Полховский Артем Федорович, гр.990541

Spectre, Meltdown

История уязвимостей.

В декабре 2017 года исследователь из Грацского технического университета Майкл Шварц обратился в компанию Intel с сенсационной, как он полагал, информацией. Ему и его коллегам Дэниэлу Груссу, Моритцу Липпу и Стефану Мангарду удалось создать эксплойт, с помощью которого можно добыть информацию из внутренней памяти процессоров.

Через девять дней Шварцу позвонили. Кто-то из Intel сообщил ему, что компании уже известно о проблемах с процессорами и сейчас она работает над их устранением. Более того, Intel старательно скрывает эту информацию и делает всё для того, чтобы она не стала известна посторонним. Шварца поблагодарили и сказали, что он раскрыл секрет, который будет обнародован в назначенный день.

Проблема, которую обнаружил Шварц и несколько других людей независимо друг от друга, затрагивала почти все процессоры, выпущенные за последние два десятилетия. Она настолько фундаментальна, что не может быть полностью решена в короткий срок. Устранять её совместными усилиями должны производители процессоров, разработчики программного обеспечения, а также крупнейшие владельцы серверов и облачных сервисов. Обычно, когда исследователь безопасности обращается к вендору с информацией о проблемах в оборудовании или ПО, он даёт несколько месяцев на её устранение. Затем он имеет право обнародовать информацию, и тогда она становится известна хакерам, которые создают эксплойты и подвергают опасности пользователей.

В данном случае проблема затрагивала интересы очень многих компаний, и удержать её в секрете было бы очень трудно. Слишком многим людям было необходимо рассказать об уязвимостиях, а они должны были тайно и синхронно подготовить «заплатки» для своих продуктов. В какой-то момент кое-кто нарушил молчание, из-за чего об уязвимостях стало известно до того, как эти они были закрыты.

В сети начала появляться противоречивая информация (например, о том, подвержены ли уязвимостям процессоры AMD), а некоторые антивирусы по ошибке блокировали установку важных патчей безопасности, принимая их за вредоносный код. Часть патчей оказалась сырой, и их распространение пришлось приостановить спустя несколько дней после выпуска. Компания Google создала инструмент Retpoline, с помощью которого можно проверять устройства на наличие уязвимостей, однако из-за эмбарго он был выпущен лишь после публикации официальных сведений о Meltdown и Spectre. Компьютерная группа реагирования на чрезвычайные ситуации (CERT) при Университете Карнеги узнала об уязвимостях из СМИ вместе со всеми, что привело к дополнительной неразберихе. Поначалу группа выпустила доклад, в котором советовала компаниям заменить

процессоры в своих устройствах на более новые. Как только IT-менеджеры представили, насколько затратным будет это решение, они загрустили, однако чуть позже специалисты CERT скорректировали доклад, указав, что менять процессоры бессмысленно, достаточно установить софтверные патчи.

Первое упоминание уязвимостей, которые затем получили название Meltdown и Spectre, датируется июнем 2017 года — именно тогда сотрудник Google Project Zero Янн Хорн взломал собственный компьютер и убедился в реальности самой, пожалуй, серьёзной проблемы процессоров. Результатами эксперимента он поделился с Intel, AMD и ARM. Каждая компания получила подробное описание проблем, а также предупреждение о том, что продукция других вендоров тоже может содержать схожие уязвимости. Хорн также написал, чтобы они не распространяли переданную им информацию, не скоординировавшись со всеми, кого касается эта проблема. Выяснить, кто должен быть в курсе ситуации, было не так просто.

Изначально казалось, что исправлять нужно процессоры, затем стало понятно, что требуется пропатчить операционные системы, то есть информация должна быть передана ещё и их разработчикам. Браузеры и облачные платформы тоже было необходимо защищать, а это означает, что десятки или сотни сотрудников Google, Microsoft, Apple, Amazon и других компаний становились носителями секрета.

Официальная политика Google Project Zero предполагает, что на решение проблем выделяется 90 дней, после чего информация обнародуется независимо от того, закрыты уязвимости или нет. Однако из-за того, что проблема была слишком серьёзной, и все компании устраняли её секретно, слаженно общаясь друг с другом, специалисты Google назначили конкретную дату, к которой всё должно быть готово. 14 декабря клиенты Amazon Web Server получили оповещение о том, что 5 января будет запущена серия перезагрузки серверов, которая скажется на производительности сервиса.

