ランサムウェアに対する破壊的書き込みの監視による仮想ディスク保護機構

榎本 秀平[1]　葛野 弘樹[1]　山田 浩史[2]　白石 善明[1]

概要：仮想マシン (VM) は，Infrastructure as a Service を展開するクラウドプロバイダや，オンプレミスにおいて広く運用されている． 近年では，VM のホストの脆弱性や設定不備を突いたランサムウェアによる，仮想ディスクの暗号化被害が広く確認されている． ランサムウェアの検出および防御においては，ユーザプロセスの振る舞いの監視による検出や，ファイルコンテンツのリカバリを可能とするストレージデバイスといった既存手法が提案されているが，VM のホストにおいては適用が困難である． 振る舞いの監視による検出手法は，ランサムウェアの一定時間の動作を必要とするために，被害発生後の検出となる可能性がある． リカバリを可能とするストレージデバイスは，False Positive および False Negative の発生頻度が高い点が指摘されており，仮想ディスクにおけるデータコンテンツの消失が生じる可能性がある．本研究では，VM のホストにて，ランサムウェアによるファイル暗号化から仮想ディスクを保護するための新たな手法を提案する． 提案手法では，ランサムウェアが仮想ディスクを暗号化する際に，仮想ディスクのデータフォーマットを考慮することのない，破壊的な書き込みを行う点に着目した． 事前に定義したフィルタリングルールに基づき，仮想ディスクに対する変更が，データフォーマットを考慮した書き込みであるかを監視し，破壊的な書き込みを検出および防止する． 提案手法を Linux Kernel Module として実装し，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仮想ディスクに対する暗号化を検出および防止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

Effective VM Disk Protection against Ransomware Encryption

Shuhei ENOMOTO[1]　Hiroki Kuzuno[1]　Hiroshi YAMADA[2]　Yoshiaki SHIRAISHI[1]

Abstract: Ransomware attacks targeted virtual storages of virtual machines raise serious damage to virtual machine users and hypervisor providers. The existing ransomware detections incur unacceptable false positives and negatives. Operating these detections in the virtualization environment can lead to stopping the virtual machines. This study proposes a novel ransomware defense mechanism for virtual storage on the hypervisor. The proposal mechanism achieves low false positives and negatives by monitoring suspicious file writing that compromises the file format of virtual storage. We implemented our proposal mechanism and conducted experiments regarding security and performance. The experimental result indicated effective prevention against real-world ransomware attacks and negligible performance overhead.

1. はじめに

仮想マシン (VM) を用いたコンピュータシステムは，インターネットやローカルネットワークにて広く運用されており，VM に構築されたアプリケーションにより数多くのネットワークサービスが展開されている．インターネットにおいては，クラウドプロバイダにより Infrastructure as a Service (IaaS) が提供され，IaaS ユーザは VM を用いたサービスの運用が可能となる．例として，Netflix では

[1] 神戸大学
Kobe University
[2] 東京農工大学
TUAT

Amazon Web Services による IaaS である Elastic Compute Cloud を用いた VM により，ストリーミングサービスを展開する [1]．ローカルネットワークにおいては，プライベートクラウドを構築可能なプラットフォームが広く登場しており，プライベートクラウドユーザは VM を用いたサービスの運用が可能となる．例として，Adobe では VMware によるプライベートクラウドである vSphere を用いて，ビッグデータに対する分散処理を展開する [2]．

近年では，VM ホストにて稼働するネットワークサービスの脆弱性を突いた攻撃が広く確認されている．攻撃者は脆弱性により，VM ホストにてランサムウェアを実行する．例として，DarkSide [3] では，VMware ESXi の脆弱性である CVE-2021-21974 [4] を用いて，VM ホストにてファイル暗号化を試行する．VM ホストにて動作するランサムウェアは，VM の仮想ディスクをファイル暗号化の標的とし，仮想ディスクの復旧を代償とした身代金の支払いをクラウド管理者に対し要求する．

ランサムウェア対策として，既存研究ではユーザプロセス動作時の振る舞いに着目した検出手法 [5–9] や，ファイルコンテンツのリカバリを可能とするストレージ [10,11] が提案されている．しかし，既存研究における手法を VM ホストに適用するためには，以下の課題が存在する．

