3. TDXの基盤技術

Ao Sakurai

2026年度セキュリティキャンプ全国大会
L1 – 暗号・CVMビルド&スクラップゼミ

本セクションの目標

- TDXを実現するために使用されている基盤技術について説明する

TDXを構成する基盤技術

Intel TDXを取り巻く背景

- はじめにTDXの基盤技術を以下の表にまとめる[1]：

技術名	説明
Intel VT	• CPU、メモリ、I/OのHW支援された仮想化を提供 • 信頼可能なハイパーバイザ（HV）によりVM間を分離する
Intel TME	• メインメモリ全体を暗号化する • ブート時に作成された単一の一時的な鍵を仕様 • 128bitまたは256bitのAES-XTSを使用
Intel TME-MK	• メモリ暗号化に複数の鍵を使えるようにしたTME • ページ単位の粒度でメモリ暗号化可能
Intel SGX	• アプリケーションの機密コード・データをEnclaveで保護する • メモリバス監視やコールドブート攻撃からもメモリ暗号化で守れる • Local AttestationやRemote Attestationに対応

Intel VT

Intel VT（1/2）

- **Intel VT**：Virtualization Technologyの略。ハードウェア（HW）支援型の仮想化環境のセット

- それまでは完全にソフトウェア（SW）ベースで仮想化を実装していたが、このIntel VTのような**HW支援の仮想化実装**の登場により、より優れた性能・分離・セキュリティが可能となった
  - VTにより、CPU、メモリ、I/OのHW支援仮想化が可能になる

- AMDにおいてはAMD-Vという技術がこのIntel VTに相当する

Intel VT（2/2）

- Intel VTは、さらに**VT-x**と**VT-d**の2つがサブセットとして存在する

- 簡単に言えば、VT-xがCPU仮想化を担当し、VT-dがI/O仮想化を担当する

- ただし当然であるが、VT自体は従来型VMに関する機能であり、これそのままではTDの仮想化を実現する事はできない点に注意
  - 後述する、さらに新たな要素の拡張により実現される

Intel VT-x（1/4）

- **Intel VT-x**: VTの内CPU仮想化に関する技術で、これを搭載したプロセッサは**VMX**という特別な命令セットを有し、このVMXを用いる事でHWベースの仮想化制御が行えるようになる
  - VT-x : Virtualization Technology for x86
  - VMX : Virtual Machine Extentions

- VT-xを搭載したプロセッサは、新たに以下の2つのモードで動作する事ができる：
  - **VMX root**：ハイパーバイザ（HV）が動作するモード
  - **VMX non-root**：ゲストVMが動作するモード

Intel VT-x（2/4）

- ゲストとHVとの切り替えを行う際には、**VMENTRY**と**VMEXIT**を用いる

- VMENTRYは文字通りHV側からVMに進入、つまりは**VMX non-root への切り替え**を行う

- 一方で、VMEXITは反対にVMからHVへの脱出、つまりは**VMX rootへの切り替え**を実施する

- TDXでは、より高いセキュリティが必要なため、後述する通りさらに新たな2つの動作モードが追加される

Intel VT-x（3/4）

- この2つのVMXモード間の遷移（**コンテキストスイッチ**）時には、遷移前のレジスタ状態等のコンテキストをどこかに退避しておく必要がある
  - そうでなければ、遷移前のモードに復帰する際、どこで中断したのかが分からなくなり、処理の続行が不可能となってしまう

- 例えばVMからHVにコンテキストスイッチする際、VMのコンテキストを退避しておかないと、HVからの復帰時にVMのどこから再開すれば良いのかがわからなくなる

Intel VT-x（4/4）

- このようなモード遷移時のVMやホストのステート等を保持するために、Intel VT-xでは**VMCS**と呼ばれるデータ構造を用いる
  - VMCS : Virtual Machine Control Structure
  - VMCSは、他にもどのゲスト操作がVMEXITを誘発するかの制御等も行う

- AMD-Vにおいては、VMCS相当のデータ構造としてVMCBというデータ構造が存在する

仮想化におけるアドレス変換（1/2）

- 仮想化を行う際には、通常の非仮想化環境と異なり、ホスト側とゲスト側それぞれで独自のアドレス空間を有する

- よって、アドレス変換を行う際にも多段階で行う必要があり、このようなアドレス変換を**SLAT**（セカンドレベルアドレス変換）または**Nested Paging**と呼ぶ
  - ちなみにSEV-SNPの「NP」はこのNested Pagingである

