# 投機実行のサイドチャネル攻撃

この記事は、インテル® デベロッパー・ゾーンに公開されている「Speculative Execution Side Channel Mitigations」の日本語参考訳です。

---

リビジョン 2.0 (初版出版日: 2018 年 5 月)

## 1. はじめに

サイドチャネル攻撃とは、システムの物理特性を測定するなどして、攻撃者がシステムを観測することにより情報を盗み取る手法です。このドキュメントでは、分岐ターゲット・インジェクション、境界チェックのバイパス、投機ストアのバイパスといった複数のサイドチャネル手法について考察します。

セクション 2 では、分岐ターゲット・インジェクション (英語) について説明し、プロセッサーとシステム・ソフトウェアの間の新しいインターフェイスである、間接分岐制御メカニズムに基づく緩和策を紹介します。

セクション 3 では、境界チェックのバイパス (英語) とソフトウェア変更に基づく緩和策を説明します。

セクション 4 では、投機ストアのバイパス (英語) と投機ストアのバイパスの無効化やソフトウェア変更による緩和策を説明します。

セクション 5 では、一部の緩和策で利用可能な CPUID の列挙とアーキテクチャー MSR について説明します。

## 2. 間接分岐を制御する緩和策

### 2.1. 分岐ターゲット・インジェクションの概要

インテル® プロセッサーは、「間接分岐予測器」を使用して、間接分岐命令の後に投機実行される操作を決定します。分岐ターゲット・インジェクション (英語) は、間接分岐予測器を利用したサイドチャネル攻撃です。攻撃者は、間接分岐予測器の動作を制御 (「トレーニング」) することで、特定の命令を投機実行させ、サイドチャネル解析に利用します。

### 2.2. 間接分岐制御メカニズムの概要

インテルは、分岐ターゲット・インジェクションに対する緩和策を開発しました。その 1 つが、プロセッサーとシステム・ソフトウェア間の新しいインターフェイスである、「間接分岐制御メカニズム」を使用する方法です。このメカニズムにより、システム・ソフトウェアは、適切なタイミングで間接分岐予測器を無効にするなどして、攻撃者が間接分岐予測を制御することを防ぐことができます。

次の 3 つの間接分岐制御メカニズムが定義されています。

- 間接分岐の投機実行制限 (IBRS) (英語): 間接分岐の投機実行を制限します。

- シングルスレッド間接分岐予測器 (STIBP) (英語): 間接投機予測がハイパースレッドによって制御されるのを防ぎます。
- 間接分岐予測器バリア (IBPB) (英語): バリア後の間接分岐予測が、バリア前に実行されたソフトウェアによって制御されることを防ぎます。

適切に記述されたソフトウェアは、これらの間接分岐制御メカニズムにより分岐ターゲット・インジェクション攻撃を防ぐことができます。

## 2.3. 背景と用語

### 2.3.1. 間接分岐予測

プロセッサーは、間接分岐予測器を使用して、以下の表に列挙されている分岐命令の動作のみを制御します。

| 間接分岐予測器を使用する命令 | | |
|---|---|---|
| **分岐の種類** | **命令** | **オペコード** |
| ニアコール間接分岐 | `CALL r/m16、CALL r/m32、CALL r/m64` | `FF /2` |
| ニアジャンプ間接分岐 | `JMP r/m16、JMP r/m32、JMP r/m64` | `FF /4` |
| ニアリターン分岐 | `RET、RET Imm16` | `C3、C2 Iw` |

このドキュメントでは、ニアコール間接分岐、ニアジャンプの間接分岐、およびニアリターン分岐命令のみを間接分岐とします。

### 2.3.2. 間接分岐予測とインテル® ハイパースレッディング・テクノロジー (インテル® HT テクノロジー)

インテル® ハイパースレッディング・テクノロジー (インテル® HT テクノロジー) をサポートするプロセッサーでは、1 つのコア (または物理プロセッサー) に複数の論理プロセッサーが含まれる場合があります。このようなプロセッサーでは、コアを共有する論理プロセッサーが間接分岐予測器を共有することがあります。この共有により、あるコアの論理プロセッサー上のソフトウェアが、同じコアの別の論理プロセッサーで実行される間接分岐予測ターゲットを制御できる場合があります。

この共有はコア内でのみ行われます。あるコアの論理プロセッサー上で実行されるソフトウェアが、別のコアの論理プロセッサーによる間接分岐予測ターゲットを制御することはできません。

### 2.3.3. リターン・スタック・バッファー (RSB)

リターン・スタック・バッファー (RSB) は、ニア `RET` 命令の実行予測を保持するマイクロアーキテクチャー構造です。

非ゼロのディスプレイスメントのニア `CALL` 命令を実行するたびに [1]、`CALL` 命令に続く命令のアドレスを含むエントリーが RSB に追加されます。RSB は、ファー `CALL`、ファー `RET`、`IRET` 命令によって使用または更新されません。

### 2.3.4 予測器モード

インテル® プロセッサーは、異なる権限に対応したさまざまな操作モードをサポートしています。VMX ルート操作 (「ホスト」 - 仮想マシンのモニター) は、VMX 非ルート操作 (「ゲスト」 - 仮想マシン) よりも権限があります。VMX ルート操作内あるいは VMX 非ルート操作内では、「supervisor モード」 (`CPL < 3`) のほうが「user モード」 (`CPL= 3`) よりも権限があります。

分岐ターゲット・インジェクション攻撃を防ぐには、権限の低いソフトウェアが、権限の高いソフトウェアによる分岐予測の使用を制御できないようにすることが重要です。このため、「predictor モード」という概念を導入すると便利です。次の 4 つの predictor モードを利用できます。

- host-supervisor
- host-user
- guest-supervisor
- guest-user

guest predictor モードは、host predictor モードよりも低い権限レベルです。同様に、user predictor モードは supervisor predictor モードよりも権限レベルは低くなります。