- **ファイル暗号化検出の遅れ**
  ユーザプロセスの振る舞い監視によりランサムウェアを識別する手法 [5–9] は，識別にあたりランサムウェアの振る舞いを要する．したがって，検出までの間にランサムウェア動作を許容する必要があり，デコイファイルの監視による検出手法 [8] においては，デコイファイルへのアクセス検出時に仮想ディスクが既に暗号化されている可能性を持つ．
- **ファイル暗号化の誤検知および検出見逃し**
  ユーザプロセスの振る舞いを監視する手法のうち，ファイル暗号化の実被害が生じる前に検出を可能とする手法 [5–7,9] が提案されている．しかし，正規のファイル暗号化や，攻撃者による回避への対応が十分に検討されていないために，誤検知および検出見逃しが生じうる点が既存研究の調査 [12] にて指摘されている．
  ランサムウェア攻撃後にファイルコンテンツのリカバリを可能とするストレージ [10,11] では，リカバリのトリガーとなるランサムウェア検出にて，誤検知および検出見逃しの発生頻度が高い点が既存研究の調査 [13] にて指摘されている．したがって，仮想ディスクに対する誤ったリカバリや，リカバリの見逃しにより，VM 内のファイルコンテンツが消失する可能性を持つ．

本研究では，VM ホストにてランサムウェアによるファイル暗号化から仮想ディスクを保護するための新たな手法である STED[*1] を提案する．VM による仮想ディスクへのファイル書き込みは，仮想ディスクのフォーマットやファイルシステム (FS) のセマンティクスに従い，書き込みパターンにおいて規則性を持つ．ランサムウェアによる仮想ディスクへのファイル暗号化は，これらの規則性に従うことのない，破壊的書き込みを行う．STED は仮想ディスクへの書き込みを監視，破壊的書き込みを発行するユーザプロセスを検出，終了することで，仮想ディスクの保護を行う．STED は仮想ディスクに対するファイル暗号化が開始された早期段階より誤検知および検出見逃しが生じることなく保護を実現する．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から，仮想ディスクを保護するための機構として STED を提案した．STED は既知および未知のランサムウェアに対し，仮想ディスクに対するファイル暗号化を開始した段階からの早期的な保護を実現する．
(2) 破壊的書き込みを検出するための設計と実装を示した．STED の実現にあたり，要求される既存の OS の変更は僅かである．
(3) STED を Linux Kernel Module として実装し，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 2.0 [14] の実装を模した PoC，および 6 種類の実検体から仮想ディスクを保護可能であることを実証した．パフォーマンス評価としてマイクロベンチマークおよび UnixBench [15] を実行し，VM の実行性能に与える影響を確認した．

2. 背景

2.1 仮想マシンの運用

クラウドプロバイダによる IaaS や，オンプレミスにおけるプライベートクラウドでは，VM の運用をユーザが可能とするサービスが展開される．ユーザは VM の生成および運用が可能であり，IaaS のクラウドプロバイダやプライベートクラウドの管理者は，VM の実行環境を構成するコンポーネントの運用を担う．例として，ハイパーバイザ，仮想ディスクといった VM ホストにおけるソフトウェアコンポーネントや，ストレージといったハードウェアコンポーネントがある．

ストレージ I/O に焦点を置いた VM の実行環境の概観を図 1 (A) に示す．VM が仮想ディスクに対する I/O を行った際，VM ホストにおけるハイパーバイザ，I/O エミュレータに遷移がなされる．I/O エミュレータにてストレージに対する I/O がなされ，ストレージからの応答により，ハイパーバイザ，VM に遷移がなされる．

2.1.1 QEMU + Linux KVM における構成

I/O エミュレータおよびハイパーバイザのソフトウェア

[*1] STED: STorage Encryption Defense

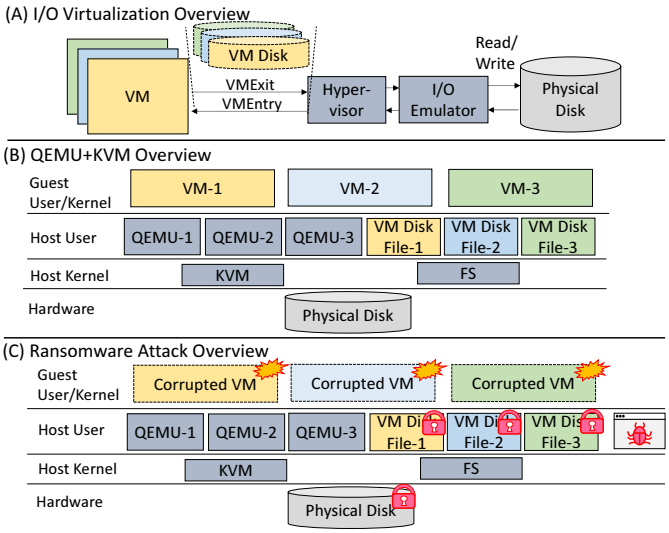

図 1 I/O 仮想化の概観，QEMU+KVM [16,17] における構成，および VM ホストを標的とするランサムウェアによる攻撃

として，QEMU + Linux KVM [16,17] を使用した際の構成を図 1 (B) に示す．VM による I/O 要求は，VM ホストにおける Linux カーネル空間に位置する KVM によって捕捉され，ユーザ空間に位置する QEMU によって I/O エミュレーションがなされる．QEMU は，VM ホスト上の FS にてファイルとして配置される仮想ディスクに対し，FS を介した I/O を行う．FS はストレージに対し，デバイスドライバを通じた I/O を行う．

