の比較

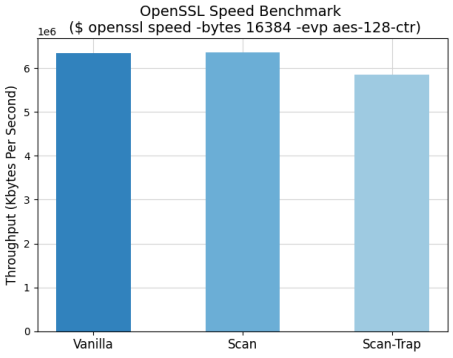

図 5　OpenSSL Speed ベンチマークにおけるスループットの比較

(3) CryptoWarp を構成するコンポーネントのうち，ページスキャンおよび命令トラップが有効な状態の OS

専用の計測プログラムにおける実行結果を図 4 に示す．CryptoWarp の未適用状態からページスキャンコンポーネントを追加することで約 1.34 倍，トラップコンポーネントを追加することで約 15.73 倍の実行時間増加となることが示された．

OpenSSL Speed による実行時間結果を図 5 に示す．CryptoWarp の未適用状態とページスキャンコンポーネント追加状態での差は見られず，トラップコンポーネント追加により，約 8.43 % のスループット減少が確かめられた．

マクロベンチマーク

CryptoWarp を構成するコンポーネントが全て適用された状態において，実環境に近いワークロードを動作させ，性能に与える影響について調査した．ベンチマークについては UnixBench 5.1.3 [11] を使用し，CryptoWarp 未適用のネイティブ OS と CryptoWarp でのパフォーマンススコアを比較した．

UnixBench の実験結果を図 6 に示す．各ベンチマーク項目のうち，スコアに大きな差が見受けられた項目は *Execl Throughput*, *Shell Scripts (1 concurrent)*, *Shell Scripts (8 concurrent)* であり，CryptoWarp 適用によるスコア減少の割合は，それぞれ 232.20 %，125.26 %，118.24 % 程度であった．また，ベンチマーク全体の評価スコアである *System Benchmarks Index Score* でのスコア減少割合は約

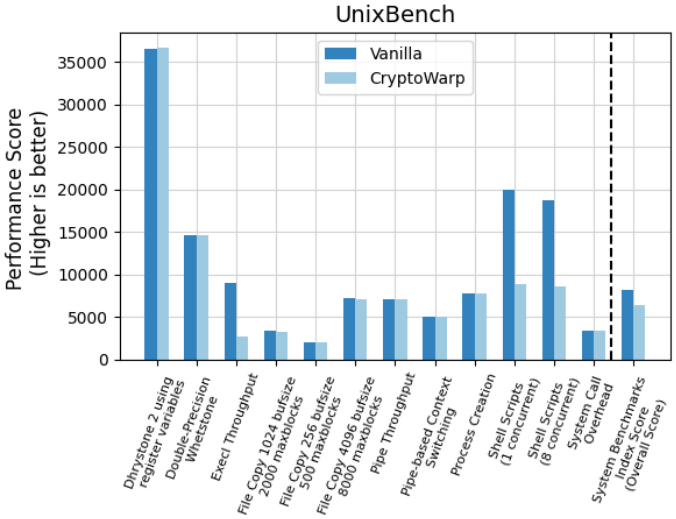

図 6 UnixBench におけるパフォーマンススコアの比較

26.83 % 程度であった．

8. 考察

8.1 評価結果に対する考察

セキュリティ評価

7.1 節では，暗号化実施前後でのファイルハッシュ値を比較し，暗号化が無効化されていることを確かめた．しかし，ランサムウェアが暗号化後のファイルに対してファイルサイズやタイムスタンプ等のメタデータを付加するようなケースが存在するため [8]，暗号化を無効化した場合においても，ハッシュ値は一致しない場合がある．ハッシュ値が一致しない場合においては，暗号化後のファイルに対し，メタデータ除去を行い，暗号化前ファイルと比較を行う評価手法が求められる．

パフォーマンス評価

CryptoWarp を構成する各コンポーネントの導入により発生するオーバーヘッド，および CryptoWarp 全体のオーバーヘッドを示した．

AES-NI を用いた暗号化を一回行うベンチマークでは，ページスキャンの追加により，約 1.34 倍，命令トラップの追加により，約 15.73 倍の実行時間増加が確認された．実行時間への影響から，CryptoWarp における主なオーバーヘッドの要因は，命令トラップによるハードウェアデバッグ割り込みといえる．

