Intel SGXを用いたVMのメモリとディスクの安全な監視

中野 智晴[1,a)]　光来 健一[1,b)]

概要：クラウドにおいて仮想マシン（VM）を安全に監視するために，侵入検知システム（IDS）を VM の外側で実行する IDS オフロードが提案されている．しかし，IDS オフロードを用いてもクラウド内の信頼できない管理者や外部の攻撃者によってオフロードした IDS が攻撃される恐れがある．これまでに提案されてきた手法では，クラウド内で高度な IDS を安全に実行し，かつ，システム性能への影響を小さくするのは難しかった．そこで本稿では，Intel SGX を用いてクラウド内で IDS を安全かつ軽量に実行するためのシステム SGmonitor を提案する．SGmonitor はエンクレイヴと呼ばれる保護領域内で IDS を動作させることにより，IDS への様々な攻撃を防ぐことを可能にする．エンクレイヴはアプリケーションの一部であるため，高度な IDS の開発が行いやすく，システム性能に及ぼす影響も小さい．エンクレイヴ内の IDS は VM のメモリやディスク上のデータを安全に取得することにより，VM の監視を行う．我々は SGX をサポートした Xen に SGmonitor を実装し，オフロードした IDS の性能について調べた．

キーワード：Intel SGX，VM，IDS，クラウド

Secure Monitoring of VM's Memory and Disk Using Intel SGX

Tomoharu Nakano[1,a)]　Kenichi Kourai[1,b)]

Abstract: To monitor virtual machines (VMs) in clouds securely, IDS offloading has been proposed for executing intrusion detection systems (IDSes) outsides VMs. However, even if IDS offloading is used, offloaded IDSes can be attacked by untrusted cloud administrators and external attackers. In previously proposed systems, it is difficult to execute advanced IDSes inside clouds securely and suppress the impact on system performance. In this paper, we propose SGmonitor for enabling users to execute IDSes inside clouds more securely in a lightweight manner using Intel SGX. SGmonitor executes IDSes in protected memory regions called enclaves so that it can prevent various attacks against IDSes. Since an enclave is part of an application, SGmonitor makes it easier to develop advanced IDSes and does not degrade system performance. IDSes in enclaves monitor VMs by obtaining data in the memory and disk securely. We have implemented SGmonitor in Xen with SGX support and examined the performance of offloaded IDSes in SGmonitor.

Keywords: Intel SGX, VMs, IDSes, clouds

1. はじめに

近年，ユーザに仮想マシン（VM）を提供する IaaS 型クラウドの普及が進んでいる．IaaS 型クラウドでは，ユーザは必要に応じて様々な OS やソフトウェアを VM 内にインストールすることができる．一方で，クラウド内の VM はインターネットを経由して攻撃を受けやすい．VM が攻撃された場合，VM 内の機密情報が盗まれる恐れがあるため，侵入検知システム（IDS）を用いて VM を監視することがますます重要となっている．ホスト型 IDS は監視対象の VM 内で動作させるのが一般的であるが，VM に侵入した攻撃者に無力化される危険性がある．そこで，監視対象 VM の外側で IDS を安全に実行する IDS オフロードと呼ばれる手法が提案されている [1]．IDS オフロードを行うことによって，VM に侵入したとしても攻撃者が IDS を無

[1] 九州工業大学
Kyushu Institute of Technology
[a)] naka_tomo@ksl.ci.kyutech.ac.jp
[b)] kourai@ksl.ci.kyutech.ac.jp

効化することはできなくなる．

しかし，クラウドの管理者は必ずしも信頼できるとは限らず，信頼できない管理者からオフロードした IDS への攻撃が行われる危険性がある．また，IDS に対してクラウド外部から攻撃が行われる可能性もある．その結果，オフロードした IDS が取得した VM 内の機密情報を管理者や攻撃者に盗まれる恐れがある．これまでに，オフロードした IDS を保護することのできる様々なシステムが提案されてきた [2–10] が，クラウド内で高度な IDS を安全に実行でき，かつ，システム性能に対する影響を小さくするのは難しかった．

本稿では，Intel SGX を用いてクラウド内で IDS を安全かつ軽量に実行するシステム SGmonitor を提案する．SGX は，エンクレイヴと呼ばれる保護領域内でプログラムを安全に実行するための CPU 機構である．エンクレイヴ内で IDS を実行することにより，クラウド内でも IDS の改ざんや IDS からの情報漏洩を防ぐことができる．エンクレイヴは通常のアプリケーションの一部であるため，高度な IDS の開発が行いやすく，システム全体の性能に及ぼす影響も小さい．ただし，攻撃者が IDS の実行を停止することは防げないため，クラウド外部にある VM ユーザのホストからハートビートを送ることで IDS の正常な動作を確認する．

我々は SGX をサポートした Xen [11] を用いて SGmonitor を実装した．SGmonitor は，SGX 仮想化を有効にした VM 内で IDS を SGX アプリケーションとして動作させる．IDS が VM のメモリ上の OS データを取得する際には，信頼できるハイパーバイザ内でデータの暗号化を行い，エンクレイヴ内で復号することにより情報漏洩を防ぐ．LLView フレームワーク [12] を用いることにより，IDS が透過的に OS データを取得できるようにする．また，エンクレイヴ内で軽量なファイルシステムを動作させることで，VM の暗号化ディスク上のデータを安全に取得する．SGmonitor を用いて実験を行い，エンクレイヴ内で実行した IDS が監視対象 VM 内のメモリやディスク上のデータを取得して侵入検知ができることを確認した．また，SGmonitor を用いてオフロードした IDS と従来手法を用いてオフロードした IDS の実行時間を測定し，SGmonitor によって生じるオーバヘッドを調べた．