2.2 ランサムウェアによる攻撃

ランサムウェアは，感染先ホスト上のリソースの使用を不可能にすることや，ファイルコンテンツの盗取を行い，リソース復旧やファイルコンテンツの公開を控えることの代償として，ユーザに対し身代金を要求する．リソース使用を不可能にする手法として，FS におけるファイル暗号化が広く取り入れられている [18]．

従来のランサムウェアは，エンドユーザにおけるクライアント PC を標的とするが，近年のランサムウェアは VM ホストをはじめとする，サービスを運用するコンピュータについても標的とする傾向にある [19]．VM ホストを標的とするランサムウェアの概観を図 1 (C) に示す．VM ホストに侵入した攻撃者は，ユーザ空間にてランサムウェアを実行し，ランサムウェアは仮想ディスクのファイル暗号化を試行する．仮想ディスクの暗号化により，VM 内の FS の破壊がなされ，VM は実行不可能となる．

3. 脅威モデル

本研究の脅迫モデルとして，攻撃者は VM ホストにて稼働するネットワークサービスの脆弱性や，盗取されたユーザの認証情報を用いて，標的とする VM ホストにアクセス可能であるものと想定する．攻撃者は VM ホストのユーザ空間にてランサムウェアの実行開始が可能であり，ランサムウェアは仮想ディスクのファイル暗号化を試行する．攻撃者による OS のユーザは，VM に対する停止や起動が可能な権限を持ち，仮想ディスクに対し読み込みや書き込みといったファイル操作権限を持つ．カーネル空間やハードウェアに対する攻撃は，本研究における対象外とする．

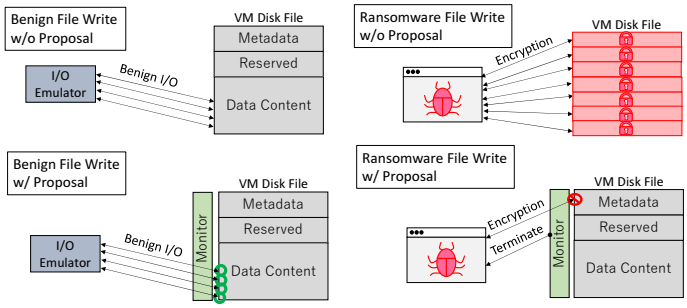

図 2 STED による仮想ディスク保護の概観

4. 提案手法

本研究では，VM ホストにてランサムウェアによる仮想ディスクに対するファイル暗号化を緩和し，仮想ディスクの保護を行うための新たな手法である STED 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：ランサムウェア起動時からの早期的な検出および仮想ディスク保護を可能とする．

要件 2：ランサムウェアによるファイル暗号化時の正確な検出および仮想ディスク保護を可能とする．

要件 1 を満たすため，STED はランサムウェアが標的とする仮想ディスクのうち，最初のファイルが被害を受ける前にランサムウェア検出および仮想ディスク保護が可能な機構として設計を行う．

要件 2 を満たすため，STED は仮想ディスクに対する正規の I/O による誤検知，ランサムウェアによるファイル暗号化時の検出見逃しが最小となる機構として設計を行う．

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配置後，ユーザプロセスとしてランサムウェアを実行，FS 内における仮想ディスクのファイル暗号化を試みる．

STED では，ランサムウェアが仮想ディスクのファイル暗号化を行う際にて，仮想ディスクのデータフォーマットを考慮することのない，破壊的な書き込みを行う点に着目する．VM ホストにて，仮想ディスクに対し書き込みが行われる際にて，データフォーマットの整合性が維持された書き込みパターンであるかを監視する．維持されない書き込みパターンにおいては，対象の書き込みパターンを破壊的書き込みと見なし，ユーザプロセスの終了を行う．

STED による仮想ディスク保護の概観を図 2 に示す．仮想ディスクのデータフォーマットにて，メタデータにおける Read Only 領域や予約領域をはじめとする，正規の I/O

