

エンクレイヴとセキュリティモニタの連携による IDS の安全な実行

岩野 空仁^{1,a)} 光来 健一¹

概要: IoT 機器はインターネットからの攻撃を受けやすいため、侵入検知システム (IDS) による監視が必要である。しかし、監視対象システム上で動作する IDS は、攻撃者によって無効化される可能性がある。この問題に対し、高信頼実行環境 (TEE) を用いて IDS を安全に実行する手法が提案されている。一方で、既存手法には、IDS に過大な権限が付与されることや、監視対象システムのメモリへアクセスする際のオーバーヘッドが大きいなどの課題がある。本稿では、RISC-V の TEE であるエンクレイヴと特権ソフトウェアのセキュリティモニタを連携させることで IDS を安全に実行する Keyspector を提案する。Keyspector では、セキュリティモニタによるアクセス制御を用いて、IDS が動作するエンクレイヴのみが監視対象システムのメモリにアクセスできるようにする。これにより、IDS はシステムメモリに格納された OS データを直接取得できる。さらに、セキュリティモニタが IDS の実行状態を監視することで、侵入者による IDS の無効化を検知する。Keyspector を RISC-V のエンクレイヴおよびセキュリティモニタに実装し、その有効性と性能を調べた。

Secure IDS Execution through Enclave–Security Monitor Coordination

TAKAHITO IWANO^{1,a)} KOURAI KENICHI¹

Abstract: Internet of Things (IoT) devices need to be monitored using intrusion detection systems (IDS) because they tend to suffer from attacks from the Internet. However, host-based IDS running inside target systems may be disabled by attackers. To address this problem, several monitoring systems have been proposed to securely execute IDS using trusted execution environments (TEEs). Nevertheless, existing systems have several issues, including significant overhead for accessing the memory of the target system and excessively high privileges for IDS. In this paper, we propose Keyspector, which securely executes IDS through coordination between an enclave and a security monitor. In Keyspector, access control enforced by the security monitor allows only the enclave running IDS to access the memory of the target system. This enables the in-enclave IDS to directly obtain OS data stored in the system memory. In addition, the security monitor inspects the execution state of the in-enclave IDS to detect attempts by attackers to disable it. We implemented Keyspector in the enclave and the security monitor and examined its effectiveness and performance.

1. はじめに

近年、Internet of Things (IoT) が急速に普及しており、2034 年までに IoT 機器は 406 億台まで増加すると予測されている [1]。Wi-Fi ルータ、プリンタ、テレビ、自動車といった IoT 機器はインターネットを介してサーバや他の機器と接続されるため、外部から攻撃を受けるリスクが高

い。実際、IoT 機器にマルウェアを感染させることを目的とした攻撃は増加しており [2]、IoT 機器内に侵入され、分散サービス妨害 (DDoS) 攻撃などに利用されたり [3]、機器内で動作するプログラムが改ざんされたり [4] する事例が発生している。

安全に IoT 機器を運用するために、侵入検知システム (IDS) を動作させて監視を行う必要があるが、監視対象システム内で動作する IDS は侵入者に無効化される恐れがある。そこで、最近のプロセッサが提供している高信頼実

¹ 九州工業大学
Kyushu Institute of Technology
^{a)} iwano@ksl.ci.kyutech.ac.jp

行環境 (TEE) を用いて、IDS を安全に実行する手法が提案されている。例えば、Arm TrustZone のセキュアワールドと呼ばれる保護領域で IDS を安全に実行する手法 [5] では、IDS がシステムメモリに直接アクセスして OS データの監視を行う。しかし、セキュアワールドは IDS が必要とするよりも高い権限を持っており、IDS の不具合でシステムメモリを破壊する可能性がある。IoT 向けではないが、Intel SGX のエンクレイヴと呼ばれる保護領域で IDS を実行する手法 [6] では、エンクレイヴの権限が低いため、監視対象システムへの影響は小さい。しかし、エンクレイヴはシステムメモリに直接アクセスできないため、他の信頼できるソフトウェアを介してメモリデータを取得する必要がある、オーバーヘッドが大きい。

この問題を解決するために、本稿では RISC-V プロセッサの TEE である Keystone [7] のエンクレイヴと特権ソフトウェアのセキュリティモニタを連携させることで IDS を安全に実行する Keyspector を提案する。RISC-V のエンクレイヴも標準ではシステムメモリにアクセスできないが、セキュリティモニタにおいて RISC-V のメモリ保護機構である Physical Memory Protection (PMP) を用いることにより、エンクレイヴ内の IDS がシステムメモリへ直接アクセスできるようにする。エンクレイヴの悪用を防ぐために、セキュリティモニタは IDS を実行するエンクレイヴに対してのみシステムメモリの共有を許可する。さらに、セキュリティモニタは IDS の実行状態を監視し、IDS の実行を妨害する攻撃を検知する。Keyspector は、攻撃を検知した際にリモートホストへの通知や、セキュリティモニタによる強制的なネットワーク通信の遮断を行うことで、検知後の対応も可能にする。