- VT-xでは、このSLATを行う際に**EPT**と呼ばれるものを使用する
  - EPT：Extended Page Table（拡張ページテーブル）

仮想化におけるアドレス変換（2/2）

- SLATではまず、ゲスト内の通常のページテーブルで、**ゲスト仮想アドレス（GVA）**から**ゲスト物理アドレス（GPA）**へのアドレス変換を実施する

- GPAはゲストからすれば物理アドレスであるが、ホストからすればまだ一種の**仮想的なアドレス**であるため、さらにホストから見た**真の物理アドレスに変換**する必要がある（逆方向の変換も然り）

- そこで、HVはEPTを使用してゲスト物理アドレス（GPA）から真の物理アドレスである**ホスト物理アドレス（HPA）**への変換を実施する、という流れである

Intel VT-d（1/3）

- VMがI/Oにアクセスする際、主にSWベースのI/OモデルとHWベースのI/Oモデルの2つに大別できる
- SWベースのI/Oモデルとしては、エミュレートされたデバイスへのアクセスや、準仮想化[2]を用いる方法といったものがある
- HWベースとしては、デバイスの直接割り当て、SR-IOV、S-IOVデバイスが存在する
  - SR-IOV：シングルルートI/O仮想化
  - S-IOV：スケーラブルI/O仮想化

Intel VT-d（2/3）

- このように様々な方法でデバイスはVMに接続されるが、いずれの場合も**Intel VT-d**があると、デバイスを管理する主体（例：HV）がデバイスアクセスの分離・制限を行う事が可能になる
  - VT-d : Virtualization Technology for directed I/O

- 例えば、VT-dを利用する事でI/Oデバイスの割り当て、DMA再マッピング、割り込み再マッピング、割り込みポスティング等が可能となる
  - 割り込みポスティング：割り込みをキューイングし必要に応じて自動注入する仕組み

Intel VT-d（3/3）

- またVT-dがあると、VMはIOMMUの助けを借りて仮想-物理アドレス変換を行う事ができる
  - これはつまり、MMIO等によりIOMMUにアクセスすれば直接物理I/Oにアクセスできる事を意味する
- 他にも、VT-dにより前述のデバイス割り当てや各種再マッピングを行う事により、HVがそれらのI/O操作に介入する必要がなくなる
  - これにより、HWベースの高速なデバイス仮想化を実現できる
- 総じて、VT-dはI/Oデバイスへの直接アクセスを必要とする仮想化環境のパフォーマンスを向上させられる技術であると言える

VTからTDXへの拡張（1/2）

- TDXはTD間分離にこのVTに依存しているが、VTが想定している脅威モデルと異なり、TDXの脅威モデルでは**HVを信頼する事はできない**
  - よって、TDXではセキュリティクリティカルなTDの管理操作はHVではなくTDX Moduleが担当する

- これに伴い、前述の従来のVMXモードに加え、以下の2つの新しいモードがTDXでは追加されている：
  - **SEAM VMX root**：主にTDX Moduleと、そのロードを行うP-SEAM Loaderが動作する特権モード
  - **SEAM VMX non-root**：TDが動作するモード

VTからTDXへの拡張（2/2）

- SEAMはSEcure Arbitration Mode（直訳：安全仲裁モード）の略

- TDXにおいてもSLATのためにEPTを用いるが、単一のTDに対し、TDのプライベートメモリ用（**SEPT**）と共有メモリ用（**共有EPT**）の2つのEPTを使用する

- ネストされた仮想化（VM内でさらに入れ子のVMを持つような仮想化構成）は論文[1]時点では未対応だったが、現在では**TD Partitioning**としてネストされたTDに対応している
  - 昔はTD内でVMX命令を使おうとすると未定義例外（#UD）が出ていた

Intel TME/TME-MK

Intel TME/TME-MK（1/5）

- Intel **TME**及び**TME-MK**は、前回スライドでも説明の通り、**CPU内に搭載された暗号エンジン**により**メモリ暗号化**を行う事ができる機能である
  - TME：Total Memory Encryption、MK：Multi-Key
  - 第11世代Core vProプロセッサで初めてTMEが搭載され、その後TME-MKへと拡張された

- TMEは、起動の度に動的に生成される**単一の鍵**により**メモリ全体を暗号化**する事で、メモリに物理的にアクセスし盗聴しようとする攻撃者から保護する事を目的としている
  - ただし、前スライドの通り脅威モデルに限界はある

Intel TME/TME-MK（2/5）

- 具体的には、このTME鍵は起動時にHWベースの乱数生成器（RNG）と、チップセットに組み込まれたセキュリティ機能の組み合わせにより生成される
  - 暗号化方式は**128bitまたは256bitのNIST標準AES-XTS**
  - Intel SGXのEnclaveはこのTMEにより暗号化される