関連性のないソフトウェア・コンポーネント間の移行に使用され、CPL の変更や VMX の移行が発生しない操作があります。これらの操作は predictor モードを変更しません。例えば、`CR3` への `MOV`、`VMPTRLD`、EPTP の切り替え (VM function 0 を使用)、`GETSEC[SENTER]` などです。

## 2.4. 間接分岐制御メカニズム

### 2.4.1 間接分岐の投機実行制限 (IBRS)

**間接分岐の投機実行制限 (IBRS)** は、間接分岐の投機実行を制限する間接分岐制御メカニズムです。CPUID.(EAX=7H,ECX=0):EDX[26] が 1 の場合、プロセッサーは IBRS をサポートします。

#### 2.4.1.1. IBRS: 基本サポート

IBRS をサポートするプロセッサーは、ソフトウェアで有効にされなくても、以下の保証を提供します。

- エンクレーブ (インテル® SGX で定義された保護されたコンテナー) 内で実行されるニア間接分岐予測ターゲットは、エンクレーブ外で実行されるソフトウェアによって制御されません。
- SMI および SMM のデフォルト処理が有効な場合、システム管理割り込み (SMI) の前に実行されるソフトウェアは、SMI の後にシステム管理モード (SMM) で実行される間接分岐予測ターゲットを制御できません。

#### 2.4.1.2. IBRS: ソフトウェアで有効にされるサポート

IBRS は、重要なソフトウェアが間接分岐予測を保護する方法を提供します。

権限レベルの高い predictor モードに移行した後、ソフトウェアが IA32_SPEC_CTRL.IBRS を 1 に設定した場合、IA32_SPEC_CTRL.IBRS = 1 でその predictor モードで実行された間接分岐予測ターゲットは、権限レ

ベルの低い predictor モードで実行されたソフトウェアによって制御されることはありません [2]。さらに、そのコアのいずれかの論理プロセッサーで IA32_SPEC_CTRL.IBRS が 1 に設定されている場合、間接分岐予測ターゲットは、同じコアの別の論理プロセッサー上で実行される (または以前に実行された) ソフトウェアによって制御されることはありません。

権限レベルの高い predictor モードに移行する前に IA32_SPEC_CTRL.IBRS がすでに 1 に設定されている場合、一部のプロセッサーでは、その predictor モードで実行される間接分岐予測ターゲットが、移行前に実行されたソフトウェアによって制御されることがあります。このような移行の後に、IA32_SPEC_CTRL MSR に対して `WRMSR` を使用して IBRS ビットを 1 に設定することで、以前の IBRS ビットの値に関係なく、ソフトウェアはこれを回避できます。最初にビットをクリアする必要はありません。ビットの元の値に関係なく、移行後に 1 を書き込むだけです。

IA32_SPEC_CTRL.IBRS を 1 に設定しても、ニアリターンの予測ターゲットが、権限レベルの低い predictor モードで作成された RSB エントリーを使用するのを防ぐことはできません。ソフトウェアは、権限レベルの高い predictor モードへ移行した後に RSB 上書きシーケンス [3] を使用することで、これを回避できます。supervisor モード実行防止 (SMEP) 機能を有効にすると、user モードから supervisor モードへの移行時にこれらの処理を行う必要がありません。SMEP は、supervisor モードで user モードのページ上のコードを (投機実行を含め) 実行することを防ぎます。user モードのコードは、自分自身のリターンアドレスのみを RSB に挿入できます。supervisor モードのコードページ上のターゲットのリターンアドレスを挿入することはできません。OS とアプリケーションに別々のページテーブルが使用されている SMEP が有効でない部分では、OS のページテーブルが user モードのコードを実行不可 (no-execute) としてマッピングできます。プロセッサーは、実行不可とマークされた命令を投機実行しません。

IBRS を有効にしても、同じ predictor モードで後から実行される関連性のないソフトウェアの間接分岐予測ターゲットをソフトウェアが制御することはできません (例えば、2 つの異なるユーザー・アプリケーション間や 2 つの異なる仮想マシン間など)。このような分離には、「間接分岐予測器バリア (IBPB)」セクションで説明する IBPB コマンドを使用できます。

インテル® HT テクノロジーをサポートするコアの 1 つの論理プロセッサーで IBRS を有効にすると、同じコアのほかの論理プロセッサーの分岐予測に影響を与える可能性があります。そのため、ソフトウェアは `HLT` や `MWAIT` の実行などによりスリープ状態に入る前に、IA32_SPEC_CTRL.IBRS をクリアして IBRS を無効にし、ウェイクアップ時や間接分岐を実行する前に IBRS を再度有効にする必要があります。

#### 2.4.1.3. 拡張 IBRS

一部のプロセッサーでは、ソフトウェアによる有効化を容易にし、パフォーマンスを向上することで、IBRS を拡張できます。`RDMSR` が IA32_ARCH_CAPABILITIES MSR のビット 1 に対して 1 を返す場合、プロセッサーは拡張 IBRS をサポートします。

拡張 IBRS は、IA32_SPEC_CTRL.IBRS を設定することで IBRS を一度有効にして、その後無効にしないようにする「always on」モデルをサポートします。拡張 IBRS をサポートするプロセッサーで IA32_SPEC_CTRL.IBRS = 1 にすると、実行される間接分岐予測ターゲットは、権限レベルの低い predictor モードやほかの論理プロセッサー上で実行されるソフトウェアでは制御できません。

その結果、拡張 IBRS をサポートするプロセッサー上で動作するソフトウェアは、権限レベルの高い predictor モードに移行するたびに、`WRMSR` を使用して IA32_SPEC_CTRL.IBRS を設定する必要がありません。ソフトウェアは、このビットを一度設定するだけで、predictor モードを効果的に分離することができます。拡張 IBRS