以下，2 章でクラウドにおける IDS オフロードとその問題点について述べる．3 章で Intel SGX を用いて IDS を保護することにより，VM を安全に監視できるようにするシステム SGmonitor を提案する．4 章で SGmonitor の実装について述べる．5 章で SGmonitor の性能を調べた実験について述べる．6 章で関連研究に触れ，7 章で本稿をまとめる．

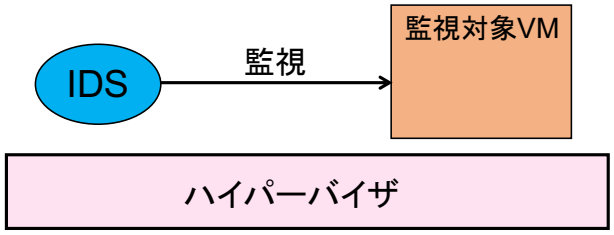

図 1　IDS オフロード

2. クラウドにおける IDS オフロード

IDS を安全に実行する手法として，VM を用いた IDS オフロードが提案されている [1]．この手法は図 1 のように，IDS を監視対象 VM の外側で実行し，VM 内部のシステムの監視を行う手法である．IDS オフロードを用いることにより，監視対象 VM に侵入されたとしても VM 内で IDS は動作していないため，IDS が無効化される恐れはない．オフロードした IDS は監視対象 VM のメモリを解析して OS が管理しているデータを取得したり，VM の仮想ディスク上のファイルを読み込んだりする．それにより，監視対象 VM 内で動作させた場合と同様にシステムの監視を行い，攻撃を検知することができる．例えば，VM 内で実行されているプロセスの一覧を取得することにより，不正なプログラムが実行されていないかをチェックすることができる．

しかし，IDS オフロードを行ったとしても，まだ IDS が攻撃を受ける可能性がある．オフロードした IDS はクラウド管理者の管理下におかれるが，クラウドの管理者は必ずしも信頼できるとは限らない．実際に，Google の管理者がユーザの機密情報を閲覧し，プライバシを侵害した事件が発生している [13]．また，サイバー犯罪の 28%は内部犯行であるという調査結果 [14] や，管理者の 35%は機密情報をのぞき見したことがあるという調査結果 [15] もある．そのため，悪意のある管理者によってオフロードした IDS への攻撃が行われる危険性がある．また，クラウド外部の攻撃者がクラウドに侵入してオフロードした IDS への攻撃を行う危険性もある．その結果，オフロードした IDS が無力化させられたり，IDS が取得した VM 内の機密情報を盗まれたりする恐れがある．

これらの問題を解決するために，VM の下で動作するハイパーバイザを信頼して，オフロードした IDS を安全に実行する手法が提案されてきた．例えば，ハイパーバイザ内に IDS をオフロードし，VM の監視を行うシステム BVMD [2] が提案されている．しかし，ハイパーバイザは低レベルな処理しか行うことができないため，ハイパーバイザ内で高度な IDS を動作させるのは難しい．Self-Service Cloud （SSC）[3] を用いることで，サービスドメインと呼ばれる VM の中で IDS を安全に実行することもできる．

表 1　従来手法の比較

	IDS の安全性	IDS の機能	システムへの影響	クラウド内で実行
BVMD	○	×	○	○
SSC	△	○	○	○
Remote Trans	○	○	○	×
Hyper Guard	○	×	×	○
V-Met	○	○	×	○

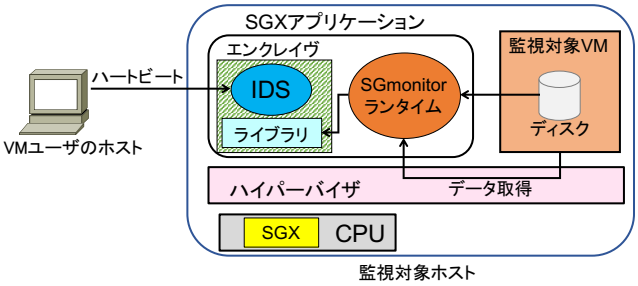

図 2　SGmonitor のシステム構成

サービスドメインへのアクセスは制限されているが，その中では通常のシステムが動作しているため，システムに脆弱性があった場合，IDS が攻撃を受ける危険性がある．Copilot [4] や HyperCheck [5]，RemoteTrans [6] はクラウド外部の信頼できるホストで IDS を実行し，監視を行うことができる．ただし，これらの手法ではクラウド内で IDS を実行できず，クラウド外部に別のホストを用意して IDS を実行する必要がある．