OpenSSL Speed ベンチマークにおいては，プログラムがプロセスとしてロードされた後に性能の計測を開始するため，ページスキャンによるオーバーヘッドがスループットに対して影響を与えない．そのため，命令トラップにかかるコストが CryptoWarp 全体のオーバーヘッドとなる．

UnixBench では，全ベンチマーク項目にて AES-NI は使用されない．そのため，ページスキャンにかかるコストが CryptoWarp のオーバーヘッドとなる．結果として，*Execl Throughput*，*Shell Scripts (1 concurrent)*，ならびに *Shell Scripts (8 concurrent)* のスコア低下が顕著に見られた．一方，他のベンチマーク項目には，スコア差異は計測されなかった．スコア低下のあるベンチマーク項目は exec システムコールを繰り返す挙動を行う．このため，exec を繰り返し発行する短期稼働のアプリケーションはオーバーヘッドが大きくなる傾向がある．一方，exec システムコールを一回のみ発行し，長期稼働を行うようなアプリケーションにおいては，オーバーヘッドを無視できるといえる．

8.2 移植可能性

x86 アーキテクチャにおける CryptoWarp は，ハードウェアブレークポイント機能であるデバッグレジスタを用いて，対象の命令実行時にデバッグ例外割り込みを発生させ，カーネルによるハンドリングを行う．他のアーキテクチャにおいても，デバッグレジスタが搭載されている場合，CryptoWarp における同様の仕組みを実現可能である．例として，ARMv7-A アーテキチャが実装されている ARM Cortex-A7 では 6 個のハードウェアブレークポイントが搭載されており，ARM 向け Linux ではこれらのデバッグレジスタはサポートされている．

デバッグレジスタをサポートする OS においては，本稿における Linux 実装での CryptoWarp と同様の機能を移植可能である．例として，FreeBSD では exec，mmap，mprotect システムコールの発行時にそれぞれ sys_execve，sys_mmap，sys_mprotect 関数を呼び出すことから，これらをページスキャンのトリガーとすることが可能である．また，実行中のスレッドのデバッグレジスタ値は struct pcb 構造体に格納され，load_dr0 から load_dr7 までの関数を用いることで，デバッグレジスタへの設定が可能である．

9. 関連研究

暗号化 API を用いる既存研究

ランサムウェアが暗号化 API を使用する点に着目した手法として PayBreak [8] がある．PayBreak はランサムウェアによる暗号化後にファイルの復旧を行うことを目的としている．PayBreak では，Windows の暗号化ライブラリである CryptoAPI のライブラリコールをフックし，暗号鍵のトレースを行う．また，トレースにより得られた暗号鍵を Key Vault に格納し，攻撃成功後に Key Vault から暗号鍵を取得，復旧処理を可能とする．PayBreak では，ライブラリを利用せずに直接 AES-NI 命令を用いて暗号化を行うランサムウェアへの対応は困難であり，CryptoWarp による補完は可能である．

デコイファイルを用いる既存研究

ランサムウェアは暗号化を行うファイルを選定するため，暗号化の前段階で様々なファイルへのアクセスを行う．この特性に着目した既存研究として Routa et.al による手

法 [9] がある．Routa らの手法は専用のデコイファイルを複数用意し，スレッド単位でデコイファイルに対するアクセスを監視する．デコイファイルへのアクセスが一定の閾値を越えた場合には，対象のスレッドをランサムウェアと判定し，プロセスを停止する．しかし，デコイファイルを回避するランサムウェア [14] への対応は困難であるため，CryptoWarp による補完は有効である．

ファイルのエントロピーを用いる既存研究

暗号化後のファイルコンテンツにエントロピーの偏りが生じる点に着目した手法として UNVEIL [7]，Redemption [15] がある．UNVEIL はファイル I/O のリクエストを監視し，データバッファのエントロピーを計算することで，暗号化を行うプロセスを検出する．Redemption はファイルのエントロピーや，ファイルに対する上書きの割合，ならびにファイル削除を行ったか否かといった複数の項目を用いて独自のスコアを定義し，プロセスがスコアを超過した場合にはランサムウェアと判定する．これらの手法はいずれもカーネル内でのランサムウェア対策であり，CryptoWarp と同様にランサムウェアに対策環境を検出させないよう，透過性の高いシステムとして実現されている．