Keyspector をファームウェアの OpenSBI 内で動作するセキュリティモニタとエンクレイヴ内で動作する Eyrie ランタイムに実装した。Keyspector 上で動作する IDS の一例として、監視対象システムの proc ファイルシステムによって提供されるシステム情報を取得する IDS を実装した。この IDS は LLVM フレームワーク [8] を利用することで、OS のソースコードを用いて開発した。IDS が OS データにアクセスできるようにするために、システムメモリ上にあるページテーブルを走査し、OS データの仮想アドレスを物理アドレスに変換する。Keyspector を用いて実験を行ったところ、監視対象システムおよび IDS に対する攻撃を検知できることを確認した。また、監視対象システム内で動作する従来の IDS と比較して、エンクレイヴ内で IDS を実行することによるオーバーヘッドは無視できるほど小さいことも確認した。

以下、2 章で IoT 機器の安全な監視における問題点について述べる。3 章でエンクレイヴとセキュリティモニタを連携させることで IDS を安全に実行する Keyspector を提案し、4 章で Keyspector の実装について説明する。5 章で

Keyspector のセキュリティについて分析を行い、6 章で Keyspector の動作確認と監視対象システム内で監視した場合との比較を行った実験について述べる。7 章で関連研究に触れ、8 章で本稿をまとめる。

2. IoT 機器の安全な監視

IoT 機器はインターネットに接続されるため、外部からの攻撃を受けるリスクが高い。攻撃者の侵入の糸口となる IoT 機器の脆弱性を完全に取り除くのは難しいため、侵入検知システム (IDS) を用いて攻撃を検知する方法がある。IDS は IoT 機器内のシステムの異常を監視し、管理者に対して通知を行う。しかし、監視対象システム内で IDS を安全に実行するのは難しい。なぜなら、システム内に侵入された場合、IDS を停止されたり、改ざんされたりする恐れがあるためである。

IDS を監視対象システムの OS カーネル内で動作させることで、ユーザ空間の攻撃者から保護することができる [9] が、OS カーネルに脆弱性があった場合は、OS カーネル内の IDS でさえ攻撃を受ける可能性がある。OS から隔離して IDS を実行する方法として、System Management Mode (SMM) を用いる手法 [10] がある。SMM は Intel や AMD のプロセッサが提供する動作モードの一つであり、BIOS 内でのみ使用できる。SMM で IDS を実行することで、OS に侵入した攻撃者から IDS を保護できる。しかし、SMM でプログラムを実行している間はシステム全体が停止するため、監視中のシステム性能への影響が大きい。また、BIOS の入れ替えは再起動を伴うため、IDS の導入・更新時のシステムへの影響も大きい。

そこで、プロセッサが提供している高信頼実行環境 (TEE) を用いて IDS を安全に実行する手法が提案されている。TEE の一つである Arm TrustZone を用いて IDS を安全に実行することができる [5]。TrustZone を用いる場合、セキュアワールドと呼ばれる実行環境で IDS を実行し、ノーマルワールドで動作するシステムを監視する。ITZ ライブラリは、セキュアワールドからノーマルワールドのシステムメモリ上の OS データに直接アクセスして監視を行う IDS の開発を可能にする。しかし、セキュアワールドはノーマルワールドのメモリに対するすべての権限を持っており、IDS の不具合によって異常な動作をした際にメモリ上の OS データを破壊してしまう恐れがある。

IoT 向けではないが、Intel SGX を用いると、アプリケーション内に作成されるエンクレイヴと呼ばれる保護領域で IDS を実行することができる [6]。SGX のエンクレイヴは TrustZone のセキュアワールドと比べてシステム資源へのアクセスが制限されているため、IDS が OS データを破壊する恐れはない。エンクレイヴ内のメモリは暗号化され、整合性が保証されるため、OS を信頼せずに IDS を安全に実行できる。一方で、この制約により、エンクレイヴ内か

らシステムメモリに直接アクセスすることはできない。そのため、BIOS 等呼び出してメモリデータの取得を行い、メモリデータの暗号化や整合性検査を行って侵入者による盗聴・改ざんを防ぐ必要がある。しかし、この BIOS 等の呼び出しとデータ保護のために、メモリデータ取得のオーバーヘッドが大きい。

3. Keyspector

本稿では、RISC-V プロセッサの TEE である Keystone のエンクレイヴと特権ソフトウェアのセキュリティモニタを連携させることで IDS を安全に実行する Keyspector を提案する。

3.1 脅威モデル

RISC-V プロセッサおよびセキュリティモニタを信頼し、これらには脆弱性が存在しないものと仮定する。また、IDS を実行するエンクレイヴ内のソフトウェアにも外部から侵害できる脆弱性は存在しないものとする。攻撃者はネットワーク経由で監視対象システムへ侵入し、OS および OS 上で動作するソフトウェアを完全に制御できるものとする。このため、攻撃者は、新たなエンクレイヴの作成、IDS を実行しているエンクレイヴを含む既存のエンクレイヴの終了、およびエンクレイヴを実行するプロセスのスケジューリングの変更が可能である。

