Spectre и Meltdown

Начнем с истории, в нюне 2017 года, производителям процессоров (intel, amd, arm), прислали информацию о нахождении в архитектуре процессоров, фундаментальных дыр. Начинаются закрытые обсуждения в компаниях по исправлению ситуации.

Первая публикация данного инцидента в сми была сделана в январе следующего года. Производители ОС выпускают патч, который препятствует воспроизведению данных уязвимостей.

После этого эксперты гугл пишут статью где они успешно смогли реализовать данные уязвимости, так благодаря spectre мы можем читать виртуальную память других процессов, а благодаря Meltdown – появляется возможность доступа чтения привилегированной памяти ядра.

Обе уязвимости схожи на логическом уровне и предоставляют возможность проведения атаки по стороннему каналу – кэш процессора. Meltdown использует задержку в обработке исключений. Spectre – обманывает блоки предсказания.

Изначально процессоры выполняли команды последовательно – Процессор с последовательным выполнением команд начинает выполнение очередного цикла команды только после того, как будет закончен предыдущий, то есть в каждый момент времени выполняется только одна команда., однако такой подход имеет ряд минусов

• допустим у нас есть две команды первая обращается к памяти и из за этого очень долго находится на процессоре, а вторая маленькая – все это время ждет

• Ступор конвеера - команды ждут результаты друг друга

Спекулятивное выполнение команд

Основан на том, чтобы не ждать долгих расчетов предыдущих команд, а выполнять то что можно выполнить. Так после декодера инструкций появился буфер переупорядочивания микроопераций.

Таблицы страниц

Содержат сопоставления между виртуальной памятью, которую использует процессор и физической памятью, которую использует диспетчер памяти. Код ядра современной ОС отображается в виртуальное адресное пространство пользовательского процесса. В нормальных условиях память ядра не может быть прочитана из пользовательского пространства, будет создано исключение.

Meltdown

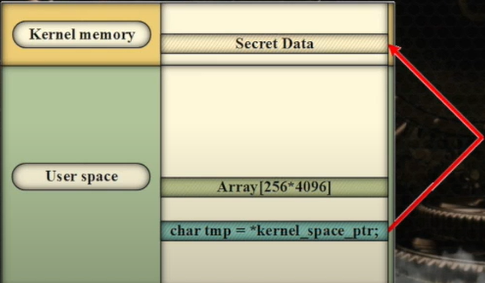
Цель нарушителя – чтение привилегированной памяти из пользовательской программы

Ограничения – доступ к памяти только своего виртуального адресного пространства

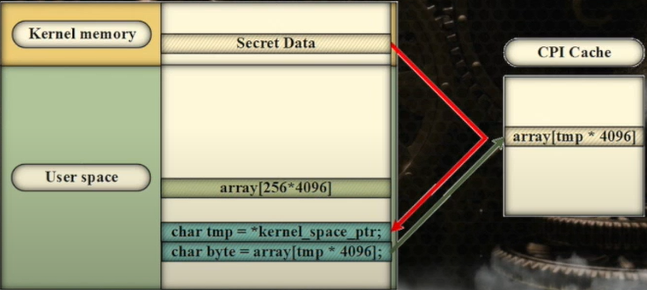
Все проверки на легитимность происходят на заключительном этапе – из за спекулятивного вычисления

Атака

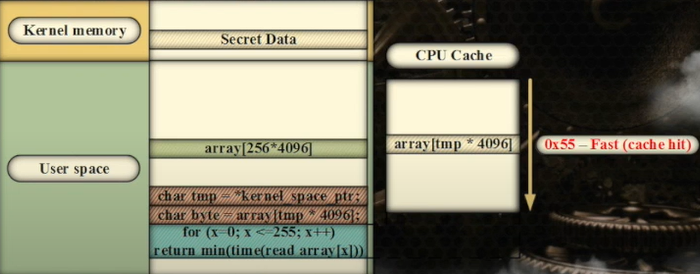
1. Нарушитель создает большой массив, к которому мы не обращаемся, значит он не попал в кэш. Программа пытается прочитать байт из памяти ядра (который нельзя читать), поэтому перед этим вставляется какая то сложная арифметика, которое требует большое количество обращений к памяти. Пока мы ждем – начинаем спекулятивно тянуть данные.