が有効な部分では、ソフトウェアは MWAIT や HLT などのスリープ状態に入る前に IBRS や STIBP を無効にする必要はありません。

拡張 IBRS をサポートするプロセッサーでは、RSB の上書きシーケンスを使用しても、ニアリターンの予測ターゲットが権限レベルの低い predictor モードで作成された RSB エントリーを使用するのを防げないことがあります。ソフトウェアは、user モードから supervisor モードへの移行時に SMEP を有効にしたり、VM 終了時に IA32_SPEC_CTRL.IBRS を設定することで、これを防ぐことができます。拡張 IBRS をサポートするプロセッサーは、SMEP を有効にする OS の OS/VMM でのみ IBRS が設定される使用モデルをサポートします。このようなプロセッサーでは、IBRS が設定されると、VM 終了時に IBRS が設定されていなくても、VM 終了後にゲストの動作が RSB を制御することはできません。ゲストが IBRS をクリアした場合、ハイパーバイザーは、IBRS はサポートし拡張 IBRS はサポートしていないプロセッサーと同様に、VM 終了後に IBRS を設定する必要があります。IBRS と同様に、拡張 IBRS を有効にすると、ソフトウェアが同じ predictor モードで実行される間接分岐予測ターゲットに影響を与える可能性があります。このような場合には、「間接分岐予測器バリア (IBPB)」セクションで説明する IBPB コマンドを使用します。

拡張 IBRS をサポートするプロセッサーでは、IBRS を 1 に設定し、そのままにしておくことを推奨します。リング 0 の実行時にのみ IBRS を設定する従来の IBRS モデルは、拡張 IBRS をサポートするプロセッサーでも、標準の IBRS をサポートするプロセッサーと同様に安全ですが、リングの遷移時や VM の終了/開始時に WRMSR が発生するため、IBRS を設定したままにしておくよりもパフォーマンスが低下します。繰り返しになりますが、IBRS が設定されている場合、STIBP を使用する必要はありません。ただし、特定のハードウェア・スレッド上で最後に実行されたアプリケーション/ゲストを信頼しない別のアプリケーション/ゲストに切り替える場合は、IBPB を使用する必要があります。拡張 IBRS をサポートしていないハードウェアを含む VM マイグレーション・プールのゲストは、IA32_ARCH_CAPABILITIES.IBRS_ALL (拡張 IBRS) が列挙されていない可能性があるため、リング 0 でのみ IBRS を設定する従来の IBRS 使用モデルを使用できます。パフォーマンス上の理由から、ゲストが IA32_SPEC_CTRL 書き込みを頻繁に行う場合、VMM がそのような WRMSR で VM を終了することは推奨されません。拡張 IBRS をサポートするプロセッサー上で実行されている VMM は、IA32_SPEC_CTRL を書き込むゲストがゲストの IA32_SPEC_CTRL を制御できるようにする必要があります。したがって、VMM は自身を保護するため、そのようなゲストから VM が終了した後に IBRS を設定する必要があります (または、retpoline、秘密除去、間接分岐の除去などの代替手法を使用します)。

### 2.4.2. シングルスレッド間接分岐予測器 (STIBP)

シングルスレッド間接分岐予測器 (STIBP) は、コア上の論理プロセッサー間での分岐予測の共有を制限する、間接分岐制御メカニズムです。CPUID.(EAX=7H,ECX=0):EDX[27] が 1 の場合、プロセッサーは STIBP をサポートします。論理プロセッサー上の IA32_SPEC_CTRL MSR のビット 1 (STIBP) を設定することで、そのコアの任意の論理プロセッサー上の間接分岐予測ターゲットが、同じコアの別の論理プロセッサー上で実行される (または以前に実行された) ソフトウェアによって制御されることを防ぎます。

IBRS や IBPB とは異なり、STIBP は間接分岐予測を含むすべての分岐予測器に影響を与えるわけではありません。STIBP は、あるハードウェア・スレッドのソフトウェアが予測を作成し、それをほかのハードウェア・スレッドが使用できるような分岐予測にのみ影響を与えます。これは、現在の実装で STIBP のパフォーマンス・オーバーヘッドが IBRS よりも低い原因の 1 つです。

特にセキュリティーを重要視するプロセスでは、実行時に STIBP を設定して、同じ物理コア上の別のハードウェア・スレッドによって間接分岐予測が制御されないようにしたほうが良いでしょう。既存のインテル® Core™ プロセッサー・ファミリーでは、前述のように、一部の間接分岐予測を無効にするため、両方のハードウェア・ス

レッドに大きなパフォーマンス・コストがかかります。このため、すべてのアプリケーションの実行中に STIBP を設定することは推奨しません。

「間接分岐予測とインテル® ハイパースレッディング・テクノロジー (インテル® HT テクノロジー)」セクションに示すように、コアを共有する論理プロセッサーは間接分岐予測器を共有することができ、1 つの論理プロセッサーが同じコアのほかの論理プロセッサーの間接分岐予測ターゲットを制御できます。STIBP と IBRS の両方をクリアした状態で実行する前に、IBPB を実行して、STIBP が設定されている間に信頼できないプロセスによってインストールされた間接分岐予測が、STIBP と IBRS がクリアされた後にほかのハードウェア・スレッドによって使用されないようにします。どのような使用モデルであっても、STIBP はパフォーマンスに影響を与えるため、慎重に使用する必要があります。

間接分岐予測器はコア間で共有されることはありません。そのため、あるコアで実行された間接分岐予測ターゲットは、ほかのコアで動作するソフトウェアの影響を受けることはありません。ほかのコアで動作するソフトウェアから間接分岐予測を分離するため、IA32_SPEC_CTRL.STIBP を設定する必要はありません。

多くのプロセッサーでは、STIBP の設定に関係なく、ほかの論理プロセッサー上で動作するソフトウェアによって間接分岐予測ターゲットが制御されることはありません。このようなプロセッサーには、インテル® HT テクノロジーが有効でないプロセッサーや、論理プロセッサー間で間接分岐予測器のエントリーを共有していないプロセッサーが含まれます。ソフトウェアによる有効化を容易にし、ワークロードの移行を促進するため、このようなプロセッサーでは STIBP を列挙できます (IA32_SPEC_CTRL.STIBP の設定を許可します)。