ただし、攻撃者が監視対象システムのカーネルデータ構造やページテーブルを改ざんし、IDS に実際とは異なるシステム状態を観測させる攻撃は本研究の対象外とする。この種の攻撃は、仮想マシンイントロスペクション (VMI) をはじめとする、監視対象システムの外部からメモリを解析する監視方式に共通する制約である [11]。また、DMA を悪用したエンクレイヴのメモリに対する攻撃も本研究の対象外とする。これは、CPU を介さずに発行されるメモリアクセスに対して RISC-V のメモリ保護が適用されないためである。このような攻撃は I/O デバイスからのメモリアクセスを制御するハードウェア機構 [12] を用いることで防止できる。なお、デバイスへの物理的なアクセスを伴う攻撃も本研究の対象外とする。

3.2 設計

Keyspector のシステム構成を図 1 に示す。Keyspector はエンクレイヴ内で IDS を実行することにより、IDS の盗聴や改ざんを防ぐ。エンクレイヴと監視対象システムはこれらの動作する権限より上位の特権ソフトウェアであるセキュリティモニタ上で実行され、互いに強固に隔離される。エンクレイヴ内の IDS はシステムメモリにアクセスすることで、監視対象システムの OS データの監視を行う。

Keyspector はエンクレイヴ内の IDS が監視対象システムのメモリに直接アクセスすることを可能にする。従来

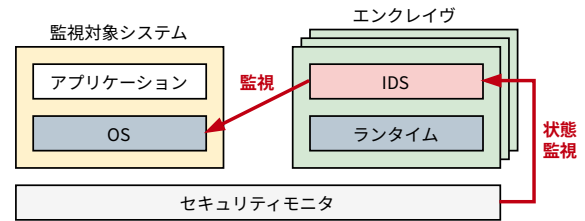


図 1 Keyspector のシステム構成

Fig. 1 The system architecture of Keyspector.

は、セキュリティモニタがシステムとエンクレイヴのメモリを論理的に分離しており、エンクレイヴからシステムメモリには一部を除いてアクセスすることはできない。Keyspector では、セキュリティモニタにおいてエンクレイヴがアクセス可能なメモリを変更することにより、システムメモリ全体を読み取りのみ可で共有できるようにする。これにより、IDS は効率よくシステムメモリ上の OS データにアクセスすることができ、たとえば異常な振る舞いをしたとしても、システムメモリを破壊することはない。

Keyspector のセキュリティモニタは、IDS が実行されているエンクレイヴに対してのみシステムメモリへのアクセスを許可する。これは、すべてのエンクレイヴがシステムメモリにアクセスできると、システムへの侵入者がエンクレイヴを実行してシステムメモリを盗聴できてしまうためである。セキュリティモニタは、リモートアテストーションのためにエンクレイヴ内部で実行されるソフトウェアのハッシュ値を計算している。そこで、Keyspector はリモートアテストーション時に検証サーバにおいてそのハッシュ値と登録された IDS のハッシュ値を比較し、一致した場合にのみ、システムメモリの共有を許可する。

Keyspector のセキュリティモニタは、エンクレイヴ内の IDS の実行状態を監視することで、攻撃者による IDS の実行妨害を検知する。攻撃者に監視対象システムの管理者権限を奪われると、IDS が動作するエンクレイヴを制御するための IDS プロセスを不正に停止されて、IDS を無効化される恐れがあるためである。このような攻撃を防ぐことは困難であるため、セキュリティモニタは IDS からの定期的な通知を受信できなくなった場合に、不正に停止されたと判断する。また、IDS プロセスのスケジューリングを妨害する攻撃を検知するために、セキュリティモニタが IDS の実行時間を計測し、閾値から大きく逸脱した場合に実行を妨害されていると判断する。

IDS の実行ログおよび実行状態は信頼できるリモートホストがセキュリティモニタから取得することができ、監視対象システムや IDS への攻撃を検知した場合には IoT 機器の管理者に通知する。しかし、誤検知もあり得るため、多数の IoT 機器を管理する環境では、管理者が個々の通知を確認し、それぞれに対応するには時間がかかる。その結果、当該 IoT 機器を介して感染が拡大したり、踏み台攻撃

に用いられたりする可能性がある。そこで、IDS やセキュリティモニタが攻撃を検知した際には、即応的な対策として IoT 機器のネットワーク通信を遮断することができる。これにより、攻撃の二次被害を防ぐことができる。

4. 実装

Keyspector をファームウェアの OpenSBI 内で動作するセキュリティモニタとエンクレイヴ内で動作する Eyrie ランタイムに実装した。

4.1 システムメモリの共有

Keyspector は Physical Memory Protection (PMP) を用いてエンクレイヴがシステムメモリを共有できるようにする。PMP はソフトウェアがアクセス可能な物理アドレスの範囲にアクセス権限を設定するための RISC-V 独自のメモリ保護機構である。従来は、システムとエンクレイヴがアクセスできる物理アドレスの範囲とアクセス権限をセキュリティモニタが PMP に設定する。システムからエンクレイヴにコンテキスト切り替えを行う際に、セキュリティモニタはシステムメモリに対するアクセス権限を無効にし、エンクレイヴに割り当てた物理メモリに対するアクセス権限のみを有効にする。逆にエンクレイヴからシステムにコンテキスト切り替えを行う際は、システムメモリに対するアクセス権限を有効にし、エンクレイヴに割り当てたメモリに対するアクセス権限を無効にする。

