САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе №6

«SPECTRE»

Выполнил(а): Султанов Мирзомансурхон Махсудович

студ. гр. М313Д

Санкт-Петербург

2020

Цель работы: знакомство с аппаратной уязвимостью Spectre.

Инструментарий и требования к работе: рекомендуется использовать C, C++.

Теоретическая часть

Meltdown и Spectre – это легко запоминающиеся имена близких по своим особенностям аппаратных уязвимостей в современных процессорах. На большинстве процессорах существует speculative execution или же русскими словами спекулятивное выполнение. Это метод оптимизации, при котором компьютерная система выполняет некоторую задачу, которая может не понадобиться заранее, когда есть свободный конвейер, чтобы обеспечить более эффективный параллелизм. Именно из-за этой технологии и появились на свете Meltdown и Spectre.

Общая идея уязвимости Spectre заключается в следующем: хакер считывает приватные данные через сторонний канал в виде общей иерархии кэш-памяти. С Meltdown же всё посерьёзнее. Хакер, используя ошибку спекулятивного выполнения, может игнорировать права доступа к страницам. Meltdown, по сути, крадёт данные из самого ядра операционной системы, а Spectre же влияет именно на программы. Однако на практике именно Spectre сейчас наиболее опасен, так как противодействовать им намного сложнее, чем Meltdown.

Вот как именно реализуется уязвимость Spectre: на высоком уровне атаки Spectre нарушает границы изоляции памяти, используя при этом спекулятивное выполнение и получение приватных данных по микроархитектурным скрытым каналам. Более конкретно, чтобы организовать атаку Spectre, злоумышленник начинает с обнаружения или введения последовательности инструкций в адресном пространстве процесса, которая при выполнении действует как скрытый передатчик

канала, который пропускает память жертвы или содержимое регистра. Затем злоумышленник обманывает процессор, заставляя его спекулятивно и ошибочно выполнять эту последовательность команд, тем самым и происходит утечка информации жертвы по тайному каналу. Наконец, злоумышленник получает информацию о жертве по тайному каналу.

Вот пример реализации Spectre с использованием неверного предсказания условных переходов. Рассмотрим следующий код на рисунке 1:

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Рисунок 1 – Пример работы Spectre с условным переходом

Допустим, в x хранится вредителем. Чтобы обеспечить правильность доступа к памяти array1, приведённый выше код содержит оператор if, целью которого является проверка того, что значение x находится в допустимом диапазоне. Теперь, что же необходимо сделать злоумышленнику, чтобы заполучить данные? Сначала ему необходимо “натренировать” предсказатель ветвей ожидать, что if будет истинным. Затем злоумышленнику надо вызвать значение x, которое выходит за диапазон array. Не дожидаясь результата if, процессор будет подготавливать результат на случай, если if - true. Заметим, что чтение из array2 загружает данные в кэш на адрес, который зависит от массива array1[x], который в свою очередь зависит от x. Изменяя x, можно регулировать то, к каким строкам кэша обращается программа. Когда же будет получено, что if – на самом деле false, состояние кэша не будет возвращено. Однако изменения будут сохранены и их вполне можно извлечь из кэша. Вот здесь и появляются проблемы для пользователей.