クラウド内のハイパーバイザを信頼せず，仮想化システムの外側で IDS を安全に動作させる手法も提案されている．HyperGuard [7] と HyperSentry [8] は CPU のシステムマネジメントモード（SMM）を用いて IDS を安全に実行する．Flicker [9] は AMD SVM や Intel TXT を用いて IDS を安全に実行することができる．しかし，SMM や SVM/TXT を用いて IDS を実行する場合，実行中は仮想化システムが停止してしまう．一方，V-Met [10] はネストした仮想化を用いて，仮想化システムを VM 内で動作させ，その外側で IDS を安全に実行する．しかし，ネストした仮想化のオーバヘッドのため，仮想化システムの性能が低下する．

表 1 に従来手法の比較を示す．いずれの手法でも，クラウド内で高機能な IDS を安全に実行し，かつ，システム性能への影響を小さくするのは難しい．

3. SGmonitor

本稿では，Intel SGX を用いてクラウド内の IDS を保護することにより，VM を安全に監視できるようにするシステム SGmonitor を提案する．Intel SGX は，エンクレイヴと呼ばれる保護領域を用いてプログラムを安全に実行することを可能にする CPU 機構である．エンクレイヴ内で IDS を実行することにより，クラウド内の IDS を以下のように安全に実行することができる．エンクレイヴで IDS の実行を開始する際には SGX によってコードの電子署名が検査されるため，攻撃者が改ざんした IDS を実行することはできない．悪意ある管理者によって署名された IDS が実行された場合でも，リモートアテステーションにより信頼できる第三者機関でチェックを行うことができる．また，SGX によってエンクレイヴのメモリの整合性が保証されるため，実行中に IDS を改ざんすることはできない．それに加えて，エンクレイヴのメモリは暗号化されるため，IDS が取得した監視対象 VM 内の情報が漏洩することもない．

SGmonitor のシステム構成を図 2 に示す．SGmonitor では，IDS は SGX アプリケーションとして作成され，従来の IDS オフロードと同様に監視対象 VM が動作しているハイパーバイザ上で実行される．SGX アプリケーションはエンクレイヴと SGmonitor ランタイムで構成される．エンクレイヴ内の IDS は SGmonitor ライブラリを用いて SGmonitor ランタイム経由でハイパーバイザとの通信を行い，VM のメモリ上の OS データを取得して VM 内のシステムを監視する．また，IDS は SGmonitor ライブラリが提供するファイルシステムを用いて，SGmonitor ランタイム経由で VM の仮想ディスク上のファイルを監視する．SGX では攻撃者が IDS の実行を停止することは防げないため，クラウド外部にある VM ユーザのホストからハートビートを送ることで，IDS が正常に動作していることを確認する．

SGmonitor では以下のような脅威モデルを考える．まず，クラウドプロバイダは信頼し，プロバイダが提供しているクラウド内のハードウェアも信頼できるものとする．クラウドプロバイダは一度信用を失うと致命的であるため，この仮定は妥当であると考えられ，様々な研究で広く用いられている [16] [17] [18] [3] [6] [10]．また，クラウド内のハイパーバイザも信頼する．ハイパーバイザが信頼できる状態にあることは様々な手法で確認できる．例えば，TPM を用いたリモートアテステーションによりハイパーバイザの正常起動を確認することができる．また SMM などのハードウェア機構を用いることにより，実行中にハイパーバイザの改ざんを検出することもできる [7] [5] [8] [9]．一方で，オフロードした IDS を動作させる OS などの実行環境や SGX アプリケーション内の SGmonitor ランタイムは信頼しない．本稿では，クラウド内の信頼できない管理者や外部の攻撃者が IDS への攻撃を行う状況を想定する．また，悪意のある管理者が作成した IDS によって，VM 内の機密情報が盗まれる攻撃も想定する．

エンクレイヴはアプリケーションの一部であるため，高度な IDS を比較的容易に開発することができる．SGmonitor

では，エンクレイヴ内で動作する IDS は LLView [12] と呼ばれるフレームワークを用いて OS のカーネルモジュールのように開発することが可能である．開発者は OS カーネルの構造体やマクロ，インライン関数などを利用することができる．また，VM 内の OS データを取得するたびに SGmonitor ランタイムを呼び出す必要があるが，SGmonitor では透過的に呼び出しが行われるようにプログラム変換を行う．これにより，IDS の開発者はオフロードを意識することなく IDS を開発することができる．

SGX アプリケーションは OS 上の一つのアプリケーションであるため，仮想化システムの性能に影響を与えることはない．SMM や SVM/TXT のように，IDS の実行中に仮想化システムを停止させる必要はなく，VM を実行したまま監視を行うことができる．また，ネストした仮想化を用いる場合のように，仮想化システム全体に常にオーバヘッドがかかることもない．

4. 実装

我々は SGX をサポートした Xen 4.7 を用いて SGmonitor を実装した．

4.1 Intel SGX

SGX を用いることにより，アプリケーションはメモリ上にエンクレイヴを作成することができる．エンクレイヴのメモリにはエンクレイヴ内のコードだけがアクセスでき，エンクレイヴで実行されるコードはロード時に電子署名が検査される．SGX アプリケーションは，信頼できるエンクレイヴとエンクレイヴの外で動作する信頼できないコードで構成される．SGX には信頼できないコードからエンクレイヴを安全に呼び出す ECALL と，エンクレイヴから信頼できないコードを呼び出す OCALL と呼ばれる機構が用意されている．また，SGX の SDK [19] には ECALL と OCALL のインタフェースを定義するための EDL と呼ばれる言語が用意されている．