Keyspector では、システムからエンクレイヴにコンテキスト切り替えを行う際にシステムメモリに対する読み込み権限も有効にすることにより、エンクレイヴがシステムメモリを参照できるようにする。IDS からシステムメモリにアクセスできるようにするために、エンクレイヴ内のランタイムが IDS のメモリアドレス空間へシステムメモリをマッピングする。このマッピングは IDS がシステムコールを用いてランタイムを呼び出した際に行われる。

4.2 メモリ共有制御

Keyspector は、エンクレイヴ内で動作するソフトウェアのハッシュ値に基づいて、システムメモリの共有を許可するかどうかを判定する。セキュリティモニタはエンクレイヴの起動時に、その内部で実行されるローダ、ランタイム、およびアプリケーションのハッシュ値を計算する。これをエンクレイヴのハッシュ値と呼ぶ。ローダは軽量 OS であるランタイムを起動するために用いられ、ランタイムの上でアプリケーションを 1 つだけ実行する。計算したエンクレイヴのハッシュ値が事前に登録された正規のローダ、ランタイム、IDS のハッシュ値と一致した場合にのみ、システムメモリの共有を許可する。

Keyspector はこの判定をエンクレイヴのリモートアテステーションと同時に行う。エンクレイヴの起動後、IDS

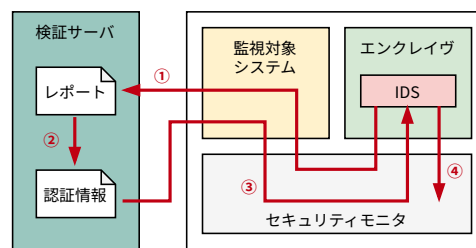


図 2 リモートアテステーションを用いたメモリ共有制御

Fig. 2 Remote attestation for controlled memory sharing.

はセキュリティモニタを呼び出し、アテステーションレポートを取得する。図 2 に示すように、IDS は OCALL を用いて監視対象システムを呼び出し、アテステーションレポートを検証サーバに送信する。検証サーバは、セキュリティモニタおよびエンクレイヴのハッシュ値が事前に登録しておいた値と一致することを確認すると、エンクレイヴのハッシュ値などの情報に署名した IDS 認証情報を返す。IDS は受け取った情報をセキュリティモニタに登録し、セキュリティモニタはその情報を検証して正規の IDS であればシステムメモリの共有を行う。

検証サーバに一時的に接続できない場合や、攻撃者による妨害に備えて、セキュリティモニタだけでシステムメモリの共有の可否を判断することもできる。この手法では、信頼できるサーバにおいてエンクレイヴのハッシュ値や IDS の登録・更新・削除といった操作に対して署名を行った IDS 認証情報を発行し、その情報をセキュリティモニタに登録しておく。エンクレイヴ起動時に、エンクレイヴのハッシュ値の計算をする際に、セキュリティモニタに登録されたハッシュ値との比較を行い、一致する場合は正規の IDS であることを示すフラグを有効にする。システムからエンクレイヴへのコンテキスト切り替え時に、当該フラグが有効であればシステムメモリの共有を行う。

4.3 OS データの監視

エンクレイヴ内で動作する IDS は OS データにアクセスするために、監視対象システムのページテーブルを用いて OS データの仮想アドレスを物理アドレスに変換する。そのために、まず、IDS はシステムメモリ上にあるページテーブルの物理アドレスを取得する。RISC-V ではページテーブルのアドレスは `satp` レジスタに格納されているが、エンクレイヴの実行中はエンクレイヴ用のページテーブルのアドレスが格納されている。システムの実行中に使われていた `satp` レジスタの値は、システムからエンクレイヴにコンテキスト切り替えを行う際にセキュリティモニタ内に保存されている。そこで IDS はセキュリティモニタを呼び出して、保存されている `satp` レジスタの値を取得する。その値からページテーブルを特定した後、IDS はページテーブルをたどって仮想アドレスに対応するページテーブルエントリを見つけ、物理アドレスを用いてシステムメ

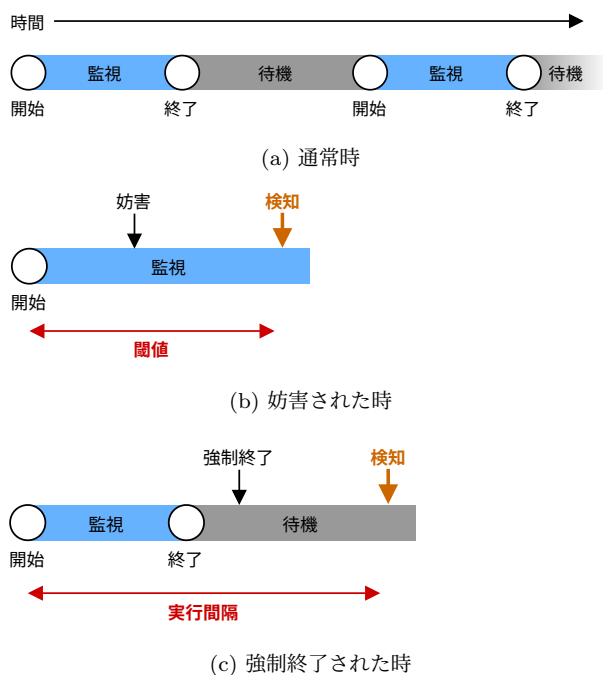


図 3 IDS の実行状態の監視と妨害の検知

Fig. 3 Monitoring IDS execution states and detecting execution interference.