Перед новым годом компания Microsoft подготовила патч безопасности для Windows и серверов, который должен был выйти примерно в то же время. 18 декабря Линус Торвальдс сообщил, что в Linux будет внесено глобальное изменение, связанное с работой процессоров на архитектуре x86. Все обновления Linux публикуются в репозиториях, где коммьюнити может обсуждать внесённые изменения, но на этот раз апдейт был засекречен, подробной информации о нём не было. Компания AMD была не в восторге от того, что патчи от уязвимостей Meltdown и Spectre замедляют процессоры, поэтому попросила разработчиков Linux не менять операционную систему для её чипов. Инженер AMD Том Леданski объяснил это тем, что архитектура процессоров AMD фактически не позволяет перехватывать данные из внутренней памяти микросхем. Даже если такая возможность существует в теории, невозможно устранить её софтверным патчем, не затрагивая стандарты работы процессоров, принятые за последние двадцать лет, требуется более глубинный подход.

Журналисты The Register проанализировали изменения, которые готовят производители процессоров и разработчики операционных систем, и

пришли к выводу, что уязвимости, о которых говорилось запутанным техническим языком, есть только в процессорах Intel. Если говорить простыми словами, опасность Meltdown и Spectre заключается в том, что эти баги позволяют программам добывать данные из памяти процессора и сторонних процессов. Такая возможность появилась из-за особенности работы процессоров, а её устранение приводит к понижению их вычислительной способности.

После публикации The Register в Twitter и других соцсетях стали появляться доказательства существования рабочих эксплойтов, с помощью которых можно извлекать данные из памяти процессора. Журналисты раскрыли тайну, о которой на тот момент ещё нельзя было рассказывать: у Intel, AMD, ARM, Microsoft, Google, Apple и других компаний не было готовых и протестированных решений. Им пришлось спешно дорабатывать «заплатки» и выпускать их без тщательного тестирования, что привело к дополнительным проблемам.

Meltdown и Spectre показали, как изменился мир высоких технологий за последние годы. Раньше уязвимости в каком-либо продукте затрагивали лишь ту компанию, которая его производила, а теперь баг в популярном оборудовании или софте приводит к цепной реакции: требуется устранять не только его, но и перерабатывать кучу другой, связанной продукции. Так происходит из-за того, что компании создают не изолированные продукты, а экосистемы, тесно переплетающиеся друг с другом.

Meltdown. Он затрагивает в первую очередь процессоры Intel, хотя есть подтверждение об уязвимости нескольких процессоров ARM. С помощью Meltdown вредоносный код может злоупотреблять реализациями спекулятивного выполнения Intel и ARM, что приведет к утечке информации из других процессов — напрямую из всезнающего ядра операционной системы. В результате Meltdown можно легко использовать для отслеживания других процессов и похищения информации, которая должна быть изолирована ядром, другими программами или другими виртуальными машинами.

Между тем второй тип эксплойта называется Spectre, а число процессоров, подверженных риску эксплуатации, еще шире. По сути, каждый высокопроизводительный процессор, когда-либо созданный — Intel, AMD, ARM и POWER — считается уязвимым. Подобно Meltdown, атака Spectre злоупотребляет спекулятивным выполнением, чтобы получить информацию, которая должна быть изолирована. Однако особенность Spectre заключается в том, что это более сложная и гораздо коварнейшая атака; в то время как Meltdown основан на злоупотреблении конкретной реализацией спекулятивного выполнения, Spectre можно рассматривать как (ранее неизвестную) фундаментальную уязвимость спекулятивного исполнения, которую теперь можно использовать для атак. Spectre требует больше усилий по подготовке, чтобы принудить целевое приложение к утечке информации, но фундаментальная природа риска означает, что уязвимость Spectre намного

сложнее устранить, и она еще не полностью исследована.

Meltdown разрушает наиболее фундаментальную изоляцию между приложениями и операционной системой. Эта атака позволяет программе получить прямой доступ к памяти, а значит к секретным данным других программ и оперативной системы.

Spectre нарушает изоляцию между различными приложениями. Это позволяет атакующему получить данные даже из программ, которые соблюдают все правила безопасности и работают без ошибок. Фактически, проверки безопасности упомянутых программ увеличивают поверхность атаки и могут сделать приложения более восприимчивыми к Spectre.