プロセッサーは、IA32_SPEC_CTRL MSR のサポートを列挙できますが (例えば、CPUID.(EAX=7H,ECX=0):EDX[26] を 1 として列挙)、STIBP のサポートは列挙されません (CPUID.(EAX=7H,ECX=0):EDX[27] は 0)。このようなプロセッサーでは、IA32_SPEC_CTRL への WRMSR の実行はビット 1 (STIBP) の値を無視し、ソースオペランドのビット 1 が設定されていても、一般保護例外 (#GP) は発生しません。これにより、仮想化が容易になると期待されています。

「間接分岐の投機実行制限 (IBRS)」セクションに示すように、IBRS を有効にすると、ある論理プロセッサー上で動作するソフトウェアは、別の論理プロセッサー上で実行される間接分岐予測ターゲットを制御できなくなります。そのため、IBRS が有効な場合は、STIBP を有効にする必要はありません。

拡張 IBRS をサポートする今後のプロセッサーでは、必要な間接分岐予測器にスレッド ID ビットが追加され、そのビットを使用して、間接分岐予測が作成したスレッドによってのみ使用されるようになります。IBRS および拡張 IBRS は、現在および将来のプロセッサーの両方において、機能のスーパーセットであるため、STIBP と IBRS の両方を設定することは (IBRS だけを設定する場合に比べて) 冗長になります。

インテル® HT テクノロジーをサポートするコアの 1 つの論理プロセッサーで STIBP を有効にすると、同じコアのほかの論理プロセッサーの分岐予測に影響を与える可能性があるため、ソフトウェアは HLT や MWAIT の実行などによりスリープ状態に入る前に、IA32_SPEC_CTRL.STIBP をクリアして STIBP を無効にし、ウェイクアップ時や間接分岐を実行する前に STIBP を再度有効にする必要があります。

拡張 IBRS をサポートするプロセッサーでは、IBRS を 1 に設定し、そのままにしておくことを推奨します。リング 0 の実行時にのみ IBRS を設定する従来の IBRS モデルは、拡張 IBRS をサポートするプロセッサーでも、IBRS をサポートするプロセッサーと同様に安全ですが、リングの遷移時や VM の終了/開始時に WRMSR が発生するため、IBRS を設定したままにしておくよりもパフォーマンスが低下します。繰り返しになりますが、IBRS が設定されている場合、STIBP を使用する必要はありません。ただし、特定のハードウェア・スレッド上で最後に実行されたアプリケーション/ゲストを信頼しない別のアプリケーション/ゲストに切り替える場合は、IBPB を使用する必要があります。拡張 IBRS をサポートしていないハードウェアを含む VM マイグレーション・プール

のゲストは、IA32_ARCH_CAPABILITIES.IBRS_ALL (拡張 IBRS) が列挙されていない可能性があるため、リング 0 でのみ IBRS を設定する従来の IBRS 使用モデルを使用できます。パフォーマンス上の理由から、ゲストが IA32_SPEC_CTRL 書き込みを頻繁に行う場合、VMM がそのような `WRMSR` で VM を終了することは推奨されません。拡張 IBRS をサポートするプロセッサー上で実行されている VMM は、IA32_SPEC_CTRL を書き込むゲストがゲストの IA32_SPEC_CTRL を制御できるようにする必要があります。そのため、VMM は自身を保護するため、そのようなゲストから VM が終了した後に IBRS を設定する必要があります (または、retpoline、秘密除去、間接分岐の除去などの代替手法を使用します)。

### 2.4.3 間接分岐予測器バリア (IBPB)

**間接分岐予測器バリア (IBPB)** は、同じ論理プロセッサーでバリア以前に実行されたソフトウェアが、バリア以降に実行される間接分岐予測ターゲットを制御できないようにバリアを設ける間接分岐制御メカニズムです。CPUID.(EAX=7H,ECX=0):EDX[26] が 1 の場合、プロセッサーは IBPB をサポートします。

IBRS や STIBP とは異なり、IBPB は分岐予測器を制御するプロセッサーの新しい動作モードを定義しません。そのため、IA32_SPEC_CTRL MSR のビットを設定することで有効になるわけではありません。代わりに、IBPB はソフトウェアが必要に応じて「コマンド」を実行します。

ソフトウェアは、IA32_PRED_CMD MSR のビット 0 (IBPB) を設定することで、IBPB コマンドを実行します。これは、`WRMSR` 命令を使用するか、MSR ロード領域から MSR をロードする VMX 遷移の一部として行われます。IBPB コマンドの前に実行されたソフトウェアは、同じ論理プロセッサー上でコマンドの後に実行される間接分岐予測ターゲットを制御できません [4]。IA32_PRED_CMD MSR は書き込み専用であり、1 を書き込む前に IBPB ビットをクリアする必要はありません。

IBPB は IBRS と併用することで、IBRS がカバーしないケースに対応できます。

- 「間接分岐の投機実行制限 (IBRS)」に示すように、IBRS は、同じ predictor モードで実行される無関係なソフトウェア (例えば、別のユーザー・アプリケーションや別の仮想マシン) の間接分岐予測ターゲットをソフトウェアが制御することを妨げません。ソフトウェアは、特定の predictor モードで動作するソフトウェアの ID の変更時 (ユーザー・アプリケーションや仮想マシンの変更時など) に IBPB コマンドを実行することで、そのような制御を防ぐことができます。
- ソフトウェアは、特定の状況で IA32_SPEC_CTRL.IBRS をクリアできます (例えば、VMX ルート操作を `CPL = 3` で実行する場合など)。このような場合、ソフトウェアは特定の遷移時 (信頼できない仮想マシンを実行した後など) に IBPB コマンドを使用して、以前に実行したソフトウェアが、IBRS が無効な以降に実行される間接分岐予測ターゲットを制御するのを防ぐことができます。