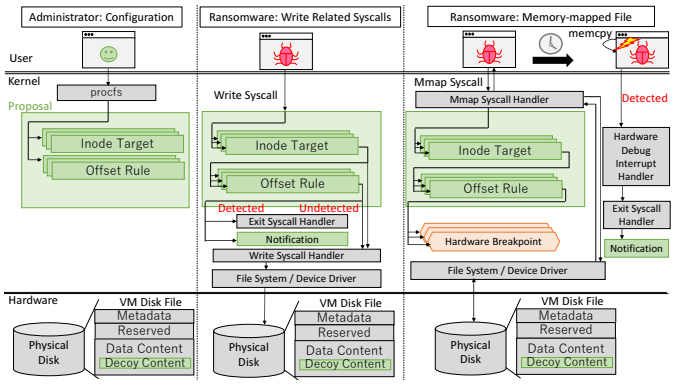

図 3　STED の設計の概観

においては変更されることのない領域に対する変更要求を監視する．変更要求が検出された際においては破壊的書き込みと見なし，ユーザプロセスの終了を行う．

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．

- **仮想ディスクおよびランサムウェアの種類に依存することのない設計**
  仮想ディスクの種類はハイパーバイザや I/O エミュレータの種類によって異なり，ランサムウェアファミリにおいても多様な種類が確認されている．STED では，特定の仮想ディスクやランサムウェアの種類に依存することなく，幅広い VM ホスト環境の多様なランサムウェア向けの対策として導入が可能な設計が求められる．
- **検出回避に対する緩和を有する設計**
  既存研究の調査 [12, 18] は，既存のランサムウェア対策手法の設計が攻撃者に認識されていた場合にて，検出を回避するようにランサムウェアによるファイル暗号化を継続可能である点を指摘している．VM ホストにて検出回避によってファイル暗号化の継続がなされた場合においては，VM 内にて運用されるサービスの停止といった二次被害が生じうる．STED では，攻撃者による検出回避に対し，一定の緩和を有するような設計が求められる．

5.　設計

5.1　破壊的書き込みの検出

ランサムウェアによる仮想ディスクに対する暗号化を防止するため，VM ホストにおけるユーザプロセスからのファイル書き込みに関する操作を捕捉，事前定義されたフィルタリングルールに基づいた書き込みの監視を行う．破壊的書き込みの検出がなされた際には，当該のユーザプロセスを終了し，管理者に対し検出通知を行う．図 3 に，フィルタリングルールの登録，書き込みの監視，検出時のユーザプロセス終了，管理者への通知の流れを示す．

5.1.1　フィルタリングルールの登録

VM ホストにおける管理者となる特権ユーザは，STED に対し以下のフィルタリングルールを登録する．

- **ファイル Inode 番号**
  仮想ディスクの Inode 番号を STED に対して登録する．ユーザプロセスによるファイル書き込み時にて，STED は登録された Inode に対する書き込みか否かを監視する．
- **ファイルオフセット**
  仮想ディスクに対する書き込みにて，正規の書き込みにおいては書き換えが生じることのないファイルオフセットを STED に対して登録する．ユーザプロセスによるファイル書き込み時にて，登録された Inode に対する書き込みである場合においては，STED は登録されたファイルオフセットに対する書き込みか否かを監視する．

5.1.2　ファイル書き込みの監視

STED は以下の二種類のファイル書き込み操作における監視を施行する．

- **システムコールによる書き込み**
  Linux における write システムコールをはじめとする，システムコールによるファイル書き込みを捕捉する．捕捉後にて，フィルタリングルールによる監視が施行される．登録された Inode かつ，ファイルオフセットへの書き込みである場合には，Linux における exit システムコールをはじめとする，プロセス終了の操作を行い，仮想ディスクへの書き込みを防ぐ．
- **Memory-mapped File による書き込み**
  Linux における mmap システムコールをはじめとする，ユーザプロセスの仮想アドレス空間に対するファイルのマップ，仮想アドレスへのアクセスを通じたファイル書き込みを捕捉する．ファイルマップ時にて，登録された Inode のマップである場合においては，登録されたファイルオフセットに該当する仮想アドレスを算出，ハードウェアブレークポイントに対して登録を行う．ハードウェアブレークポイントにより書き込みが検出された場合においては，書き込み箇所のロールバック，プロセス終了の操作を行う．

5.1.3　管理者に対する検出通知

破壊的書き込みの検出後，管理者への通知の役割を担うサーバに対しネットワーク接続を行い，検出を示すメッセージを送信する．STED より検出メッセージを受信したサーバは，管理者に対し検出通知を行う．

5.2　デコイコンテンツの配置

ランサムウェアによる回避を緩和するため，STED は VM 内における FS に対しランダムなデータが格納され

る専用のファイル (デコイコンテンツ) を挿入する．仮想ディスク内にて，デコイコンテンツが配置されるファイルオフセットをフィルタリングルールとして登録する．

仮想ディスク内におけるデコイコンテンツが配置されるファイルオフセットに対し，ユーザプロセスが書き込みオペレーションを発行した際には，破壊的書き込みと見なし，ユーザプロセスの終了を行う．

6. 実装

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

6.1 フィルタリングルールの登録

STED は，VM ホストにおける管理者となる特権ユーザに対し，procfs を通じた設定インタフェースを提供する．表 1 に示すように，インタフェースはファイル Inode およびファイルオフセットに関する設定項目を持ち，root ユーザからのファイル Inode およびファイルオフセットの設定入力を受け付ける．設定入力を受けた STED は，フィルタリングルールとして Inode およびファイルオフセットの登録を行い，ルールを用いたファイル書き込み監視を行う．

