今更学ぶ、CPU脆弱性

JP3BGY

前提知識

- デジタル回路及びCPUの基本的な構造
  - パイプライン
  - 権限(Intelだとring protection)
  - MMU
  - 割り込み
- OS Kernelの仕組み
  - システムコール

CPU脆弱性

- 2017年に発見、2018年に発表されたSpectre/Meltdown以降サイドチャネル攻撃が2020年ぐらいまでかなり発見された
  - 最近は多く見つかってるわけではないが、2022年にVUSecがSpectre-BHBを発見

- 今回は皮切りになったSpectre/Meltdownについて軽く説明

前提知識：Out-of-Order実行（以降OoO）

- 演算回路を複数持たせて並列化(スーパースカラ)
  - 1サイクルで複数命令動かせる
  - 順番通りに実行するとデータの依存関係で実はあまり並列実行できない
- IO待ちによるCPUの無駄な待ち時間
  - メモリアクセスは数十クロック、キャッシュでも数クロック
  - できる限り待ち時間を減らしたい

1.AにB+Cを代入
2.CにA*メモリXを代入
3.DにE*Fを代入←

- 命令列は全部が全部データの依存関係があるわけじゃない
  - なので順番を入れ替えられれば上記の課題が解決する！
- あくまで計算の順序を変えるだけで、計算結果の書き込みは順序通り
  - レジスタとは別に計算結果を保持して、順番が来たらレジスタやメモリに反映する

前提知識：投機的実行

- パイプライン化やOoOによって命令が完全に実行が終わる前に数十命令が読み込まれるようになった
  - 条件分岐があると、たいてい条件の計算結果が計算される前に読み込まれるから飛ぶ先がわからない

- 100%正しくはないが、そこそこの精度で分岐先が予測できれば無駄な時間を減らせる！
  - 100%ではないうえに数十命令程度で短期だから”投機的”

OoO&投機的実行がだす副作用

- OoOはあくまで計算処理の順序を変えるだけで計算結果の書き込みの順序は変えない
  - というのは実はちょっと嘘
  - 計算処理に使うデータはメモリキャッシュに乗っかってしまう
  - あくまで変わらないのは外から直接アクセスできるデータのみ

- つまりキャッシュのデータから計算結果の書き込みがされなかった命令の様子が観測できる
  - 少なくとも読み書きが可能なメモリ領域であればアクセス時間を計ることでキャッシュに載ってるかがすぐにわかる
  - 例外処理で止まった命令の様子を見ることができる.....?

Meltdown(variant 3)

- OoOで書き出されなくても、実行した命令がわかるなら権限のないメモリの値もその方法でわかるんじゃね？
  - 実際にやったら読めちゃったのがMeltdown　　x=buf[(*kerneladdr&1)*1000]
- 権限の有無の確認よりも先にメモリへのアクセスが走ってしまうために発生してしまった脆弱性
  - AMDのCPUには存在していないらしい
- ちなみにメモリだけじゃなくて特権レジスタの値とかも読める(v3a)

Meltdownの対策

- ユーザープログラムを実行してるときにOS空間のページテーブルって必要ないんじゃない？
  - KPTI
  - システムコール等割り込みに必要な最低限のメモリ空間だけを残して他のテーブルを全削除
- 最近のCPU(Intelだと10世代前後から)ではハードウェアレベルで修正されているらしい
  - Intelの発表 等参照

Spectre variant 1

- arr[user_input]の1bitを user_input>=nで推測可能
  - 分岐予測のアルゴリズムを悪用して、ifの中に入るように予測させる
  - arr2のキャッシュの状態から値の判別ができる

```
size_t user_input;
char arr[n],arr2[m];
if(user_input < n){
    size_t index = (arr[user_input]&1)*0x1000;
    if(index < m){
        char value = arr2[index];
    }
}
```

Spectre v1の悪用方法

- ただ無理やり範囲外参照ができるコードが作れるだけなら悪用できないのでは？
  - No
  - JIT

- ブラウザやOSに組み込まれた言語処理系は内部でJIT機能を持つ
  - JIT機能を悪用して外から先ほどのコード辺のようなものを実行させることができる

Spectre v1の対策

- ブラウザはSite Isolationと呼ばれる、サイトごとに異なるプロセスでブラウザを動かすという機能を導入
  - Spectre v1が読めるのはあくまで同じプロセス内
- JITごとの個別の対応
  - LFENCEと呼ばれる投機的実行を抑制する命令がある
  - ビットマスクで範囲外参照できないようにする
- 根本の脆弱性はまだ修正されてない
  - JITの対応がちょっと杜撰で漏れてたという話もそこそこあり、100%安全とは言えない状況
    - 悪用する側も難易度が高いのでおそらくは大丈夫・・・なはず

Spectre variant 2

- 間接呼び出しの分岐予測が同じコアであれば共有されている
  - 予測先を先のコード断片にすれば仮想マシンなども乗り越えられる？
    - 乗り越えられちゃった

call %rax

- Linux KernelのeBPFコードを用いればゲストからホストのデータが読める
  - eBPFコードはゲストにおいて、分岐予測結果のダンプからゲストメモリがホストメモリのどこに置いてあるかを推論してeBPFコードを注入する

Spectre v2の対策

- Intelにおいてret命令の分岐予測はjmp,callとは異なる予測が用いられるため、ret命令で間接呼び出しを行うretpolineがコンパイラに導入されてる
  - ちなみにARMは無理

- Intelの最近のCPUは権限をまたいだ分岐予測テーブルの使い回しを防止(Enhanced IBRS)やハイパースレッドでの共有防止(STIPB)、分岐予測を消す命令などが導入されている
  - 細かい対策は基本v1と変わらない
  - 完全に穴がふさがれたわけではない

その他のお話

- CPU脆弱性はSpectre/Meltdownの派生が多い
  - Wiki の投機的実行の脆弱性リストなどを参照
  - 今回学んだことは割と他の論文・脆弱性を見るのにも使える

- 他のタイプのCPU脆弱性
  - 電圧を意図的に下げてSGXのデータをリーク(Plundervolt)
  - secretによって実行時間が異なることからsecretを当てる(tpmfail)
  - 電圧のデータから暗号化キーを復元する(PLATYPUS)

Thank you!