メモリにアクセスする。

IDS は OS データを監視することにより監視対象システムへの攻撃を検知する。例えば、特定の TCP ポートへの通信や特定のプロセス名からマルウェアが動作していることを検知できる。攻撃を検知すると、IDS の実行ログに記録して 4.5 節の仕組みを用いて管理者に通知できるようにする。

4.4 IDS の実行状態の監視

Keyspector のセキュリティモニタは IDS の監視処理の実行時間と実行間隔を監視する。IDS の監視処理は通常時には図 3(a) に示すように周期的に行われる。そのため、図 3(b) のように、監視処理の実行時間が所定の閾値を超えた場合、IDS の実行が妨害されたと判断する。また、図 3(c) のように、次の監視処理が所定の実行間隔内に開始されない場合にも、IDS の実行が妨害されたと判断する。セキュリティモニタは IDS の実行妨害を検知すると、検知フラグを設定して 4.5 節の仕組みを用いて管理者に通知できるようにする。

実行時間の閾値は、実行環境に応じて適応的に決定する。具体的には、高負荷環境下で最初の 100 回の監視処理の実行時間を計測し、その分布から算出した四分位範囲 $IQR = Q_3 - Q_1$ を用いて、 $Q_3 + 1.5 \times IQR$ を閾値とする。 Q_1 、 Q_3 はそれぞれ第一四分位数、第三四分位数である。これにより、実行環境による処理時間の変動を考慮しつつ、正常な処理の誤検知を抑制する。

4.5 リモートホストへの通知

Keyspector のセキュリティモニタは、信頼できるリモートホストからの要求に応じて、IDS の実行ログおよび実行状態を含むレポートを返す。セキュリティモニタはリモートホストと直接通信できないため、監視対象システム上で動作する信頼できないホストプロセスを介して通信する。まず、リモートホストは、チャレンジを含むレポート要求をホストプロセスへ定期的送信する。ホストプロセスはこの要求を受信すると、監視対象システムの OS が提供するデバイスファイルを通じて、セキュリティモニタにレポートの生成を要求する。

セキュリティモニタは要求を受信すると、4.3 節で述べた IDS の実行ログと 4.4 節で述べた検知フラグからレポートを生成する。その後、生成したレポートを AES-GCM を用いて暗号化し、受信したチャレンジを追加認証データ (AAD) として使用する。これにより、レポートと要求との対応関係を検証し、過去に生成された暗号文を再送するリプレイ攻撃を検知できる。セキュリティモニタは、生成した暗号文をホストプロセスを介してリモートホストへ返送する。

リモートホストは、受信した暗号文を復号するとともに、認証タグと AAD を検証する。これらの検証に成功した場合、復号したレポートから IDS の実行状態および実行ログを確認する。検知フラグが設定されている場合、リモートホストは、当該 IoT 機器の管理者に対して IDS への攻撃が検知されたことを通知する。実行ログに攻撃の痕跡が記録されている場合、管理者に監視対象システムへの攻撃が検知されたことを通知する。

4.6 ネットワーク通信の遮断

ネットワーク通信を遮断するために、ネットワークデバイスが使用する MMIO 領域に対して、PMP を用いてアクセス権限を無効化する。これにより、監視対象システムから当該 MMIO 領域へのアクセスを禁止し、ネットワーク通信を強制的に停止させる。一方、Linux は PMP によるメモリ保護によって発生するアクセスフォールトを適切に処理できないため、MMIO 領域へのアクセスを単純に禁止するとカーネルパニックが発生する。そこで、ネットワークデバイスが使用する MMIO 領域へのアクセスをセキュリティモニタにおいてエミュレーションすることで、アクセスフォールトの発生時でもカーネルの実行を継続できるようにする。具体的には、当該 MMIO 領域に対するロード命令に対しては常に 0 の値を返し、ストア命令については書き込みを行わず、次の命令を実行する。

5. セキュリティ分析

Keyspector はエンクレーブ内で IDS を実行することにより、監視対象システムに侵入した攻撃者から IDS を保護

することができる。セキュリティモニタとエンクレイヴはメモリ保護機構のPMPを用いて保護されるが、PMPの設定はセキュリティモニタのみが変更することができる。そのため、監視対象システムの攻撃者はセキュリティモニタやエンクレイヴにアクセスできるようにPMPの設定を変更することはできない。

KeyspectorはIDSのバイナリファイル自体は保護しないため、攻撃者は当該バイナリの内容を解析することができる。しかし、KeyspectorではIDSのバイナリに秘密鍵などの機密情報を含める必要がないため、バイナリを解析されてもセキュリティは低下しない。また、IDSのバイナリの改ざんや改ざんされたIDSの起動を防ぐ仕組みも持たないが、改ざんされたIDSが動作するエンクレイヴに対しては監視対象システムのメモリを共有しない。当該IDSのハッシュ値が正規の値と一致なくなるためである。したがって、攻撃者は改ざんしたIDSを介して監視対象システムのメモリを盗聴したり改ざんしたりすることはできない。