1. Выборка байта из массива, где в качестве индекс выступает байт из памяти ядра



1. После этого процессор понимает что мы обратились в память ядра генерит исключение и останавливает процесс, однако байт, который мы вытянули находится в кэше процессора, и мы можем навесить обработку исключения которая будет замерять время доступа к каждому байту нашего массива, и тот байт который мы вытянули будет найден быстрее всех. Делая так для каждого байта в ядре мы можем получить полный дамп ядра.



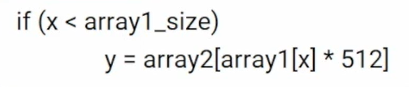
Был выпущен патч, который удаляет отображение ядра из польщовательского окружения. Поскольку ядро боль не отображается его память нельзя прочитать с помощью meltdown. Однако это сильно снизило производительность системы.

**Spectre**

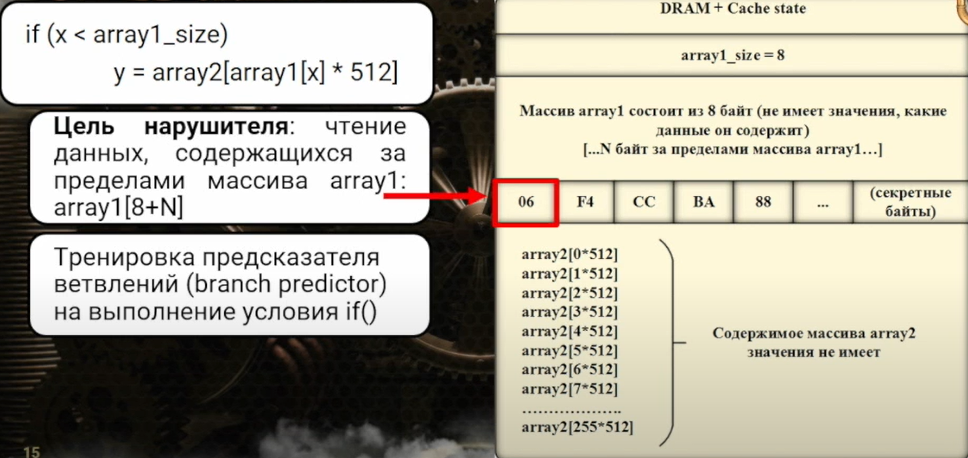
Пытается обмануть предсказатель ветвлений.

Предсказатель ветвлений - Устройство, входящее в состав микропроцессоров, имеющих конвейерную архитектуру, предсказывающее, будет ли выполнен условный переход в исполняемой программе. Предсказание ветвлений позволяет сократить время простоя конвейера за счёт предварительной загрузки и исполнения инструкций, которые должны выполниться после выполнения инструкции условного перехода.

Допустим у нас есть такой кусок кода, мы должны манипулировать переменной x, если бы выполнение было последовательным, то мы бы постоянно упирались в проверку.



Также допустим у нас есть какая то программа с уязвимым кусом кода



Мы пишем максимально похожий код и начинаем иренеровать предсказатель ветвлений, так запустив данный код много раз он поймет что почти 100% будет так.

Далее нам необходимо исключить второй массив и размер первого массива из кэша для того чтобы процессор выполнил спекулятивное вычисление, т.к. если оно будет в кэше то проверка пройдет быстрее, а нам этого не надо.

Нарушитель запускает свой код и подает на вход неправильный x, мы начинаем чтение байта по данному адресу и добавляем его в кэш. После этого все как в meltdown.

Методы защиты от spectre

Производители процессов решили добавить новый моделезависимый регистр, который ограничивает выполнение косвенных переходов, ограничивает обучение предсказателя переходов, и умеет его очищать. Однако тем самым это сделало процессоры более медленными.

Retpoline -