Эксплойт Spectre применить более сложно, чем Meltdown, однако нивелировать угрозу такой атаки, соответственно, намного сложнее. Тем не менее возможно предотвратить некоторые уже известные варианты эксплойта установкой последних софтварных патчей.

Meltdown, и Spectre — локальные атаки, требующие выполнения вредоносного кода на целевой машине. Это означает, что эти атаки не являются (напрямую) атакой удалённого запуска кода – как были Nimda или Code Red – а значит, системы нельзя атаковать, просто подключившись к сети. Понятно, что они ближе к атакам эскалации привилегий, классу атак, которые помогают глубже проникнуть в систему (к которой у вас уже есть доступ). Однако, исследователи показали, что они могут выполнять атаки на основе Spectre, используя JavaScript, поэтому веб-браузер может запустить вредоносный файл JavaScript, и таким образом позволить атаку.

Хотя мы и упомянули, что атаки не могут быть проведены удаленно, в случае если Meltdown и Spectre удалось запустить в локальной системе, характер эксплойта заключается в том, что это read-only атаки. То есть, все что они могут – это прочитать информацию из системы. Они не могут напрямую принудительно произвести выполнение кода в ядре ОС, других виртуальных машинах или других программах. И все же атаки раскрытия информации могут быть очень разрушительными в зависимости от того, какая информация получена. Всегда есть риск использования этой информации для проведения успешной атаки уже путем выполнения кода, а значит, угроза все же существует. Но реальный риск заключается именно в использовании этих уязвимостей злоумышленниками для похищения информации, а не для контроля над системой.

Код атаки

```
; rcx = kernel address
; rbx = probe array
retry:
mov al, byte [rcx]
shl rax, 0xc
jz retry
mov rbx, qword [rbx + rax]
```

Теперь по шагам, как это работает.

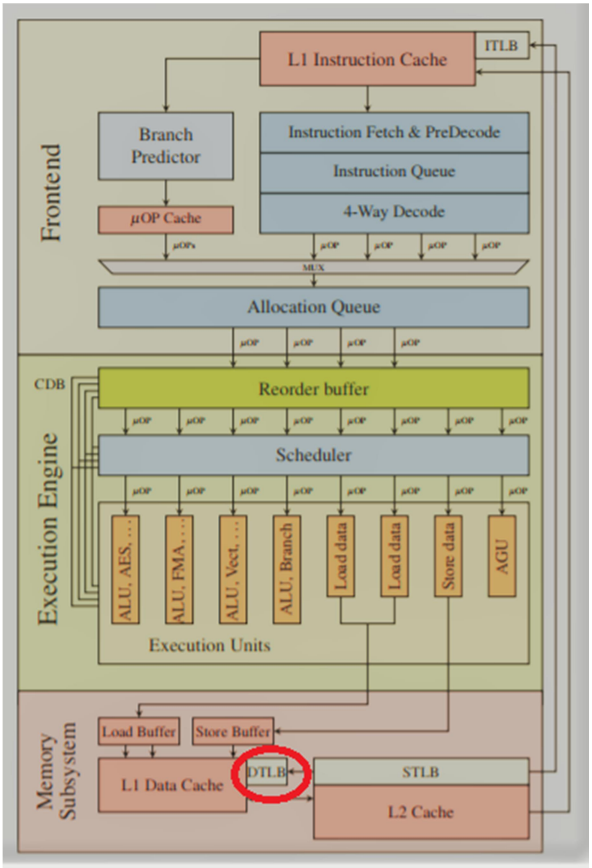
`mov al, byte [rcx]` — собственно чтение по интересующему атакующего адресу, заодно вызывает исключение. Важный момент заключается в том, что исключение обрабатывается не в момент чтения, а несколько позже.