4.2 Xen-SGX

Xen-SGX [11] は SGX 仮想化をサポートした仮想化ソフトウェアである．SGX を使用するにはエンクレイヴ・ページキャッシュ（EPC）と呼ばれるエンクレイヴのためのセキュアな記憶域が必要になる．EPC はハードウェアによって提供される資源であり，我々の実行環境ではそのサイズは 93MB であった．通常，EPC は BIOS によって予約され，SGX ドライバをインストールした OS によって管理される．Xen-SGX では，ハイパーバイザに EPC を管理する機能を追加することにより，完全仮想化 VM 内で SGX を使用することを可能にする．Xen-SGX では VM を立ち上げる際に，その VM に割り当てる EPC のサイズを決めることができる．SGmonitor では，IDS を実行するための VM として，EPC を割り当てた VM を作成し，その VM を IDS VM と呼ぶ．

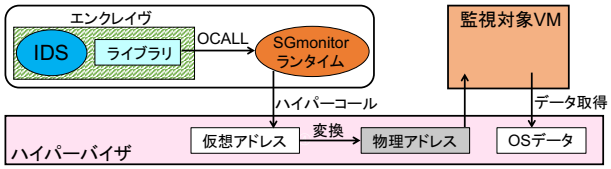

図 3　VM 内の OS データ取得

4.3 VM 内の OS データの取得

エンクレイヴ内の IDS は監視対象 VM のメモリ上の OS データを取得するために，図 3 のように，まず OCALL を用いてエンクレイヴ外部の SGmonitor ランタイムを呼び出す．これはエンクレイヴが直接ハイパーバイザを呼び出すことができないためである．そして，SGmonitor ランタイムはハイパーコールを用いてハイパーバイザを呼び出す．ハイパーバイザは仮想 CPU の CR3 レジスタが指している VM 内のページテーブルを参照して，取得しようとしているデータの仮想アドレスを対応する物理アドレスに変換する．その物理アドレスを用いて VM のメモリ上にある OS データをページ単位で取得する．

OS データをエンクレイヴに返す際に，SGmonitor では図 4 のように，取得した OS データをハイパーバイザ内で暗号化する．これにより，信頼できない SGmonitor ランタイムでの情報漏洩を防ぐ．暗号化された OS データが SGmonitor ランタイムを介してエンクレイヴに返されると，SGmonitor ライブラリがそれを復号して IDS に渡す．データの暗号化・復号化を行うために，エンクレイヴとハイパーバイザに wolfSSL [20] の AES 関数を移植した．エンクレイヴ内で CPU の AES 支援機構である AES-NI を利用するには，AES-NI が利用可能かどうかを調べる CPUID 命令を使わないようにする必要があった．また，ハイパーバイザ内で AES-NI を利用できるようにするために，UVBond [21] の暗号化機構を移植した．

SGmonitor ライブラリは取得した OS データをハッシュ表を用いてキャッシュする．これにより，IDS が同じデータを必要とした時には再度データを取得する必要がなくなる．また，OS データはページ単位で取得するため，同一ページ上の別のデータにアクセスする際にも OCALL やハイパーコールを呼び出さずにアクセスすることができる．OS データをキャッシュすることにより最新のデータが得られない可能性があるため，定期的にキャッシュ内のデータを消去する必要がある．

4.4 VM の仮想ディスクの監視

SGmonitor では，IDS が VM の仮想ディスクを安全に監視できるようにするために，図 5 のようにエンクレイヴ

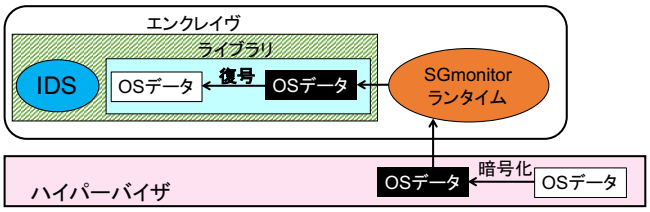

図 4　OS データの暗号化

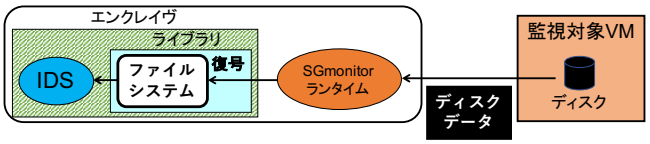

図 5　VM のディスク監視

内で軽量なファイルシステムを動作させる．エンクレイヴは直接 VM の仮想ディスクにアクセスすることはできないため，ブロックレベルのアクセスを行う際に OCALL を用いて SGmonitor ランタイムを呼び出す．システムコールレベルで SGmonitor ランタイムを呼び出して OS のファイルシステムを利用することも考えられるが，SGmonitor ではこのアプローチは採用していない．これは，信頼できない SGmonitor ランタイムや OS への情報漏洩を防ぐために VM の仮想ディスクをブロックレベルで暗号化できるようにするためである．

