Дефекты проектирования Intel Core 2 Duo

Аналитический обзор с точки зрения безопасности

Крис Касперски

Процессоры Intel Core 2 Duo (и не только они одни!) содержат множество ошибок, приводящих к сбоям программного обеспечения, зависаниям операционной системы и даже возможности удаленного захвата управления компьютером! Часть ошибок обходится программным путем, часть – обновлением микрокода ЦП или прошивки BIOS, оставшиеся – неисправимы и требуют смены процессора. Насколько реальны эти угрозы? Попробуем разобраться!

Критикуя Windows (и отчасти Linux) за большое количество программных ошибок, мы «по умолчанию» закладываемся на непорочность аппаратного обеспечения, проектировщики которого ничем не отличаются от разработчиков операционных систем. До тех пор, пока процессоры были простыми (относительно операционных систем), выходили редко и тестировались тщательно – ошибки «кремниевых коней» носили единичный характер и учитывались разработчиками компиляторов. Сейчас они представляют разве что познавательный интерес. Легендарный «Hamarsoft's 86BUGS list», насчитывающий свыше сотни ошибок, недокументированных машинных команд и особенностей их поведения, последний раз обновлялся в 1994 году, после чего отправился на свалку истории, захлебнувшись в потоке дефектов, обнаруженных в первых моделях Pentium-процессоров, причем ни один из этих дефектов (за исключением знаменитой ошибки деления, описанной в одноименной врезке) до широкой общественности так и не дошел, ограничившись кругом производителей материнских плат, прошивок BIOS, разработчиков операционных систем/компиляторов и прочей технической элитой.

Возьмем, к примеру, инструкцию битового сканирования «BSF dst, src», копирующую в dst индекс первого ус-

тановленного бита в src. Как вам нравится тот факт, что если src равен нулю, то содержимое dst становится неопределенным, то есть там может оказаться любой мусор, что серьезно осложняет отладку программ. Допустим, на машине разработчика установлен ЦП, оставляющий dst неизменным, и разработчик (или его компилятор) закладывается именно на такое поведение ЦП. А вот у конечного пользователя dst может сбрасываться в нуль, нарушая работоспособность программы и заставляя разработчика терять в догадках, с какого момента программа пошла разнос. Обычно в таких случаях все списывается на Windows или «у вас на компьютере вирусы, переустановите операционную систему… ах, вы уже ее переустановили?! ну тогда мы не знаем, разбирайтесь со своей машиной сами».

Кстати, это уже давно не ошибка, а вполне документированная особенность. Intel даже приводит псевдокод команды BSF во втором томе «Instruction Set Reference»:

Листинг 1. Псевдокод команды BSF с «узаконенной» ошибкой

```
IF SRC = 0
      THEN
                ZF := 1;
                DEST is undefined;
      ELSE
                ZF := 0;
                temp := 0;
      WHILE Bit(SRC, temp) = 0
      DO
                temp := temp + 1;
                DEST := temp;
      OD;
FI;
```

С такими «особенностями» можно еще и смириться, но куда прикажете девать эпизодически возникающие исключения на 486+, возникающие при загрузке регистра CR3 (указатель на каталог страниц) в регистр общего назначения? Конечно, непредвиденные исключения можно и подавить, что делает Linux и OpenBSD. Делать-то она это делает, но… местами. А местами не делает. Обращения к регистру CR3 происходят из многих функций ядра, а ядро пишет целая армия разработчиков, часть из которых осведомлена об этом дефекте, а часть – даже не подозревает. В результате мы имеем нестабильно работающую систему.

Windows не пытается сражаться с этим. А зря. Запрещение кэширования на запись в некоторых моделях процессоров разрушает содержимое кэша, откуда процессор продолжает брать ранее скэшированные данные (уже разрушенные), и чтобы программа не «грохнулась», кэш необходимо очистить программным образом, последовательно считывая ячейки памяти – неважно какие – лишь бы заполнить его. Поскольку считываемые данные не были модифицированы, то при последующем поступлении данных в кэш ранее прочитанные ячейки памяти не будут вытесняться ни в кэш вышестоящего уровня, ни в оперативную память, замещаясь новыми данными. На прикладном уровне такая ситуация, конечно, маловероятна, да и нельзя на прикладном уровне управлять кэшированием, но вот драйверы совсем другое дело! Если некоторый регион памяти используется для обмена данными с внешним устройством (видеоконтроллером или платой телеметрии), владельцы «неправильных» процессоров окажутся очень «рады» всевозможным артефактам на экране монитора и неверным телеметрическим результатам. Стоит ли удивляться, что x86 не используются в критических инфраструктурах, где работают простые и тщательно протестированные микроконтроллеры?

Впрочем, мы отвлеклись, хотя это весьма полезное отвлечение, позволяющие читателю понять, что далеко не все странности поведения программного обеспечения имеют программную природу и дефекты процессоров в спонтанно вспыхивающих «голубых экранах смерти» играют далеко не последнюю роль.

Производители ЦП ведут с дефектами проектирования ожесточенную борьбу, прогоняя каждую команду (а то и комбинации команд) через серию агрессивных тестов, результаты которых так или иначе отражаются в документации или же «specification updates». В частности, последние обновления спецификации на Intel Core 2 Duo в любой момент можно скачать с официального сайта Intel: http://www.intel.com/design/core2duo/documentation.htm#specupdt, раздел erraata которого на момент написания этих строк (май 2008) насчитывает 126 ошибок, многие из которых критические и затрагивают не только ядерный, но и прикладной уровень.