### 2.4.4. retpoline

IBRS に代わる方法として、Google 社が開発した「retpoline」というソフトウェア手法があります。retpoline の詳細は、「retpoline: 分岐ターゲット・インジェクションの緩和策」 (英語) を参照してください。

# 3. 境界チェックのバイパスの緩和策

## 3.1. 境界チェックのバイパスの概要

境界チェックのバイパスは、条件分岐命令の後に発生する投機実行を利用したサイドチャネル攻撃です。具体的には、入力が境界内にあるかどうかをプロセッサーがチェックしている状況で使用されます (例えば、読み込まれる配列要素のインデックスが許容値内にあるかどうかをチェックしているときなど)。プロセッサーは、境界チェックを行わずに、操作を投機実行することがあります。攻撃者がこれらの操作で境界外のメモリーにアクセスするように仕向けた場合、特定の状況下では攻撃者に情報が漏れる可能性があります。

## 3.2. 境界チェックのバイパスの緩和メカニズム

境界チェックのバイパスは、境界チェックが適切に行われていない投機的な操作を抑制するようにソフトウェアを修正することで緩和できます。具体的には、ソフトウェアは、境界チェックと、サイドチャネル攻撃を引き起こす可能性のある後続の操作との間に、投機を阻止するバリアを挿入できます。`LFENCE` 命令やその他のシリアル化命令がそのようなバリアとして機能します。これらの命令は、境界チェックが条件分岐を使用して実装されているか、インテル® メモリー・プロテクション・エクステンション (インテル® MPX) の境界チェック命令 (`BNDCL` および `BNDCU`) を使用して実装されているかにかかわらず機能します。

`LFENCE` 命令とシリアル化命令は、すべての先行する命令がローカルで完了するまで、後続の命令が (投機実行を含め) 実行されないことを保証します。LFENCE 命令はシリアル化命令に比べてレイテンシーが低いため、LFENCE 命令の使用を推奨します。

`CMOVcc`、`AND`、`ADC`、`SBB`、`SETcc` などの命令も、インテル® Core™ プロセッサー、Intel Atom® プロセッサー、インテル® Xeon® プロセッサーおよびインテル® Xeon Phi™ プロセッサーで投機実行を制限して、境界チェックのバイパスを防止するために使用できます。ただし、これらの命令は、将来のインテル® プロセッサーで動作が保証されていません。将来、インテルは動作が異なるプロセッサーを発売する前に、投機を抑制する命令の使用法に関する指針を発表する予定です。

メモリーの一義化 (「投機ストアのバイパス」で説明) は、メモリーからのロードを伴う場合、理論的にはこのような投機の抑制に影響を与える可能性があります。

以下の例では、`CMOVG` 命令を挿入することで、配列境界を超える場所からのデータでサイドチャネルが作られるのを防ぎます。

```
CMP RDX, [array_bounds]
JG out_of_bounds_input
MOV RCX, 0
MOV RAX, [RDX + 0x400000]
CMOVG RAX, RCX
<Further code that causes cache movement based on RAX value>
```

例えば、`array_bounds` の値は `0x20` ですが、この値は `array_bounds` に格納されたばかりで、以前は `0xFFFF` のように高い値が格納されていたとします。プロセッサーは、メモリーの一義化メカニズムにより、`0xFFFF` をロードされた値として使用して、`CMP` 命令を投機実行できます。この命令は、最終的に正しい配列境界の `0x20` で再実行されます。理論的に上記のシーケンスは、アドレスを `0x20` 以下に制限する代わりに、`0xFFFF` までのアドレスにあるメモリーに関する情報を明らかにするサイドチャネルの作成を可能にします。

# 4. 投機ストアのバイパスの概要

多くのインテル® プロセッサーは、ロードアドレスが先行するストアアドレスとオーバーラップするかどうかが判明する前にロードを投機実行する、メモリーの一義化予測器を使用しています。これは、ロードの実行準備が整った時点でストアアドレスが不明な場合に使用されます。プロセッサーが、ロードアドレスがストアアドレスとオーバーラップしないと予測した場合、ロードは投機実行されます。しかし、実際にオーバーラップしていた場合は、ロードが古いデータを読み取る可能性があります。その場合、プロセッサーは正しい結果を得るためロードを再実行します。

メモリーの一義化予測器を利用することで、攻撃者は特定の命令を投機実行させ、その結果をサイドチャネル解析に利用できます。例えば、次のシナリオについて考えてみます。

鍵 `K` が存在するとします。攻撃者は、`M` の値を知ることはできますが、鍵 `K` の値を知ることはできません。`X` はメモリー上の変数です。

```
1. X = &K;  // 攻撃者はポインター X に格納された K のアドレスを持つ変数を得ることができる
<この間でいくつかの命令を実行…>
2. X = &M;  // M のアドレスをポインター X にストアする
3. Y = Array[*X & 0xFFFF];  // M[15:0] で指定されたインデックスの配列からロードするため、
                            // ポインター X にある M のアドレスを逆参照する
```

上記のコードを実行すると、ステップ 3 で発生するアドレス `X` からのロードが投機実行され、メモリーの一義化により、最初に `M` のアドレスではなく `K` のアドレスの値を受け取ることがあります。`K` のアドレスの値が逆参照されると、`M[15:0]` ではなく、`K[15:0]` のインデックスで配列が投機的にアクセスされます。その後、CPU はアドレス `X` からロードを再実行して、`M[15:0]` を配列のインデックスとして使用します。攻撃者は、以前に行われた投機的な配列へのアクセスによって生じたキャッシュの動作を分析して、`K[15:0]` に関する情報を推測することができます。

## 4.1. 投機ストアのバイパスの緩和メカニズム

インテルは投機ストアのバイパスの緩和策を開発しました。この問題はソフトウェアを変更することで緩和できますが、それが難しい場合は、投機ストアのバイパスの無効化 (SSBD: Speculative Store Bypass Disable) を使用することで、すべての古いストアのアドレスが判明するまでロードが投機実行されるのを防ぐことができます。