エンクレイヴ内にファイルシステムを実装するために，Xvisor [22] と呼ばれる組込み機器向けの軽量な仮想化ソフトウェアの VFS と ext4 ファイルシステムを移植した．ディスク監視に必要な機能として，ファイルシステムのマウント，ファイルのオープン，情報取得，読み込みなどだけを実装した．これらの機能が仮想ディスクにアクセスする際には，そのオフセットとサイズを指定して OCALL で SGmonitor ランタイムを呼び出す．SGmonitor ランタイムは，kpartx コマンドを用いて作成した仮想ディスクに対応するデバイスマップからデータを読み込む．仮想ディスクイメージは管理 VM 内にあるため，NFS を用いて IDS VM と共有する．

SGmonitor ではランタイムや OS への情報漏洩を防ぐために，VM の仮想ディスクは暗号化されている．そのため，OCALL を用いて SGmonitor ランタイム経由で取得した仮想ディスク上のデータはエンクレイヴ内の SGmonitor ライブラリが復号する．VM が暗号化ディスクにアクセスする際には，UVBond [21] のディスク暗号化機構を用いてハイパーバイザがデータの暗号化・復号化を行う．現在のところ，暗号化ディスクの使用については未実装である．

4.5　IDS の開発

SGmonitor では，LLView フレームワーク [12] を用いることで Linux カーネルのヘッダファイルを利用して OS

```
#include <linux/sched.h>

int ids(void) {
  struct task_struct *p = &init_task;

  do {
    if (strcmp(p->comm, "XXX")==0)
      :
    p= list_entry(p->tasks.next,
                  struct task_struct,  tasks);
  } while (p != &init_task);
}
```

図 6　プロセス一覧を取得して検知する IDS

データを監視する IDS を開発することができる．例えば，監視対象 VM 内で動作しているプロセス一覧を取得して侵入検知を行う IDS は，図 6 のように sched.h をインクルードして task_struct 構造体からプロセス情報を取得する．その際にプログラム変換を行うことにより，明示的に OCALL を呼び出すことなく，透過的に VM 内の OS データを取得する．LLView は IDS をコンパイルする際に生成された LLVM の中間表現を変換し，OS データにアクセスしようとした時に IDS に VM 内の OS データを取得させる．そのために，メモリからデータを読み込むために用いられる load 命令の直前に OS データを取得する関数を挿入し，取得したデータが置かれているエンクレイヴのメモリに対して load 命令が実行されるようにする．

4.6　開発した IDS

LLView を用いて，エンクレイヴ内で動作する IDS を作成した．この IDS は，監視対象 VM 内のメモリ上の OS データを解析することでプロセスやカーネルモジュール，TCP・UDP 通信の情報を取得し，侵入検知を行う．例えば，プロセスやカーネルモジュールの一覧を取得し，プロセス名やモジュール名を比較することで，不正なプロセスやカーネルモジュールを検知する．また，TCP 通信や UDP 通信の一覧から使用しているポート番号を比較し，不正な通信を検知する．さらに，監視対象 VM のディスク上からファイルの情報を取得し，不正なファイルやディレクトリなどに基づいて侵入検知を行う．

4.7　暗号鍵の共有

SGmonitor では公開鍵暗号を用いて，エンクレイヴとハイパーバイザ間でデータを暗号化・復号化するための暗号鍵を共有する．まずエンクレイヴ内の SGmonitor ライブラリが暗号鍵を生成し，それをハイパーバイザの公開鍵で暗号化する．ハイパーバイザの公開鍵はあらかじめ，SGmonitor ライブラリに埋め込んでおく．暗号化された暗号鍵は SGmonitor ランタイム経由でハイパーバイザに渡され，ハイパーバイザは自身の秘密鍵を用いて暗号鍵を復号する．秘密鍵はハイパーバイザだけが知っているため，

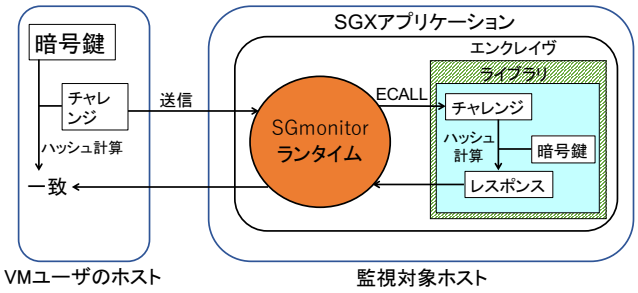

図 7　ハートビート

ハイパーバイザ以外は暗号鍵を復号することができない．

暗号鍵を登録する際に，IDS はクラウド外部の信頼できる第三者機関と通信して暗号化された暗号鍵の電子署名を取得する．そして，暗号鍵と一緒にその電子署名をハイパーバイザに渡し，ハイパーバイザにおいて検証を行う．第三者機関は電子署名を行う際に，エンクレイヴのリモートアテステーションを通して正規の IDS であることを確認する．正規の IDS に対してのみ電子署名を行うようにすることにより，正規の IDS だけがハイパーバイザに暗号鍵を登録することができる．現在のところ，電子署名の検証については未実装である．

また，IDS が正常に動作していることを確認するためのハートビートを送るために，IDS と VM ユーザのホスト間でも暗号鍵を共有する．その場合も上記と同様の方法で暗号鍵の登録を行う．