6.2 システムコールによる書き込みの監視

write システムコールにより，監視対象の Inode を持つファイルに対する書き込みが行われる際にて，監視対象のファイルオフセットに対する書き込みであるかを確認する．

STED では，システムコールのフックにより write システムコールの発行を捕捉し，引数に指定されるファイルディスクリプタが示す Inode を特定する．Inode が監視対象である場合においては，ファイルディスクリプタよりファイルオフセットを取得し，監視対象のファイルオフセットが書き込みの範囲に含まれるかを確認する．書き込みの範囲に含む場合においては，破壊的書き込みと見なし，対象のユーザプロセスの終了を行う．

6.3 Memory-mapped File による書き込みの監視

mmap システムコールにより，監視対象の Inode を持つファイルがマッピングされた際にて，監視対象となるファイルオフセットの仮想アドレスをハードウェアブレークポイントとして登録する．

ハードウェアブレークポイントの登録

x86 においては，ハードウェアデバッグレジスタに対し仮想アドレスの設定を行うことで，対象の仮想アドレスへのアクセス時にてハードウェアデバッグ割り込みが生じる．Linux カーネルにおいては，register_user_hw_breakpoint 関数にてハードウェアデバッグレジスタに対する登録を行い，do_debug 関数にてハードウェアデバッグ割り込みに対するハンドリングを行う．

表 1 フィルタリングルールの概要

ルール名	役割	procfs ファイルパス
Inode	監視対象のファイル Inode の設定	/proc/sted/inode
Offset	監視対象のファイルオフセットの設定	/proc/sted/offset

STED では，システムコールのフックにより mmap システムコールの発行を捕捉する．ファイルマッピングを目的とする mmap システムコールにおいては，引数に指定されるファイルディスクリプタが示す Inode を特定する．Inode が監視対象である場合においては，ファイルマッピング後の先頭仮想アドレスから監視対象となるファイルオフセットを加算した仮想アドレスを算出する．算出された仮想アドレスについて，読み込みおよび書き込み時にハードウェアデバッグ割り込みが生じるデータブレークポイントとして，デバッグレジスタに対し登録する．

ハードウェアデバッグ割り込みの捕捉

STED は do_debug 関数のフックを行い，ハードウェアデバッグ割り込みの発生を捕捉する．割り込み発生の原因となったスレッドが，監視対象の Inode を持つファイルをマッピングしたプロセスに属す場合は，監視対象の仮想アドレスが書き換えられているかを確認する．読み込みによる割り込み時にて，監視対象の仮想アドレスのデータコンテンツを取得し，書き込みによる割り込み時にて，読み込み時に取得したデータコンテンツと比較を行い，書き換えを検出する．書き換え検出後，読み込み時に取得したデータコンテンツを用いてロールバックを行い，書き換え前のデータコンテンツに状態を復元する．加えて，対象のユーザプロセスが破壊的書き込みを行ったと見なし，対象のユーザプロセスの終了を行う．

6.4 管理者に対する検出通知

破壊的書き込みの検出により，対象のユーザプロセスの終了がなされた後，管理者への通知の役割を担うサーバに対し TCP ソケットを用いたネットワーク接続を行い，検出を示すメッセージを送信する．Linux カーネルにおいて，ソケットは sock_create 関数により生成がなされ，connect 関数によりサーバに対する接続を行い，send_message 関数によりメッセージの送信を行う．

7. 評価

提案手法の評価実験として，ランサムウェアによる仮想ディスクに対するファイル暗号化の攻撃検証，正規アプリケーションによる仮想ディスク操作における誤検知の発生検証，VM の実行性能に与える影響の計測を行った．評価の目的と内容を以下に示す．

- **攻撃の検証**
  STED が適用された OS にて，ファイル暗号化を実施する PoC およびランサムウェア実検体を実行し，暗

表 2 実験環境

Guest CPU Core	2
Guest Memory	4 GiB
Guest Kernel	Linux 5.4.0-153-generic
Guest OS Distribution	Ubuntu 20.04.6 LTS (Focal Fossa)
Host CPU	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
Host CPU Core	6
Host Memory	48 GiB
Hypervisor	QEMU 4.2.1, Linux KVM 5.4.0-182-generic
仮想ディスク	QCOW2, 2.8 GiB

表 3 ファイル暗号化の結果

概要	ハッシュ値	仮想化ディスク保護
PoC (write)	一致	成功
PoC (mmap)	一致	成功
AvosLocker [20]	一致	成功
Conti [21]	不一致	成功
Monti [22]	不一致	成功
RansomEXX [23]	不一致	成功
RansomEXX2 [24]	一致	成功
ReEvil [25]	一致	成功