- TME-MKはこのTMEを拡張したもので、**複数の**メモリ暗号化鍵の用意や**メモリ暗号化**を**ページ単位**で行う事ができる

Intel TME/TME-MK（3/5）

- TME-MKでは複数鍵による対象の別々の暗号化の実現のため、そのメモリ位置を用いる処理が発生する度に、物理メモリアドレスから**HKID**を抽出する
  - HKID：Host Key Identifier
- HKIDは**物理アドレス**の最上位ビットから特定のビット数のビット列（**要は先頭nビット**）により**表現**される
  - このビット数は一般的にはBIOS設定から変更する事ができる
- TMEではHW完結の暗号鍵が使用されるが、TME-MKではSWにより提供される鍵を用いる事もできる
  - ただしTDXではMK-TMEエンジン内のHW鍵を用いるため、SW提供鍵が使われる事は無い

Intel TME/TME-MK（4/5）

- HKIDとTME-MK鍵は対応付けられた状態でなければならないため、このバインドを行う命令として**PCONFIG命令**が用意されている

- PCONFIG命令によりHKIDと鍵がバインドされると、そのタプルはMK-TMEエンジン内に存在する**KET**と呼ばれるテーブルにストアされる
  - KET：Key Encryption Table

- KETはプロセッサ外に出る事は無いため、**常にTME-MK鍵は**
  プロセッサ内で保持される事になる
  - これはプロセッサ内を無条件に信頼するTEEの脅威モデルに合致している

Intel TME/TME-MK（5/5）

- TME-MKはベアメタル（非仮想化）環境と仮想化環境のどちらでも利用する事ができる

- 仮想環境の場合は、HVが前述のEPT内のVM物理アドレスにHKIDをアタッチする事で、各VMのメモリを別々のTME-MK鍵で暗号化できるようにする

TME-MKからTDXへの拡張（1/2）

- 前ページの通り従来のVM環境ではHVがHKIDのアタッチを行っていたが、TDXではHVを信頼できないため、当然この方法を用いる事はできない
  - もしHVが管理してしまえば、例えばTDや従来型VM、そしてホスト側全てを同じ鍵で暗号化し、セキュリティ強度を著しく下げる事ができてしまう

- よって、TDXにおいてはこれまた**TDX Module**がこのメモリ暗号化管理を実施する

- TDXのメモリ暗号化自体は勿論TME-MKが使われるが、TME-MKは128bit/256bit双方のAES-XTSを使えるのに対し、**TDXのメモリ暗号化は128bit AES-XTSのみ**となっている

TME-MKからTDXへの拡張（2/2）

- TDXにおいては、HKIDは**プライベートHKID**と**共有HKID**の2つに分割される

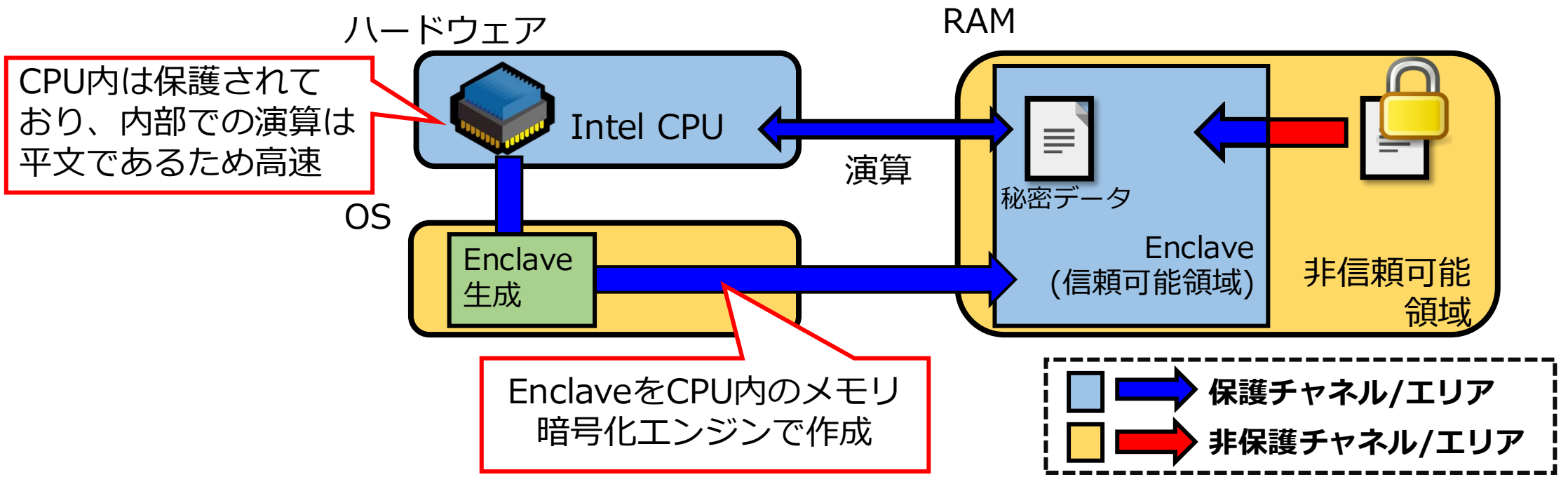
- プライベートHKIDはTDやTDX Moduleのみがこのプライベート HKIDに紐付けられた鍵で内容が暗号化されたメモリを読み書きできる
  - 要はTDXにおけるプライベートメモリの暗号化を行うTME-MK鍵に対して使用される