10. おわりに

ランサムウェアによるファイル暗号化被害の防止にはアンチウイルスの導入が有効である．一方，アンチウイルスは未知のランサムウェアへの対応には一定の時間を要する．既存研究では，デコイファイルやファイル I/O のエントロピーを用いた検出手法が提案されている．しかし，既存研究はランサムウェアによるファイル暗号化の検出までに一定時間を要し，ファイル暗号化を許容するため，未知のランサムウェアに対する早期的な緩和は不完全である．

本稿では，信頼できないプロセスによる暗号化を無効化する手法を備えた CryptoWarp を提案した．CryptoWarp はカーネル内で動作し，プロセスの利用する実行可能ページのスキャンとデバッグレジスタを組み合わせることで，AES-NI 命令による暗号化処理のスキップを実現する．

評価として CryptoWarp を Linux に実装し，ランサムウェアを模した PoC によるファイル暗号化処理を防止可能であることを確認した．また，パフォーマンスの評価により CryptoWarp によるオーバーヘッドを定量的に明らかにし，多くの環境において許容可能なオーバーヘッドで実行可能であることを示した．

謝辞 本研究の一部は総務省の「電波資源拡大のための研究開発 (JPJ000254)」における委託研究「電波の有効利用のための IoT マルウェア無害化／無機能化技術等に関する研究開発」によるものである．

参考文献

[1] Microsoft: Win32/Reveton threat description, https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32%2FReveton. (accessed 2022-06-10).
[2] TREND MICRO: Ransomware Raises the Stakes With CryptoLocker, https://www.trendmicro.com/vinfo/de/threat-encyclopedia/web-attack/3132/ransomware-raises-the-stakes-with-cryptolocker. (accessed 2022-06-10).
[3] TREND MICRO: WannaCry/Wcry Ransomware: How to Defend against It, https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/wannacry-wcry-ransomware-how-to-defend-against-it. (accessed 2022-06-10).
[4] Brierley Calvin et.al: PaperW8: an IoT bricking ransomware proof of concept, *Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20)*, ACM, pp. 1–10 (2020).
[5] TREND MICRO: Analysis and Impact of LockBit Ransomware ' s First Linux and VMware ESXi Variant, https://www.trendmicro.com/en_us/research/22/a/analysis-and-Impact-of-lockbit-ransomwares-first-linux-and-vmware-esxi-variant.html. (accessed 2022-06-10).
[6] ClamAV Documentation: Updating Signature Databases, https://docs.clamav.net/manual/Usage/SignatureManagement.html. (accessed 2022-06-15).
[7] Amin Kharaz et.al: UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware, *25th USENIX Security Symposium (USENIX Security '16)*, Austin, TX, USENIX Association, pp. 757–772 (2016).
[8] Kolodenker Eugene et.al: PayBreak: Defense Against Cryptographic Ransomware, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17)*, Association for Computing Machinery, pp. 599–611 (2017).
[9] Moussaileb Routa et.al: Ransomware's Early Mitigation Mechanisms, *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES '18)*, ACM, pp. 1–10 (2018).
[10] Intel: Intel® Advanced Encryption Standard Instructions (AES-NI) , https://www.intel.com/content/www/us/en/developer/articles/technical/advanced-encryption-standard-instructions-aes-ni.html. (accessed 2022-06-10).
[11] kdlucas: byte-unixbench, https://github.com/kdlucas/byte-unixbench. (accessed 2022-06-10).
[12] MalwareBazaar: Information on Solaso malware sample, https://bazaar.abuse.ch/browse/signature/Solaso/. (accessed 2022-06-10).
[13] The OpenSSL Project: openssl-speed, https://www.openssl.org/docs/manmaster/man1/openssl-speed.html. (accessed 2022-06-10).
[14] Han Jaehyun et.al: On the Effectiveness of Behavior-Based Ransomware Detection, *Proceedings of the 16th International Conference on Security and Privacy in Communication Systems (SecureComm '20)*, Springer, pp. 120–140 (2020).
[15] Kharraz Amin et.al: Redemption: Real-Time Protection Against Ransomware at End-Hosts, *Proceedings of the 20th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID '17)*, Springer, pp. 98–119 (2017).