`shl rax, 0xc` — зачение умножается на 4096 для того, чтобы избежать сложностей с механизмом загрузки данных в кэш

`mov rbx, qword [rbx + rax]` — "запоминание" прочитанного значения, этой строкой прогревается кэш

`retry` и `jz retry` нужны из-за того, что обращение к началу массива даёт слишком много шума и, таким образом, извлечение нулевых байтов достаточно проблематично. Честно говоря, я не особо понял, зачем так делать — я бы просто к rax прибавил единичку сразу после чтения, да и всё. Важный момент заключается в том, что этот цикл, на самом деле, не бесконечный. Уже первое чтение вызывает исключение

Уязвимость расположена на рисунке, она выделена красным кружком:



Проблема в блоке TLB, который используется в пейджинговой адресации оперативной памяти. Он является ассоциативным буфером хранящим пары значений виртуальный-физический адрес используемой приложением в оперативной памяти. Это не Кеш данных. Как его описывает Интел:

4.10.2 Translation Lookaside Buffers (TLBs)

A processor may cache information about the translation of linear addresses in translation lookaside buffers (TLBs). In general, TLBs contain entries that map page numbers to page frames; these terms are defined in Section 4.10.2.1. Section 4.10.2.2 describes how information may be cached in TLBs, and Section 4.10.2.3 gives details of TLB usage. Section 4.10.2.4 explains the global-page feature, which allows software to indicate that certain translations should receive special treatment when cached in the TLBs.

Блок TLB нужен для того чтобы сократить время обращения к ОП за счет исключения процедуры трансляции адресов памяти. Такая трансляция производится единственный раз для всей страницы (размером как раз 4К в случае современных ОС). Все остальные обращения программы к памяти в этой странице уже происходят форсированно, с использованием сохраненного значения в этом буфере. Эти блоки есть во всех современных процессорах, собственно из-за этого все они и подвержены уязвимостям называемым теперь Spectre и Meltdown.

Спекулятивное выполнение команд не единственная возможность эксплуатации этих уязвимостей, можно задействовать механизм форвардной загрузки данных из ОП.

Блок TLB не имеет программного (микропрограммного) управления, как обычный кеш. Из него невозможно удалить запись, либо принудительно внести/модифицировать какую-либо запись.

В этом и заключается аппаратная уязвимость, все произведенные трансляции адресов сохраняются в этом ассоциативном буфере.

Соответственно произведенная спекулятивно операция тоже оставит в блоке TLB след в виде записи, нужно только эту запись обнаружить и идентифицировать. А это уже «дело техники», технология давно известна и применяется, она называется «Исследование состояния аппаратуры методом временного прозвона».

Как все это эксплуатируется в реальности?

Первым делом нужно в программе создать буфер размером 2 мегабайта, причем в этот буфер нельзя писать/читать до проведения атаки и он должен располагаться на границе 4К блока. Этот размер принципиален, в буфере должны разместиться ровно 256 страниц по 4К (специфика пейджинговой адресации).

Затем выполняются команды типа:

```
Xor eax, eax;                  eax обнуляется регистр для правильной адресации
Lea ebx, Буфер размером 2M;    ebx принимает значение адреса буфера 2M
Mov al, [адрес ядра ОС];       al читается значение их ядра ОС, байт!
Shl eax, 12;                   принятый байт становится адресом 4К страницы
Mov al, [ebx+eax];             чтение из конкретной страницы буфера 2M
```

В программе после команды Mov al, [адрес ядра ОС]; произойдет прерывание и значение регистра al не изменится (останется нулевым).

А вот в буфере TLB из-за форвардного запроса выполнения операции

произойдет запоминание вычисленного соответствия виртуального адреса и физического адреса [ebx+eax]. Сбросить эту конкретную запись, как это делается в обычном Кеше, в блоке TLB невозможно и она останется

Соответственно если мы узнаем адрес страницы размером 4К в буфере размером 2М, запомненный в блоке TLB, то мы узнаем и значение прочитанное из ядра ОС в регистр al.

Это уже дело техники, «прозвоним» блок TLB. Для этого нужно по одному разу прочитать все его страницы по 4К, их ровно 256, выполним 256 операций чтения (одну на каждый 4К блок). При этом будем измерять время выполнения операции чтения. Та страница, которая будет читаться быстрее остальных, и будет иметь номер в буфере размером 2М соответствующий прочитанному значению байта из ядра ОС.

Это произойдет потому, что в буфере TLB для этой страницы уже вычислено соответствие между виртуальным и физическим адресом, а для всех остальных страниц этой операции выполнено еще не было.