KeyspectorはIDSの実行妨害を防ぐ仕組みを持たないが、セキュリティモニタにおいてこのような攻撃を検知することができる。IDSが動作しているエンクレイヴが停止された場合、IDSが監視処理の開始時と終了時に行う通知をセキュリティモニタが受信できなくなるため、実行妨害を検知することができる。IDSが正常にスケジューリングされない場合、IDSの監視処理の実行時間または実行間隔が閾値を超えるため、実行妨害を検知することができる。セキュリティモニタはタイマ割り込みの発生時に必ず実行されるため、攻撃者はこれらの検知を回避することはできない。

6. 実験

Keyspectorの有効性を調べるために、エンクレイヴで動作するIDSを用いて実験を行った。このIDSはシステムメモリ上のOSデータを解析してLinuxのprocファイルシステムが提供するシステム情報を取得する。procファイルシステムはIDSによってよく利用され、Linuxカーネルの状態が格納された疑似ファイルを作成してプロセス、ハードウェアおよびシステムリソースに関する情報を提供する。このIDSはOSデータのアドレス変換を透過的に行うことを可能にするLLViewフレームワーク [8]を用いて、Linuxカーネルのソースコードをベースにして開発した。このIDSは10種類の疑似ファイルを作成するために必要なOSデータを定期的に取得することでシステムを監視する。

このIDSを用いて、システム情報の取得時間やシステム性能への影響を調べた。比較として、監視対象システム内でprocファイルシステムを読み出す従来手法を用いた。本実験には、Freedom U740 SoCおよび16GBのメモリを搭載するSiFive製のHiFive Unmatched Rev Bを使用し、

OpenSBI 1.1 および Linux 6.1.43 を動作させた。

6.1 動作確認

Keyspectorを用いてエンクレイヴ内でIDSを実行し、監視対象システムのメモリ上のOSデータを収集して作成した疑似ファイルの内容を表示させた。この出力結果と監視対象システム内でエンクレイヴの実行直後にprocファイルシステムを読み出した時の出力を比較したところ、出力が一致することを確認した。この結果から、エンクレイヴ内のIDSがシステム情報を正しく取得できることが確認できた。

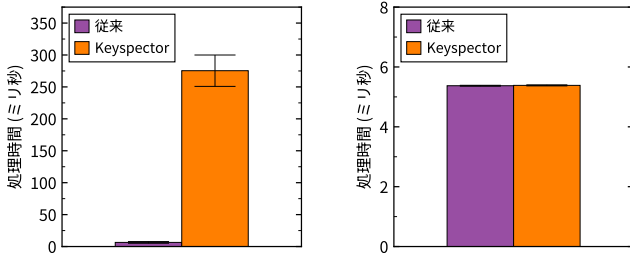
次に、IDSのプログラムをわずかに変更してエンクレイヴ内で実行した。このIDSのハッシュ値は事前に登録されているものと異なるため、IDSがシステムメモリにアクセスしようとした際にPMPのアクセスフォールトが発生し、アクセスできなかった。この結果より、侵入者がエンクレイヴを作成して不正なプログラムを実行しても、システムメモリ上の情報は盗聴できないことを確認した。

さらに、監視対象システムにおいてIDSの実行を妨害し、セキュリティモニタがそれを検知できるかどうかを確認した。まず、IDSが動作するエンクレイヴを制御しているプロセスを終了させた。この場合、IDSの終了に伴ってIDSからセキュリティモニタへの定期的な通知が途切れ、セキュリティモニタはIDSの停止を検知した。次に、大量のプロセッサ間割り込み (IPI) を発生させることで、IDSの実行が頻繁に中断されるようにした。IPIが発生すると、セキュリティモニタはエンクレイヴの実行を一時的に停止し、監視対象システムへ制御を移す。その結果、IDSからセキュリティモニタへの通知の間隔が通常より長くなり、セキュリティモニタはIDSの実行が妨害されていることを検知した。

6.2 IDS 認証のオーバーヘッド

リモートアテステーションを用いてIDSのハッシュ値をチェックするのにかかるオーバーヘッドを測定した。通常のリモートアテステーションの処理に加えて、Keyspectorでは検証サーバでハッシュ値を比較してIDS認証情報を発行し、セキュリティモニタに登録する処理が行われる。この実験では1Gbpsで接続された同一LAN内の検証サーバを用いた。リモートアテステーションを開始してからセキュリティモニタでの検証が完了するまでの時間を10回測定した平均値と標準偏差を図4(a)に示す。Keyspectorの実行時間は従来と比べて42倍になった。これはIDS認証情報をセキュリティモニタに登録する際に、SHA-256ハッシュ計算およびECDSA P-256署名検証が行われるため、それらの処理に伴うオーバーヘッドが大きいことが原因だと考えられる。

次に、セキュリティモニタにおいてIDSのハッシュ値を



(a) リモートアテステーション (b) エンクレイブの初期化

図 4 IDS 認証のオーバーヘッド

Fig. 4 The overhead of IDS authentication.