号化を防止可能か評価した．

- **誤検知の検証**
  STED が適用された OS にて，仮想ディスクの操作を行う正規のアプリケーションを実行し，STED が正規のアプリケーションに対し誤検知を行うか評価した．
- **パフォーマンスの計測**
  STED が適用および未適用の OS にて，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．パフォーマンス結果を比較し，STED が性能に与える影響を定量的に特定した．

STED の実験環境を表 2 に示す．STED は Linux Kernel Module として 802 行の実装を行うことで実現した．ハイパーバイザとして QEMU + Linux KVM が稼働するマシンにて，Linux Kernel Module として STED をインストールし，評価を実施した．

7.1 攻撃の検証

表 2 に示す Guest OS が VM として 1 台構築され (VM-1)，停止されている状態にて，VM ホストにてランサムウェアを実行し，ファイル暗号化が防止可能か否かを確認した．ランサムウェアの実行前後にて，対象の仮想ディスクの SHA256 ハッシュ値を比較し，ハッシュ値の一致が確認された場合にはファイル暗号化が防止されたと見なす．

STED におけるフィルタリングルールとして，VM-1 における QCOW2 ファイルの Inode 番号を登録し，QCOW2 ファイルにおける 0x00000000 に格納される QCOW magic string および 0xa3f80000 に格納されるデコイコンテンツをファイルオフセットとして登録した．

ランサムウェアは，ファイル暗号化を実施する PoC および QCOW2 ファイルをファイル暗号化の標的とする 6 種類の実検体を実行した．PoC は LockBit 2.0 [14] の実装を模倣したファイル暗号化プログラムである．FS 内にて .qcow2 拡張子を持つファイルの探索，AES-NI [26] を用いたファイルコンテンツの暗号化，write または mmap システムコールによるファイルコンテンツの上書きを行う．

結果を表 3 に示す．PoC，AvosLocker [20]，RansomEXX2 [24] および ReEvil [25] においては実行前後における QCOW2 ファイルのハッシュ値の一致が確認された．Conti [21]，Monti [22] および RansomEXX [23] においては，暗号化コンテンツによる上書きを開始する前に，暗号鍵をはじめとするメタデータをファイル末尾に書き込みを行う．したがって，実行前後における QCOW2 ファイルのハッシュ値は不一致であった．ただし，QCOW2 ファイル内のデータコンテンツに対する暗号化がなされる前に検出がなされ，STED によりランサムウェアとして動作するユーザプロセスの終了がなされた．

7.2 誤検知の検証

VM ホストにて，仮想ディスクに対し正規のアプリケーションによる変更を行った際における，誤検知発生の有無を確認した．変更対象となる仮想ディスクは，7.1 節と同様の VM-1 における QCOW2 ファイルであり，STED に対し 7.1 節と同様のフィルタリングルール設定を行った．QCOW2 ファイルに対する変更は qemu-img コマンド [27] を用いた，仮想ディスクの整合性チェック (check)，他フォーマットへの変換 (convert)，スナップショットの作成 (snapshot)，サイズの変更 (resize) を実施した．結果，各操作にて，STED による誤検知は確認されなかった．

7.3 パフォーマンスの計測

7.3.1 マイクロベンチマーク

VM ホストにおけるファイル書き込みにて，STED 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を用いた．

専用の計測プログラムは，単一のファイルに対し，ランダムに生成された 4 KiB のデータコンテンツを 10^5 回上書きし，完了にかかる時間の測定を行う．計測プログラムを STED 未適用時，適用時にて実行し，結果の比較を行った．STED 適用時にて，上書き対象となる単一のファイルの Inode 番号はフィルタリングルールとして STED に対し登録がなされている．また，STED 適用時において，オフセットのフィルタリングルールが 1 個，10 個，100 個の設定パターンをそれぞれ用意し，各設定パターンにて実行，結果の比較を行った．