4.8　ハートビート

SGmonitor ではチャレンジ・レスポンス方式を用いて，VM ユーザのホストから IDS に定期的にハートビートを送信する．図 7 のように，まずチャレンジとして乱数を SGmonitor ランタイムに送信し，ECALL を用いてエンクレイヴ内のライブラリ関数を呼び出す．SGmonitor ライブラリは VM ユーザのホストと共有している暗号鍵と受信したチャレンジからハッシュ値を計算した後，それをレスポンスとして SGmonitor ランタイム経由で VM ユーザのホストに返す．VM ユーザのホストでも同様に暗号鍵とチャレンジからハッシュ値を計算し，それがレスポンスと一致すれば IDS の正常な動作を確認できる．ハッシュ値計算に暗号鍵が含まれるため，正規の IDS 以外は正しいレスポンスを返すことができない．現在のところ，ハートビートは未実装である．

5.　実験

SGmonitor を用いて，侵入検知を行う IDS が正常に動作するかを確認する実験を行った．また，その性能を調べる実験を行い，従来の IDS オフロードの性能と比較した．実験には，Intel Xeon E3-1225 v5 の CPU，8GB のメモリを搭載したマシンを使用し，仮想化システムには

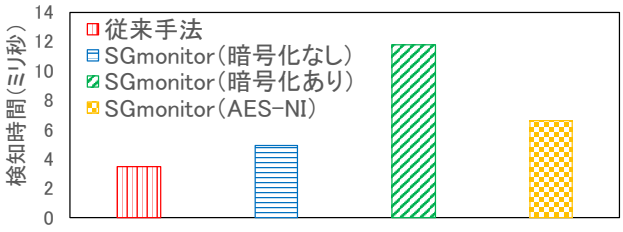

図 8　IDS の検知時間（メモリ）

Xen-SGX 4.7 を使用した．監視対象 VM には 2 個の仮想 CPU と 2GB のメモリを割り当て，OS には Linux 4.4 を使用した．IDS VM には 2 個の仮想 CPU と 2GB のメモリを割り当て，93MB の EPC を割り当てた．また，OS には Linux 4.4 を使用した．

5.1　IDS の動作確認

SGmonitor を用いて実装した IDS が VM のメモリ上の OS データを取得し，侵入検知を行うことができるかどうかを確認した．動作確認のために，監視対象 VM では検知対象となる kworkerds という名前のプロセスを実行し，adore という名前のカーネルモジュールをインストールした．加えて，2001 番ポートへのネットワーク接続を確立した．実行結果から，エンクレイヴ内の IDS が監視対象 VM のメモリからプロセス名，モジュール名，使用中のポート番号を取得することで，侵入検知を正しく行えることが確認できた．

次に，実装した IDS が VM の仮想ディスク上のファイルのデータを取得し，侵入検知を行えるかどうかを確認した．動作確認のために，監視対象 VM 内に Rocke Monero Miner マルウェアが用いる/etc/xig ファイルを作成した．実行結果から，IDS がエンクレイヴ内のファイルシステムを用いて監視対象 VM の仮想ディスク上に特定のファイルがあるかどうかをチェックし，侵入検知を正しく行えることが確認できた．

5.2　IDS の検知時間（メモリ）

VM のメモリから OS データを取得して侵入検知を行う性能を調べた．そのために，IDS が VM 内のメモリから OS データを取得して，5.1 節の不正なプロセスや不正なカーネルモジュール，不正な通信を検知するのにかかる時間を測定した．暗号化のオーバヘッドを調べるために，OS データの暗号化を行う場合と行わない場合についてそれぞれ測定を行った．暗号化を行う場合には，AES-NI を使用する場合としない場合についてもそれぞれ測定を行った．比較として，従来手法を用いて管理 VM にオフロードした IDS を実行した場合の検知時間も測定した．IDS の検知時間を 10 回測定した時の平均値を図 8 に示す．

SGmonitor において OS データの暗号化を行わない場合，検知時間は従来手法と比較して 1.4 倍に増加した．こ

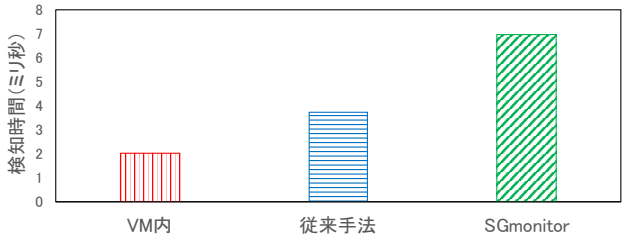

図 9　IDS の検知時間（ディスク）

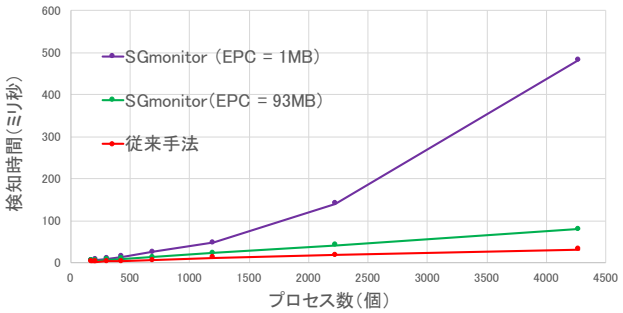

図 10　プロセス数と EPC サイズの影響

れはエンクレイヴから OCALL を用いてハイパーバイザ経由で OS データを取得することによるオーバヘッドが原因と考えられる．一方，OS データの暗号化を行う場合，従来手法と比べて 3.4 倍の時間がかかったが，AES-NI を用いることによって 1.9 倍に抑えることができた．いずれにしても検知時間は十分に短かった．