### 4.1.1. ソフトウェアベースの緩和策

投機ストアのバイパスは、ソフトウェアベースのさまざまなアプローチによって緩和できます。このセクションでは、そのようなソフトウェアベースの緩和策として、プロセスの分離と選択的な `LFENCE` の使用について説明します。

#### 4.1.1.1. プロセスの分離

1 つのアプローチは、すべての秘匿情報を信頼できないコードとは別のアドレス空間に移動させることです。例えば、ウェブサイトごとに別のプロセスを作成し、あるウェブサイトの秘匿情報が、悪意があるかもしれない別のウェブサイトのコードと同じアドレス空間にマッピングされないようにします。同様の手法は、言語ベースの

セキュリティーに依存するほかのランタイム環境でも使用可能で、信頼できるコードと信頼できないコードを同じプロセス内で実行することができます。これは、信頼できるコードが操作されてサイドチャネルが作成されるのを防ぐ深層部の防御としても有効です。このような分離を実現するには、保護キーも有効です。インテルの投機実行サイドチャネル攻撃に関するホワイトペーパー (英語) の「Protection Keys」セクションを参照してください。

#### 4.1.1.2. LFENCE を使用した投機ロード実行の制御

ソフトウェアは、ストア (例えば、セクション 4.1 のステップ 2 の `M` のアドレスのストア) とそれに続くロード (例えば、セクション 4.1 のステップ 3 の X を逆参照するロード) の間に `LFENCE` を挿入して、先行するストアアドレスが判明する前にロードが実行されるのを防ぐことができます。`LFENCE` は、ロードと、以降の返されたデータのサイドチャネルを生成する可能性のある使用 (例えば、セクション 4.1 のステップ 3 の `Array` へのアクセス) の間に挿入できます。ソフトウェアはこの緩和策を広範囲に適用すべきではなく、攻撃者がメモリー位置の古い値を制御できる場合、ストアアドレスが判明する前にロードが実行される可能性がある場合、機密性の高いメモリーの内容を明らかにするディスクロージャー・ガジェットが存在する場合など、悪用される現実的なリスクがある場合にのみ適用すべきです。

### 4.1.2 投機ストアのバイパスの無効化 (SSBD)

前述のソフトウェアベースの緩和策が実行できない場合は、SSBD (投機ストアのバイパスの無効化) を使用して投機ストアのバイパスを緩和できます。

SSBD を設定すると、ロードはすべての古いストアのアドレスが判明するまで投機実行されません。これにより、同じ論理プロセッサー上の古いストアをバイパスして、ロードが古いデータ値を投機的に読み取ることがなくなります。

#### 4.1.2.1. 基本サポート

ソフトウェアは、IA32_SPEC_CTRL.SSBD を 1 に設定することで、論理プロセッサー上の投機ストアのバイパスを無効にできます。

エンクレーブと SMM コードはどちらも、MSR ビットの実際の値に関係なく、SSBD が設定されているかのように動作します。プロセッサーは、エンクレーブまたは SMM コード内のロードが、同じ論理プロセッサー上の古いストアをバイパスしたことにより、古いデータ値を投機的に読み取らないようにします。

#### 4.1.2.2. ソフトウェア使用ガイドライン

SSBD を有効にすることで、投機ストアのバイパスを利用した攻撃を防ぐことができます。ただし、パフォーマンスが低下する可能性があります。以下は、この緩和策を使用する際の推奨事項です。

- 言語ベースのセキュリティー・メカニズムに依存するアプリケーションや実行ランタイムでは、ソフトウェアで SSBD を設定することを推奨します。例として、マネージドランタイムやジャストインタイム・トランスレーターなどが挙げられます。プロセス分離を使用している場合など、ソフトウェアが言語ベースのセキュリティー・メカニズムに依存していない場合、SSBD を設定する必要はないかもしれません。
- 現時点では、言語ベースのセキュリティーに依存しないオペレーティング・システム、仮想マシンモニター、その他のアプリケーションにおける実用的な悪用の報告はありません。言語ベースのセキュリ

ティー・メカニズムの外で SSBD を設定するかどうかを決定する際は、特定のセキュリティー要件を考慮することを推奨します。

これらの推奨事項は将来更新される可能性があります。

インテル® HT テクノロジーが有効で、拡張 IBRS をサポートしていないインテル® Core™ プロセッサーおよびインテル® Xeon® プロセッサーでは、論理プロセッサーで SSBD を設定すると、同じコアのほかの論理プロセッサーのパフォーマンスに影響を与える可能性があります。このようなプロセッサーでは、アイドル状態のときに SSBD MSR ビットをクリアすることを推奨します。