結果を図 4 に示す．オフセットのフィルタリングルールが 1 個，10 個，100 個の設定パターンにて，それぞれ 0.077

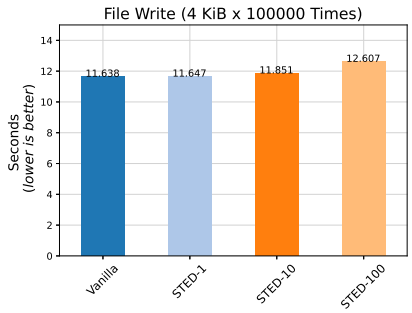

図 4　マイクロベンチマーク実行結果の比較

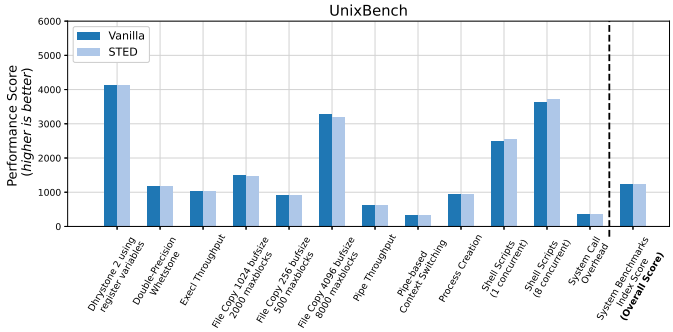

図 5　UnixBench 実行結果の比較

%，1.83 %，8.32 % 程度の実行時間増加が確認された．

7.3.2　マクロベンチマーク

表 2 に示す Guest OS が VM として 3 台構築され (VM-1, VM-2 および VM-3)，全ての VM が稼働中の状態にて，VM-1 内で実アプリケーションに近いワークロードを動作させ，STED 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 UnixBench 5.1.3 [15] を使用し，VM ホストにて STED を未適用時と適用時におけるパフォーマンススコアを比較した．STED 適用時においては，VM-1, VM-2 および VM-3 の仮想ディスクの Inode 番号がフィルタリングルールとして STED に対し登録がなされている．また，STED 適用時におけるオフセットのフィルタリングルールは 7.1 節における設定と同様である．

結果を図 5 に示す．各項目にて，スコアに大きな差異は確認されなかった．また，ベンチマーク全体の評価スコアである *System Benchmarks Index Score* のスコアは，STED 未適用時が 1248 に対して，適用時が 1249 であり，0.08 % 程度の差異であった．

8.　考察

8.1　評価結果に関する考察

セキュリティ評価

7.1 節にて，仮想ディスクに対する暗号化を開始する前に，ファイルサイズ，タイムスタンプおよび暗号鍵といったメタデータを仮想化ディスクに付加情報として書き込むランサムウェアが確認された．付加情報の書き込みは STED よる監視を通過し，ファイル書き込みがなされるため，ハッシュ値の不一致となる．現在の STED プロトタイプにおいて，付加情報の書き込みを行うランサムウェアの実行前後にて仮想ディスクのハッシュ値の一致を保証するためには，手動による付加情報の除去が必要となる．

パフォーマンス評価

7.3.1 節にて，STED が引き起こすオーバーヘッドは，フィルタリングルールに登録されるエントリ数に依存することが確かめられた．7.3.2 節においては，7.1 節にて実際のランサムウェア防御が達成された際におけるフィルタリングルール数が用いられた．QCOW magic string およびデコイコンテンツの登録がなされたエントリ数 2 の構成では，STED によるオーバーヘッドは生じず，VM の実行性能に影響を与えないことが確かめられた．

8.2　他の仮想ディスクやハイパーバイザに対する適用

フィルタリングルールは管理者が STED に対して与える設計であり，特定の仮想ディスクのフォーマットに依存した設計および実装でない．したがって，QCOW2 ファイルに限らず，他のフォーマットを持つ仮想ディスクに対しても適用が可能と考えられる．例として，FS を直接的に扱うフォーマットである QEMU RAW ファイルにて EXT4 ファイルシステムが配置されている場合においては，スーパーブロックにおける Magic signature の書き換え監視を行うことで，本稿と同様の機能を実現可能である．

I/O エミュレータによる仮想ディスクへのファイル書き込み処理に対するフックが可能なハイパーバイザにおいては，STED の適用が可能である．例として，Xen においては Domain 0 のカーネル空間にてファイル書き込みの監視を行うことで，本稿と同様の機能を実現可能である．

9.　おわりに

VM の仮想ディスクを標的としたランサムウェアによるファイル暗号化の攻撃は，VM のユーザやハイパーバイザの管理者に対し深刻な被害をもたらす．既存研究におけるランサムウェア検出手法は，誤検知および検出見逃しの発生傾向が高い点が既存調査より指摘されている．したがって，仮想化環境に対する既存の検出手法の適用は，VM の意図しない停止や，データ消失を引き起こす可能性がある．

本稿では，VM の仮想ディスクを標的としたランサムウェアによるファイル暗号化の攻撃を防御，仮想ディスクを保護するための新たな手法である STED を提案した．STED は VM ホストにて仮想ディスクに対するファイル書き込みを監視，仮想ディスクのデータフォーマットに基づいた規則的な変更でない書き込みを検出，書き込み元ユーザプロセスの終了を行う．

評価として STED を Linux Kernel Module として実装し，実際のランサムウェアによるファイル暗号化を防御可能であることを確かめた．また，正規アプリケーションによる仮想ディスクの変更時にて誤検知を引き起こさないこ

とを確認し，パフォーマンスの評価により STED によるオーバーヘッドを定量的に明らかにした．

謝辞 本研究は総務省の「電波資源拡大のための研究開発 (JPJ000254)」における委託研究「安全な無線通信サービスのための新世代暗号技術に関する研究開発」の成果を含みます．

参考文献

[1] Intel: Case Study: Netflix Chooses Amazon EC2 Instances with Intel® Xeon® Processors to Provide Fast and Seamless Streaming Experiences, https://www.intel.com/content/www/us/en/content-details/823269/case-study-netflix-chooses-amazon-ec2-instances-with-intel-xeon-processors-to-provide-fast-and-seamless-streaming-experiences.html. (accessed 2024-08-06).