5.3　IDS の検知時間（ディスク）

VM の仮想ディスク上からファイルデータを取得して侵入検知を行う性能を調べるために，5.1 節の不正なファイルを検知するのにかかる時間を測定した．暗号ディスクへの対応は未実装のため，暗号化を行う場合については測定していない．比較として，管理 VM にオフロードした IDS を実行した場合の検知時間も測定した．また，監視対象 VM 内で検知を行った際の検知時間も測定した．それぞれの検知時間を 10 回測定した時の平均値を図 9 に示す．

SGmonitor における検知時間は従来手法と比較して 1.9 倍の時間がかかった．これは，エンクレイヴ内でファイルシステムを実行すること，および，仮想ディスクにブロックレベルでアクセスする際に OCALL を用いること，VM の仮想ディスクイメージに NFS でアクセスすることにより発生するオーバヘッドが原因と考えられる．一方，従来手法でも監視対象 VM 内で IDS を実行した場合と比較して 1.9 倍の時間がかかった．これは仮想ディスクをマウントするのにかかる時間が原因と考えられる．

5.4　プロセス数と EPC サイズの影響

監視対象 VM のプロセス数を増やしながら，VM のメモリからプロセス情報を取得して侵入検知を行うのにかかる時間を測定した．IDS VM に割り当てる EPC のサイズの影響を調べるために，EPC を 1MB だけ割り当てた場合についても測定を行った．従来手法と EPC が 93MB の場合は，プロセス数に比例して取得時間が増加した．しかし，EPC が 1MB の場合はプロセス数の増加に伴い，検知時間の増加率が大きくなった．監視対象 VM 内で動作するプロセス数が一般的な数百個程度の場合には，EPC が 1MB しかなくとも性能に大きな影響はないことがわかった．

6.　関連研究

BVMD [2] は VM の下で動作するハイパーバイザを信頼して，ハイパーバイザ内に IDS をオフロードし，安全に VM の監視を行うシステムである．しかし，ハイパーバイザ内で高度な IDS を動作させるのは難しい．また，ハイパーバイザに埋め込むため，IDS の更新などの管理が難しいという問題もある．SGmonitor では SGX アプリケーションとして IDS を作成するため，高度な IDS の開発も比較的容易であり，IDS の管理もしやすい．

Self-Service Cloud（SSC）[3] はユーザにだけ自身の VM を管理する権限を与えることができる．信頼できるハイパーバイザを用いることで，ユーザはサービスドメインと呼ばれるユーザだけがアクセスできる VM を作成することができる．そして，その VM 内でオフロードした IDS を安全に実行することができる．しかし，サービスドメイン内では OS も動作するため，そのシステムに脆弱性があった場合，IDS が攻撃を受ける可能性がある．SGmonitor ではエンクレイヴ内では IDS しか動作しないため，外部から脆弱性を利用した攻撃を行うのはより難しい．

RemoteTrans [6] はクラウドの外に信頼できるリモートホストを用意し，そのホスト内で IDS を安全に実行するシステムである．リモートホストはクラウド内の信頼できるハイパーバイザと暗号通信を行うことで VM の内部情報を取得し，安全な監視を行うことができる．しかし，クラウドの資源を利用して IDS を実行できないという問題がある．SGmonitor では IDS はクラウド内で安全に実行することができる．

ハードウェア機構を用いることで IDS を安全に実行する手法も提案されている．Copilot [4] は専用の PCI カードを用意してメモリの内容をリモートホストに送信し，IDS を実行する． しかし，専用ハードウェアを用いるのはコストが高い．HyperGuard [7] は CPU のシステムマネジメントモード（SMM）を用いることで安全にハイパーバイザを監視することができ，HyperCheck [5] は SMM を用いてメモリの内容をリモートホストに送って安全に監視を行うことができる．HyperSentry [8] は SMM を用いること

でハイパーバイザ内で安全に監視プログラムを実行することができる．しかし，SMM によるプログラムの実行は低速であり，IDS の性能が低下する．また，SMM でプログラムを実行する際にはシステム全体を停止させる必要がある．AMD SVM や Intel TXT を用いて IDS を安全に実行する Flicker [9] も提案されているが，IDS を実行する際には他の CPU コアを停止させる必要がある．SGmonitor では IDS 実行時にシステム全体を停止させる必要はない．

V-Met [10] はネストした仮想化を用いて仮想化システムをクラウド VM と呼ばれる VM 内で動作させ，その外側で安全に IDS を実行する．V-Met では，IDS はクラウド VM の中の監視対象 VM のメモリを特定して必要な OS データを取得する．しかし，ネストした仮想化を用いることにより仮想化システムのオーバヘッドが大きくなるという問題がある．SGmonitor では監視対象 VM を含む仮想化システムは従来通りの性能で動作する．