- 一方で、共有HKIDはTDの共有メモリの他、従来型のVMやホストカーネルのメモリ暗号化のためのTME-MK鍵に割り当てる事ができる

Intel SGX

Intel SGX (Software Guard eXtensions)

- CPUの専用ユニット（MEEやMK-TMEエンジン）によって
  暗号学的にも保護された**保護領域** (**Enclave**)を専用のCPU命令で
  生成し、**OS**や**VMM**からすらも**データを保護**しながら**プログラムを実行**
  - 第6～10世代Core、第2世代Xeon（一部）, 第3世代以降Xeon-SPで対応
  - 第2世代XeonまではMEE、第3世代Xeon-SP以降はIntel TMEを用いている



Intel SGX

- SGXはTDXよりも大分前（2015年）からIntelにより提供されている部分隔離型のTEE実装である

- 去年まではセキュリティキャンプでSGXを取り扱っており、その説明は過去の資料が詳しいため、詳細を知りたい場合には以下のリンク先を参照
  https://qliphoth.io/seccamp/

TDXでのSGXの使われ方（1/4）

- SGXは2026年4月現在、**TDXのAttestationを実現する上で必須の存在**となっている

- SGX Enclaveは、あるTDが自身と同一のマシン上に存在する事を暗号学的に確認する処理である、**Local Attestation（LA）**を実施する事ができる

- このLAは、リモートユーザがあるTDについて真に期待する通りのTDであるか、そしてその安全性を検証する手続きである、**Remote Attestation（RA）**において必須の手続きである

TDXでのSGXの使われ方（2/4）

- RAを行うには、まず**TDQE**という専用のSGX EnclaveがTDに対しLAを実施する
  - TDQE：TD Quoting Enclave

- 具体的には、まず**Report Key**と呼ばれるそのマシン固有の鍵でMACが打たれている**TD Report**という身元証明書をTDがTDX命令で生成する

- TDQEはそれを受け取り、**ENCLU[EVERIFYREPORT2]**という専用のSGX命令でそのMAC値を同じくReport Keyで検証する

TDXでのSGXの使われ方（3/4）

- MACが一致したら、TDQEはIntelをトラストアンカーとしている**Attestation Key（AK）**でTD Reportに署名し、**TD Quote**を生成する

- TD QuoteはIntelをトラストアンカーとした公開証明書チェーンでその署名を検証できるため、リモートユーザでもその身元を検証する事ができる
  - 反対に、LAはそのマシン内以外ではアクセス不能なReport KeyでMACが打たれているので、リモートからそれを検証する事はできない

TDXでのSGXの使われ方（4/4）

- なお、2026/4時点では**TD内でSGX命令（ENCLS/ENCLU）を実行する事はできない**
  - 実行すると未定義例外（#UD）が発生する

- これはつまり、**TDからSGX EnclaveをLAする事は不可能**である事を意味する。そもそもそのようなモチベーションが無いかも知れないが、覚えておいて損はない
  - SGX EnclaveのREPORTの検証には、EGETKEYと呼ばれるSGXのENCLU命令が必要となるため

まとめ

- Intel TDXの基盤技術である、Intel VT、Intel TME/TME-MK、Intel SGXについて簡単に説明した

参考文献

[1] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2024. Intel TDX Demystified: A Top-Down Approach. ACM Comput. Surv. 56, 9, Article 238 (September 2024), 33 pages. https://doi.org/10.1145/3652597

[2] “Virtioと完全仮想化・準仮想化”, 2026/1/29閲覧, https://zenn.dev/junjunjunjun/articles/27ede76931cc85