チェックするのにかかるオーバーヘッドを測定した。エンクレイブ初期化時には、セキュリティモニタにおいてエンクレイブのハッシュ値が計算される。Keyspector ではそれに加えて、計算したエンクレイブのハッシュ値とセキュリティモニタに登録済みのハッシュ値との比較が行われる。監視対象システムでエンクレイブを起動してから IDS の実行が開始されるまでの時間を 10 回測定した平均値と標準偏差を図 4(b) に示す。Keyspector と従来の実行時間にはほとんど差がなく、ハッシュ値の比較によるオーバーヘッドは無視できる程度であった。

6.3 システム情報の取得時間

Keyspector を用いてエンクレイブ内で IDS を実行し、システム情報の取得に要する時間を測定した。監視対象システム内で動作していたプロセス (カーネルスレッドを含む) は IDS を実行するためのものも含めて 110 個であった。システム情報の取得時間を 10 回測定した平均値と標準偏差を図 5 に示す。Keyspector によるシステム情報の取得時間は、監視対象システム内でシステム情報を取得する従来手法と比較して、ほぼ同等であった。

Keyspector におけるシステム情報の取得時間が従来手法とほぼ同等となった原因を調査するため、従来手法におけるファイルシステムアクセスのオーバーヘッドを測定した。その結果、ファイルシステムのアクセス時間は、システム情報の取得時間全体の 32% を占めることがわかった。従来手法では、カーネル内部での情報取得は高速である一方、ファイルシステムを介したアクセスにオーバーヘッドが生じる。このオーバーヘッドが、Keyspector におけるソフトウェアによるアドレス変換のオーバーヘッドを相殺したため、両手法の取得時間がほぼ同等になったと考えられる。

6.4 システム性能への影響

IDS の実行がアプリケーション性能に与える影響を調べるために、監視対象システム上で CoreMark-PRO ベンチマークを実行した。CoreMark-PRO は、アプリケーションレベルのワークロードを用いて IoT 機器の性能を評価するために設計されたベンチマークスイートであり、画像圧縮、

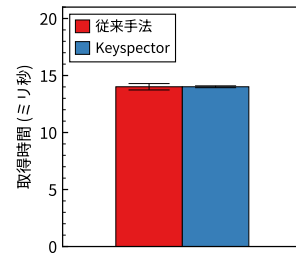


図 5 システム情報の取得時間

Fig. 5 The collection time of system information.

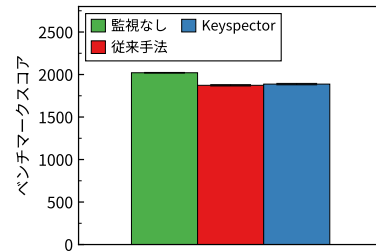


図 6 システム性能への影響

Fig. 6 The impact on system performance.

ハッシュ計算、ニューラルネットワーク推論などのワークロードを含む。

Keyspector と従来手法の IDS をそれぞれ 100ms 間隔で定期的に行うながら、CoreMark-PRO の各ワークロードを実行した。ベンチマークの総合スコアを 10 回測定した平均値と標準偏差を図 6 に示す。これより、Keyspector を用いた場合のアプリケーション性能は従来手法を用いた場合とほぼ同等であり、Keyspector による監視が監視対象システムの性能に与える影響は、従来手法と同程度であった。

7. 関連研究

TrustZone を用いてセキュアワールド内で IDS を安全に実行可能な LKRDet [13] や Trusted Monitor [14] がある。LKRDet は CPU の性能カウンタを用いてノーマルワールド上のカーネルルートキットを検知する。そのために、特定のシステムコールの実行中に発生するハードウェアイベントの値を計測し、それらをプロファイリング段階で記録した値と比較する。LKRDet では、システムコールをトラップし、性能カウンタを操作するために Trusted Application を呼び出す必要があるため、監視に伴うオーバーヘッドが大きい。

Trusted Monitor も性能カウンタを用いてノーマルワールドの異常検知を行う IDS である。この IDS は性能カウンタの値を収集し、正常にシステムが動作している時の性能カウンタの値を用いて学習させた機械学習モデルを用いて検知を行う。ノーマルワールドで動作するエージェントが IDS を定期的呼び出し、一定時間以上呼び出されない場合はウォッチドッグタイマによってシステムがリセット

される。Trusted Monitor の実行オーバヘッドは小さい一方で、検知精度は十分に高いとはいえない。

SGXを用いたネットワークベースIDSとしてS-NFV [15]やSEC-IDS [16]がある。S-NFVはオープンソースのIDSであるSnort [17]の内部状態をエンクレイヴ内に安全に保持する。エンクレイヴ内に置いた内部状態を、外部からの攻撃を受けない安全なAPI経由で利用することで、ネットワークフローごとの情報などの盗聴を防ぐ。SEC-IDSはSnortをほとんど修正なしにエンクレイヴ内で実行する。エンクレイヴ内で既存のIDSを動作させるために、Graphene-SGXライブラリOS [18]を用いる。Snortはエンクレイヴ内で安全に実行できるが、Snortがパケットを取得するまでの間にパケットを書き換えられると、攻撃を正しく検知できない可能性がある。