[2] VMware: Big Data on vSphere : Two Customer Case Study White Papers Published, https://blogs.vmware.com/vsphere/2015/05/big-data-vsphere-two-customer-case-study-white-papers-published.html. (accessed 2024-08-06).

[3] Trend Micro: DarkSide on Linux: Virtual Machines Targeted, https://www.trendmicro.com/en_us/research/21/e/darkside-linux-vms-targeted.html. (accessed 2024-08-06).

[4] The MITRE Corporation.: CVE - CVE-2021-21974, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-21974. (accessed 2024-08-06).

[5] Amin Kharaz et.al: UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware, *Proceedings of the 25th USENIX Security Symposium (Security '16)*, Austin, TX, USENIX Association, pp. 757–772 (2016).

[6] Continella Andrea et.al: ShieldFS: a self-healing, ransomware-aware filesystem, *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16)*, ACM, pp. 336–347 (2016).

[7] 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).

[8] 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).

[9] Mehnaz Shagufta et.al: RWGuard: A Real-Time Detection System Against Cryptographic Ransomware, *Proceedings of the 21st International Symposium on Research in Attacks, Intrusions, and Defenses (RAID '18)*, Springer, pp. 114–136 (2018).

[10] Baek SungHa et.al: SSD-Insider: Internal Defense of Solid-State Drive against Ransomware with Perfect Data Recovery, *Proceedings of the 38th International Conference on Distributed Computing Systems (ICDCS '18)*, IEEE, pp. 875–884 (2018).

[11] Baek SungHa et.al: SSD-Assisted Ransomware Detection and Data Recovery Techniques, *IEEE Transactions on Computers, vol. 70, no. 10*, IEEE, pp. 1762–1776 (2021).

[12] 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).

[13] Zhongyu Wang et.al: Ransom Access Memories: Achieving Practical Ransomware Protection in Cloud with DeftPunk, *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24)*, USENIX, pp. 687–702 (2024).

[14] 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 2024-08-23).

[15] kdlucas: byte-unixbench, https://github.com/kdlucas/byte-unixbench. (accessed 2022-06-23).

[16] KVM: Kernel Virtual Machine, https://linux-kvm.org/page/Main_Page. (accessed 2024-08-11).

[17] QEMU: A generic and open source machine emulator and virtualizer, https://qemu.org/. (accessed 2024-08-11).

[18] McIntosh Timothy et.al: Ransomware Mitigation in the Modern Era: A Comprehensive Review, Research Challenges, and Future Directions, Vol. 54, No. 9, ACM (2021).

[19] VMware: ESXi-Targeting Ransomware: The Threats That Are After Your Virtual Machines (Part 1), https://blogs.vmware.com/security/2022/09/esxi-targeting-ransomware-the-threats-that-are-after-your-virtual-machines-part-1.html. (accessed 2024-08-11).

[20] Trend Micro: Ransomware Spotlight: AvosLocker, https://www.trendmicro.com/vinfo/us/security/news/ransomware-spotlight/ransomware-spotlight-avoslocker. (accessed 2024-08-19).

[21] Trend Micro: Ransomware Spotlight: Conti, https://www.trendmicro.com/vinfo/us/security/news/ransomware-spotlight/ransomware-spotlight-conti. (accessed 2024-08-19).

[22] Trend Micro: Monti Ransomware Unleashes a New Encryptor for Linux, https://www.trendmicro.com/en_us/research/23/h/monti-ransomware-unleashes-a-new-encryptor-for-linux.html. (accessed 2024-08-19).

[23] Trend Micro: Ransomware Spotlight: RansomEXX, https://www.trendmicro.com/vinfo/us/security/news/ransomware-spotlight/ransomware-spotlight-ransomexx. (accessed 2024-08-19).

[24] The Hacker News: New RansomExx Ransomware Variant Rewritten in the Rust Programming Language, https://thehackernews.com/2022/11/new-ransomexx-ransomware-variant.html. (accessed 2024-08-19).

[25] Trend Micro: Ransomware Spotlight: REvil, https://www.trendmicro.com/vinfo/us/security/news/ransomware-spotlight/ransomware-spotlight-revil. (accessed 2024-08-19).

[26] 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 2024-08-19).

[27] QEMU documentation: QEMU disk image utility, https://qemu-project.gitlab.io/qemu/tools/qemu-img.html. (accessed 2024-08-19).