7. まとめ

本稿では，Intel SGX を用いてクラウド内で IDS を安全かつ軽量に実行し，情報漏洩を防ぎつつ VM 内の情報を取得できるシステム SGmonitor を提案した．SGmonitor はエンクレイヴ内で IDS を動作させることによって IDS の改ざんを防ぎ，監視対象 VM から取得した機密情報の漏洩を防ぐことを可能にする．また，リモートホストからハートビートを送ることで IDS の無効化を検出することができる．我々は SGmonitor を Xen-SGX 4.7 に実装した．実験の結果，エンクレイヴ内の IDS から監視対象 VM のメモリやディスク上のデータを取得し，侵入検知を行うことができることを確認した．

今後の課題は，監視対象 VM のディスクとして暗号ディスクを用いることができるようようにすることである．安全な鍵共有の実装を完成させることや，リモートホストからのハートビートを実装することも今後の課題である．また，エンクレイヴ内で Graphene-SGX [23] などのライブラリ OS を動作させることにより，既存の IDS を実行できるようにすることも検討している．

参考文献

[1] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, *Proceedings of Network and Distributed Systems Security Symposium*, pp. 191-206 (2003).

[2] Oyama, Y., Giang, T., Chubachi, Y., Shinagawa, T. and Kato, K.: Detecting Malware Signatures in a Thin Hypervisor, *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC' 12)*, pp. 1807- 1814 (2012).

[3] Butt, S., Lagar-Cavilla, H. A., Srivastava, A. and Ganapathy, V.: Self-service Cloud Computing, *Processings of Conference on Computer and Communications Security*, pp. 253-264 (2012).

[4] N. Petroni, Jr., T. Fraser, J. Molina, and W. Arbaugh: Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor, *Proceedings of the 13th Conference on USENIX Security Symposium*, pp.13-13 (2004).

[5] Wang, J., Stavrou, A. and Ghosh, A.: HyperCheck: A hardware-assisted Integrity Monitor, *Proceedings of International Symposium on Recent Advances in Intrusion Detection*, pp. 158-177 (2010).

[6] Kourai, K. and Juda, K.: Secure Offloading of Legacy IDSes Using Remote VM Introspection in Semi-trusted Clouds, *Proceedings of the 9th IEEE International Conference on Cloud Computing*, pp. 43-50 (2016).

[7] Rutkowska, J., Wojtczuk, R. and Tereshkin, A.: HyperGuard, *Xen 0wning Trilogy, Black Hat USA* (2008).

[8] Azab, A. M., Ning, P., Wang, Z., Jiang, X., Zhang, X. and Skalsky, N. C.: HyperSentry: Enabling Stealthy In-context Measurement of Hypervisor Integrity, *Proceedings of Conference on Computer and Communications Security*, pp. 38-49 (2010).

[9] McCune, J. M., Parno, B., Perrig, A., Reiter, M. K. and Isozaki, H.: Flicker: An Execution Infrastructure for TCB Minimization, *Proceedings of European Conference of Computer Systems*, pp. 315-328 (2008).

[10] Miyama, S. and Kourai, K.: Secure IDS Offloading with Nested Virtualization and Deep VM Introspection, *Proceedings of the 22nd European Symposium on Research in Computer Security, part II*, pp.305-323 (2017).

[11] Intel Corp., Xen SGX Virtualization Support, http://github.com/intel/xen-sgx/

[12] Ozaki, Y., Kanamoto, S., Yamamoto, H. and Kourai, K.: Detecting System Failures with GPUs and LLVM, *Proc. Asia-Pacific Workshop on Systems* (2019).

[13] TechSpot News: Google Fired Employees for Breaching User Privacy. http://www.techspot.com/news/40280-google-fired-employees-for-breaching-user-privacy.html (2010).

[14] PwC: US Cybercrime: Rising Risks, Reduced Readiness (2014).

[15] CyberArk Software: Global IT Security Service (2009).

[16] Li, C., Raghunathan, A. and Jha, N. K.: A Trusted Virtual Machine in an Untrusted Management Environment, *IEEE Transactions on Services Computing, Vol.5, No. 4*, pp. 472-483 (2012).

[17] Li, C., Raghunathan, A. and Jha, N. K.: Secure Virtual Machine Execution under an Untrusted Management OS, *Proceedings on International Conference on Cloud Computing*, pp. 172-179 (2010).

[18] Santos, N., Gummadi, K. P. and Rodrigues, R.: Towards Trusted Cloud Computing, *Proceedings of Workshop on Hot Topics in Cloud Computing* (2009).

[19] Intel Corp., Intel Software Guard Extensions SDK, http://software.intel.com/en-us/sgx/sdk/

[20] wolfSSL Inc.: wolfSSL — Embedded SSL Library for Applications, Devices, IoT, and the Cloud, http://www.wolfssl.com/

[21] Inokuchi, K. and Kourai, K.: UVBond: Strong User Binding to VMs for Secure Remote Management in Semi-Trusted Clouds, *Proc. Int. Conf. Utility and Cloud Computing*, pp.213-222 (2018).

[22] Patel, A. et al., Embedded Hypervisor Xvisor: A Comparative Analysis, PDP (2015).

[23] Tsai, C. et al.: Cooperation and Security Isolation of Library OSes for Multi-Process Applications, EuroSys (2014).