8. まとめ

本稿では、RISC-VのTEEであるエンクレイヴと特権ソフトウェアのセキュリティモニタを連携させることでIDSを安全に実行するKeyspectorを提案した。Keyspectorはセキュリティモニタにおいてアクセス権限の設定を変更することにより、IDSが動作するエンクレイヴだけが監視対象システムのメモリに直接アクセスしてOSデータを取得することを可能にする。さらに、セキュリティモニタはIDSの実行状態を監視し、攻撃者によるIDSの実行妨害を検知することができる。実験結果から、エンクレイヴ内でIDSを実行することによるオーバヘッドは無視できるほど小さいことを確認した。

今後の課題は、複数のIDSを実行した際のシステム性能への影響を調べることである。また、I/Oデバイスからのメモリアクセスを制御するハードウェア機構であるIOPMP [12]を用いて、DMAを悪用したエンクレイヴのメモリに対する攻撃を防ぐ必要がある。

謝辞 本研究は、JST 経済安全保障重要技術育成プログラム【JPMJJP24U4】の支援を受けたものである。

参考文献

- [1] Statista: Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2034, , available from <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>
- [2] Zscaler Inc: Zscaler ThreatLabz 2024 Mobile, IoT, and OT Threat Report, , available from <https://www.zscaler.com/campaign/threatlabz-mobile-iot-ot-report>
- [3] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K. and Zhou, Y.: Understanding the Mirai Botnet, *Proc. 26th USENIX Security Symp.*, pp. 1093–1110 (2017).

- [4] Alrawi, O., Lever, C., Valakuzhy, K., Court, R., Snow, K., Monrose, F. and Antonakakis, M.: The Circle Of Life: a Large-Scale Study of The IoT Malware Lifecycle, *Proc. 30th USENIX Security Symp.*, pp. 3505–3522 (2021).
- [5] Guerra, M., Taubmann, B., Reiser, H. P., Yalaw, S. and Correia, M.: Introspection for ARM TrustZone with the ITZ Library, *Proc. 18th IEEE Int. Conf. on Software Quality, Reliability and Security*, pp. 123–134 (2018).
- [6] Koga, Y. and Kourai, K.: SSdetector: Secure and Manageable Host-based IDS with SGX and SMM, *Proc. 22nd IEEE Int. Conf. on Trust, Security and Privacy in Computing and Commun.*, pp. 539–548 (2023).
- [7] Lee, D., Kohlbrenner, D., Shinde, S., Asanovic, K. and Song, D.: Keystone: An Open Framework for Architecting Trusted Execution Environments, *Proc. 15th European Conf. Computer Systems* (2020).
- [8] Ozaki, Y., Kanamoto, S., Yamamoto, H. and Kourai, K.: Reliable and Accurate Fault Detection with GPG-PU and LLVM, *Proc. 16th IEEE Int. Conf. on Cloud Computing*, pp. 540–546 (2023).
- [9] Breitenbacher, D., Homoliak, I., Aung, Y. L., Tippenhauer, N. O. and Elovici, Y.: HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices, *Proc. 14th ACM Asia Conf. on Computer and Commun. Security*, pp. 479–484 (2019).
- [10] Zhang, F., Leach, K., Sun, K. and Stavrou, A.: SPECTRE: A Dependable Introspection Framework via System Management Mode, *Proc. 43rd Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, pp. 1–12 (2013).
- [11] Jang, D., Lee, H., Kim, M., Kim, D., Kim, D. and Kang, B. B.: ATRA: Address Translation Redirection Attack against Hardware-based External Monitors, *Proc. 21st ACM SIGSAC Conf. on Computer and Commun. Security*, pp. 167–178 (2014).
- [12] RISC-V International: RISC-V IOPMP Architecture Specification v0.8.3 (2026).
- [13] Jiang, X., Lora, M. and Chattopadhyay, S.: Efficient and Trusted Detection of Rootkit in IoT Devices via Offline Profiling and Online Monitoring, *Proc. 30th ACM Great Lakes Symp. on VLSI*, pp. 433–438 (2020).
- [14] Eichler, C., Röckl, J., Jung, B., Schlenk, R., Müller, T. and Hönig, T.: Profiling with Trust: System Monitoring from Trusted Execution Environments, *Design Automation for Embedded Systems*, Vol. 28, No. 1, pp. 23–44 (2024).
- [15] Shih, M., Kumar, M., Kim, T. and Gavrilovska, A.: S-NFV: Securing NFV States by Using SGX, *Proc. ACM Int. Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 45–48 (2016).
- [16] Kuvaiskii, D., Chakrabarti, S. and Vij, M.: Snort Intrusion Detection System with Intel Software Guard Extension (Intel SGX), *arXiv:1802.00508* (2018).
- [17] Roesch, M.: Snort – Lightweight Intrusion Detection for Networks, *Proc. 13th USENIX Conf. on System Administration*, pp. 229–238 (1999).
- [18] Tsai, C., Porter, D. E. and Vij, M.: Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX, *Proc. USENIX Annual Technical Conf.*, pp. 645–658 (2017).