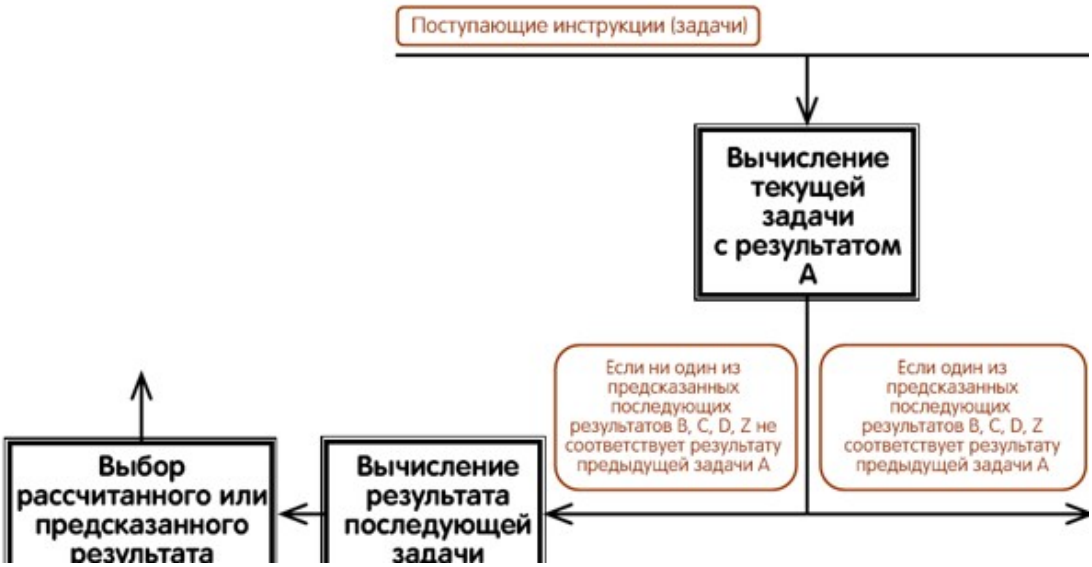
В «прозвоне» конечно не все так просто, там много нюансов, но это вполне возможно.

Spectre

Уязвимость базируется на недостатках технологии спекулятивного исполнения команд, которая значительно увеличивает производительность. Ее суть заключается в использовании предсказателя ветвлений. Процессор заранее рассчитывает некоторое множество наиболее вероятных результатов, каждый из которых может быть достоверным по результату предыдущего вычисления. Например, когда ЦП произведет вычисление результата A предыдущей задачи, уже будут рассчитаны наиболее вероятные последующие результаты B, C, D и Z. Если окажется, что результату A соответствует один из уже заранее рассчитанных результатов (например, D), то он «утверждается» и используется в дальнейшем. Таким образом, вычисление результатов A и D производится в одном периоде процессорного времени, что приводит к увеличению быстродействия.

Если выйдет так, что ни один из множества последующих результатов (B, C, D и Z) не является достоверным, то ЦП произведет его вычисление в дополнительный период времени.

Предсказатель ветвлений способен к обучению. Он запоминает закономерности в вычислении правильных результатов, а спустя время начинает считать их достоверными и предсказывает такие результаты в первую очередь. Именно это свойство и использует вредоносная **Spectre**