オペレーティング・システムは、プロセスが SSBD 緩和策による保護を要求できる API を提供する必要があります。

仮想マシンモニターは、ゲストが IA32_SPEC_CTRL に直接アクセスすることで、SSBD 緩和策を有効にするかどうかを判断できるようにすべきです。

# 5. CPUID の列挙とアーキテクチャー MSR

新しい緩和メカニズムに対するプロセッサーのサポートは、CPUID 命令といくつかのアーキテクチャー MSR で列挙されます。

## 5.1. CPUID による列挙

CPUID 命令は、CPUID.(EAX=7H,ECX=0):EDX の 5 つの機能フラグを使用して緩和メカニズムのサポートを列挙します。

- CPUID.(EAX=7H,ECX=0):EDX[26] は、間接分岐の投機実行制限 (IBRS) と間接分岐予測器バリア (IBPB) のサポートを列挙します。このビットが設定されているプロセッサーは、IA32_SPEC_CTRL MSR と IA32_PRED_CMD MSR をサポートします。これらは、ソフトウェアで IA32_SPEC_CTRL[0] (IBRS) と IA32_PRED_CMD[0] (IBPB) を設定可能にします。
- CPUID.(EAX=7H,ECX=0):EDX[27] は、STIBP (シングルスレッド間接分岐予測器) のサポートを列挙します。このビットが設定されているプロセッサーは、IA32_SPEC_CTRL MSR をサポートします。これは、ソフトウェアで IA32_SPEC_CTRL[1] (STIBP) を設定可能にします。
- CPUID.(EAX=7H,ECX=0):EDX[28] は、`L1D_FLUSH` のサポートを列挙します。このビットが設定されているプロセッサーは、IA32_FLUSH_CMD MSR をサポートします。これは、ソフトウェアで IA32_FLUSH_CMD[0] (L1D_FLUSH) を設定可能にします。
- CPUID.(EAX=7H,ECX=0):EDX[29] は、IA32_ARCH_CAPABILITIES MSR のサポートを列挙します。
- CPUID.(EAX=7H,ECX=0):EDX[31] は、投機ストアのバイパスの無効化 (SSBD) のサポートを列挙します。このビットが設定されているプロセッサーは、IA32_SPEC_CTRL MSR をサポートします。これは、ソフトウェアで IA32_SPEC_CTRL[2] (SSBD) を設定可能にします。

緩和メカニズムは、マイクロコード・アップデートをロードすることでプロセッサーに導入されることがあります。その場合、ソフトウェアは、マイクロコード・アップデートをロードした後に列挙を再評価する必要があります。

| 表 1. 構造化された拡張機能フラグの列挙リーフ (出力は ECX 入力値に依存) | | |
|---|---|---|
| **EAX の初期値** | **プロセッサーに関する情報提供** | **説明** |
| 07H | EDX | リーフ 07H メインリーフ (ECX = 0)。<br>ECX に無効なサブリーフ・インデックスが含まれる場合、EAX/EBX/ECX/EDX は 0 を返します。 |
| | | ビット 10: MD_CLEAR をサポート<br>ビット 25-00: 予約済み<br>ビット 26: IBRS および IBPB をサポート<br>ビット 27: STIBP をサポート<br>ビット 28: L1D_FLUSH をサポート<br>ビット 29: IA32_ARCH_CAPABILITIES をサポート<br>ビット 30: 予約済み<br>ビット 31: SSBD をサポート |

上記の表はこのリーフの概要を示すものです。CPUID リーフ 07H の詳細は、『Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A』の「CPUID instruction」を参照してください。

## 5.2. IA32_ARCH_CAPABILITIES MSR

追加機能は、IA32_ARCH_CAPABILITIES MSR (MSR インデックス 10AH) によって列挙されます。これは、CPUID.(EAX=7H,ECX=0):EDX[29] が 1 の場合にサポートされる読み取り専用の MSR です。

| 表 2. IA32_ARCH_CAPABILITIES MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスターアドレス Hex** | **レジスターアドレス DEC** | **レジスター名/ビット・フィールド** | **ビットの説明** | **コメント** |
| 10AH | 266 | IA32_ARCH_CAPABILITIES | アーキテクチャーの特徴の列挙 (RO) | CPUID.(EAX-07H, ECX=0):EDX[29]=1 の場合 |
| 10AH | 266 | 0 | RDCL_NO: プロセッサーは RDCL (Rogue Data Cache Load) の影響を受けません [5]。 | |
| 10AH | 266 | 1 | IBRS_ALL: プロセッサーは拡張 IBRS (間接分岐の投機実行制限) をサポートします。 | |
| 10AH | 266 | 2 | RSBA: プロセッサーは代替 RSB をサポートします。RSB が空の場合、RET 命令で別の分岐予測器を使用できます。retpoline を使用するソフトウェアは、この動作の影響を受ける可能性があります。 | |
| 10AH | 266 | 3 | SKIP_L1DFL_VMENTRY: 値が 1 の場合、ハイパーバイザーは VM エントリーで L1D をフラッシュする必要がないことを示します。 | |
| 10AH | 266 | 4 | SSB_NO: プロセッサーは SSB (投機ストアのバイパス) の影響を受けません [6]。 | |
| 10AH | 266 | 5 | MDS_NO: プロセッサーは MDS (マイクロアーキテクチャー・データ・サンプリング) の影響を受けません。 | |

| 表 2. IA32_ARCH_CAPABILITIES MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスター アドレス Hex** | **レジスター アドレス DEC** | **レジスター名/ ビット・フィールド** | **ビットの説明** | **コメント** |
| 10AH | 266 | 6 | IF_PSCHANGE_MC_NO: TLB を無効にせずにコードページのサイズを変更することで、プロセッサーがマシン・チェック・エラーになることはありません。 | |
| 10AH | 266 | 7 | TSX_CTRL: プロセッサーは RTM_DISABLE と TSX_CPUID_CLEAR をサポートします。 | |
| 10AH | 266 | 8 | TAA_NO: プロセッサーは、インテル® トランザクショナル・シンクロナイゼーション・エクステンション (インテル® TSX) の非同期アボート (TAA) の影響を受けません。 | |
| 10AH | 266 | 63:9 | 予約済み | |

## 5.3 IA32_SPEC_CTRL MSR

IA32_SPEC_CTRL MSR ビットは、論理プロセッサーのスコープとして定義されています。一部のコアの実装では、このビットが同じコア上のほかの論理プロセッサーに影響を与えることがあります。

この MSR は、リセット後は値が 0 になり、INIT# や SIPI# の影響を受けません。

IA32_TSC_DEADLINE MSR (MSR インデックス 6E0H)、x2APIC MSR (MSR インデックス 802H ~ 83FH)、IA32_PRED_CMD MSR (MSR インデックス 49H) と同様に、IA32_SPEC_CTRL MSR (MSR インデックス 48H) への WRMSR はシリアル化命令として定義されていません。