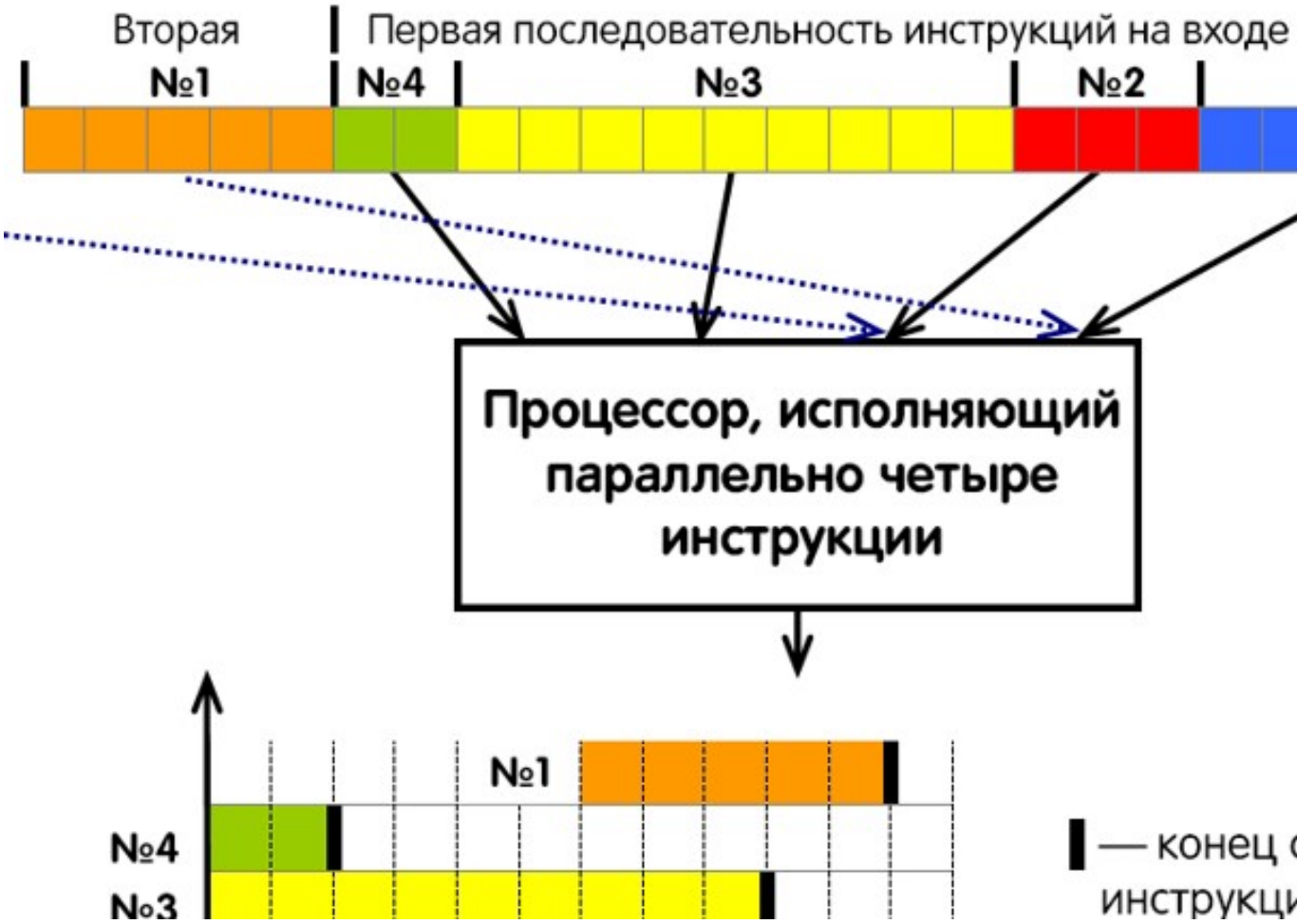
Блок-схема предсказателя ветвлений

Она встраивает свой программный код в общий поток данных и «обучает» предсказатель гарантированно прогнозировать «правильные» результаты путем многократного повторения своих команд. Когда предсказатель начнет считать результаты достоверными — то есть, не подлежащими проверке — Spectre задает в исполняемом коде другую команду. Например, на запись данных в кэш процессора. Предсказатель выполняет ее без проверки на легитимность. Спустя какое-то время процессор, конечно, поймет, что команда являлась недопустимой, и сбросит ее. Но данные атакованных приложений уже будут считаны из памяти.

Meltdown

Этой уязвимости подвержены процессоры Intel (но не AMD), а также некоторые чипы с архитектурой ARM. Она основана на недостатках технологии внеочередного исполнения машинных инструкций. Задачи, поступающие на процессор, исполняются не последовательно (одна за другой), а параллельно, то есть одновременно.

Иногда в цепочке последующая инструкция может оказаться менее сложной, чем предыдущая. Поскольку задачи исполняются одновременно, последующая инструкция будет выполнена раньше — даже если по ее результатам последующая не должна была исполняться вовсе.



Конечно, процессор поймет, что исполненная последующая инструкция недопустима, и аннулирует ее. Но до этого момента она имеет полный доступ к кэш-памяти процессора. Именно эту особенность использует уязвимость **Meltdown**. Она генерирует свою инструкцию, которая будет исполнена процессором раньше, чем предыдущая легитимная. И до исполнения запрещающей инструкции эксплойт успевает прочитать необходимые данные из памяти ядра.