IA32_SPEC_CTRL MSR への WRMSR は、先行するすべての命令がローカルで完了するまで実行されず、WRMSR が完了するまで後続の命令は実行を開始しません。

| 表 3. IA32_SPEC_CTRL MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスター アドレス Hex** | **レジスター アドレス DEC** | **レジスター名/ ビット・フィールド** | **ビットの説明** | **コメント** |
| 48H | 72 | IA32_SPEC_CTRL | 投機制御 (R/W) | 定義されたビット・フィールド位置の列挙条件のいずれかが成立する場合。 |
| 48H | 72 | 0 | IBRS。間接分岐の投機実行を制限します。 | CPUID.(EAX=07H, EXC=0):EDX[26]=1 の場合。 |
| 48H | 72 | 1 | シングルスレッド間接分岐予測器 (STIBP)。コア上のすべての論理プロセッサーの間接分岐予測が、同じコアのほかの論理プロセッサーによって制御されることを防ぎます。 | CPUID.(EAX=07H, ECX=0):EDX[27]=1 の場合 [7]。 |
| 48H | 72 | 2 | 投機ストアのバイパスの無効化 (SSBD) は、すべての古いストアアドレスが判明するまでロードの投機実行を遅延させます。 | CPUID.(EAX=07H, ECX=0):EDX[31]=1 の場合 [8]。 |

| 表 3. IA32_SPEC_CTRL MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスター アドレス Hex** | **レジスター アドレス DEC** | **レジスター名/ ビット・フィールド** | **ビットの説明** | **コメント** |
| 48H | 72 | 63:4 | 予約済み | |

## 5.4 IA32_PRED_CMD MSR

IA32_PRED_CMD MSR は、ソフトウェアが予測器の状態に影響を与えるコマンドを発行できるようにします。

| 表 4. IA32_PRED_CMD MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスター アドレス Hex** | **レジスター アドレス DEC** | **レジスター名/ ビット・フィールド** | **ビットの説明** | **コメント** |
| 49H | 73 | IA_PRED_CMD | 予測コマンド (WO) | 定義されたビット・フィールド位置の列挙条件のいずれかが成立する場合。 |
| 49H | 73 | 0 | 間接分岐予測バリア (IBPB) | CPUID.EAX=07H, ECX=0):EDX[26]=1 の場合。 |
| 49H | 73 | 63:1 | 予約済み | |

IA32_TSC_DEADLINE MSR (MSR インデックス 6E0H)、X2APIC MSR (MSR インデックス 802H～83FH)、IA32_SPEC_CTRL MSR (MSR インデックス 48H) と同様に、IA32_PRED_CMD MSR (MSR インデックス 49H) への `WRMSR` はシリアル化命令として定義されていません。

IA32_PRED_CMD MSR への `WRMSR` は、先行するすべての命令がローカルで完了するまで実行されず、`WRMSR` が完了するまで後続の命令は実行を開始しません。

## 5.5 IA32_FLUSH_CMD MSR

IA32_FLUSH_CMD MSR は、ソフトウェアがほかのアーキテクチャー手法よりも細かい粒度で構造を無効にできるようにします。

IA32_TSC_DEADLINE MSR (MSR インデックス 6E0H)、X2APIC MSR (MSR インデックス 802H～83FH)、IA32_SPEC_CTRL MSR (MSR インデックス 48H) と同様に、IA32_FLUSH_CMD MSR (MSR インデックス 10BH) への `WRMSR` はシリアル化命令として定義されていません。

IA32_FLUSH_CMD MSR への `WRMSR` は、先行するすべての命令がローカルで完了するまで実行されず、`WRMSR` が完了するまで後続の命令は実行を開始しません。

| 表 5. IA32_FLUSH_CMD MSR の詳細 | | | | |
|---|---|---|---|---|
| **レジスター アドレス Hex** | **レジスター アドレス DEC** | **レジスター名/ ビット・フィールド** | **ビットの説明** | **コメント** |
| 10BH | 267 | IA_FLUSH_CMD | フラッシュコマンド (WO) | 定義されたビット・フィールド位置の列挙条件のいずれかが成立する場合。 |
| 10BH | 267 | 0 | `L1D_FLUSH`: L1 データキャッシュをライトバックして無効にします。 | CPUID.EAX=07H, ECX=0):EDX[28]=1 の場合。 |
| 10BH | 267 | 63:1 | 予約済み | |

# 脚注

1. 次のシーケンシャル命令をターゲットにした `CALL` は、ディスプレイスメントがゼロになります。
2. `INIT#` による特権レベルの高い predictor モードへの移行はこの例外であり、新しい predictor モードで実行される間接分岐予測ターゲットが、特権レベルの低い predictor モードで動作するソフトウェアによって制御されるのを防げないことがあります。
3. RSB 上書きシーケンスとは、非ゼロのディスプレイスメントのニア `CALL` 命令が、ニア `RET` よりも 32 個多く含まれる命令シーケンスのことです。
4. 間接分岐には、ニアコール間接分岐、ニアジャンプ間接分岐、ニアリターン分岐命令が含まれます。ニアリターンも含まれるため、IBPB コマンドの前に作成された RSB エントリーは、同じ論理プロセッサー上でコマンドの後に実行されるリターンの予測ターゲットを制御できません。
5. インテルの投機実行サイドチャネル攻撃に関するホワイトペーパー (英語) の「Protection Keys」セクションを参照してください。
6. インテルの投機実行サイドチャネル攻撃に関するホワイトペーパー (英語) の「Speculative Store Bypass」セクションを参照してください。
7. IA32_SPEC_CTRL MSR をサポートし、STIBP をサポートしないプロセッサー (CPUID.(EAX=07H, ECX=0):EDX[27:26] = 01b) では、STIBP (ビット 1) を設定しても例外は発生しません。
8. IA32_SPEC_CTRL MSR をサポートし、SSBD をサポートしない将来のプロセッサー実装では、SSBD (ビット 2) を設定しても例外は発生しません。これは、IA32_SPEC_CTRL MSR をサポートするすべての現行のプロセッサーにおいて、特に最新のマイクロコード・アップデートがロードされていない場合には当てはまらないかもしれません。

---

### 製品および性能に関する情報

[1] 性能は、使用状況、構成、その他の要因によって異なります。詳細については、www.Intel.com/PerformanceIndex/ (英語) を参照してください。