Немногим лучше обстоит ситуация с серверными процессорами: Intel Xeon Quad-Core 5400 и его младший собрат Xeon Dual-Core 5100 насчитывают по 54 официально признанных дефекта каждый. И даже Itanium 9000, рекомендованный фирмой Intel для критических инфраструктур (massive, mission-critical computing), хранит в своих недрах 85 «жуков», способных обрушить сервер в любой момент. А ведь это одно из самых дорогих и тщательно протестированных произведений Intel, ориентированное на корпоративный сегмент рынка, который ошибок не прощает и на ко-

Рисунок 1. Intel Core 2 Duo со всеми ошибками, которые в нем только есть

Ошибка деления в Pentium

Самая громкая ошибка в Pentium была обнаружена в 1995 году и продемонстрирована на следующем примере: x – (x/y)*y, результат которого (если только y != 0), должен быть равен нулю, однако при определенных значениях x и y (x = 4195835, y = 3145727) процессор выдавал… 256! Потрясающая точность, однако!

Журналисты подхватили сенсацию, вынудив Intel пойти на замену процессоров, чего она изначально делать не хотела, доказывая, что людям, далеким от математики, точные вычисления не нужны, а вероятность проявления ошибки на произвольном (а не умышленно подготовленном) наборе данных близка к нулю.

С тех пор сообщений об ошибках в ЦП как будто бы не отмечалось. И потому заявление Teo де Раадта, что Core 2 Duo содержит огромное количество ошибок, многие из которых допускают удаленный захват управления, стало очередной сенсацией года.

Текст сообщения Teo де Раадта, опубликованный в конце июня 2007 года

В настоящий момент разработчики программного обеспечения и производители железа заняты разработкой «костылей», исправляющих серьезные ошибки в процессорах серии Intel Core 2, содержащих адское количество багов, и некоторых из этих багов не просто мелкие дефекты проектирования, а реальные дыры, которые несомненно могут быть использованы для атак с прикладного уровня. Ряд ошибок невозможно ни исправить, ни найти обходное решение для предотвращения их возникновения. Intel ограничивается тем, что предоставляет техническую информацию производителям BIOS и ведущим разработчикам коммерческих операционных систем. Open Source-сообщество брошено на произвол судьбы.

Вот полная (текущая) errata: http://download.intel.com/design/processor/specupdt/31327914.pdf (ссылка не работает, т.к. номер спецификации постоянно обновляется, и последняя версия может быть найдена по базовой ссылке: http://www.intel.com/design/core2duo/documentation.htm?iid=prod_core2duo+tab_techdocs#specupdt. – **Прим. переводчика**):

- ☑ Мы готовы биться об заклад, что ошибок на самом деле гораздо больше, чем анонсировано – с каждым месяцем errata становится все больше и больше (так оно и оказалось в последствии. – **Прим. автора**).
- ☑ Intel существенно преуменьшает значимость обнаруженных ошибок – практически все операционные системы вляпаются в эти баги.
- ☑ В сущности MMU (Memory Management Unit – блок управления памятью) подвергся существенным конструктивным изменениям и работает совсем не так, как в предыдущем поколении x86-процессоров. Мало того, что он превратился в сплошное скопище «тараканов», Intel сделала решительный шаг вперед, определив, по ее выражению, «новые методы обработки страничных таблиц» (см. стр. 58 – тут не совсем понятно, что именно смотреть и где, поиск по фразе в Google выдает ссылки только на сообщение Teo де Раадта и ни одного документа от самой Intel. – **Прим. автора**).
- ☑ Некоторые ошибки имеют прямое отношение к технике «переполнения буферов» (самая популярная разновидность атак на сегодняшний день. – **Прим. автора**), где защита от записи или атрибут «неисполняемый» нагло игнорируются процессором. Другие ошибки связаны с некогерентностью (т.е. несогласованностью) инструкций сопроцессора или же производят разрушения памяти – вне области, отведенной в пользование прикладного процесса, посредством последовательности «обычных» (т.е. непривилегированных) машинных инструкций.
- ☑ Все это кажется совершенно невероятным, но это факт!

Вот краткий сводный конспект ошибок Core 2 Duo для нетехнических людей: http://www.geek.com/images/geeknews/2006Jan/core_duo_errata__2006_01_21__full.gif (тут Theo de Raadt слегка противоречит сам себе, поскольку приводит конспект совсем другой errata, с другими ошибками и другой нумераций, поэтому перечисленные ниже номера ошибок не имеют к нему никакого отношения. – **Прим. автора**).

Примечание: некоторые ошибки типа AI65, AI79, AI43, AI39, AI90, AI99 пугают нас так, что душа, минуя пятки, спускается прямо в преисподнюю. Эти «штучки» не могут быть исправлены программным путем, т.е. посредством модификации исходного кода операционной системы и/или приложений, и некоторые из них затрагивают все операционные системы, выпущенные до середины 2008 года, поскольку MMU любых Intel/AMD и совместимых с ними процессоров всегда управлялся одним и тем же способом. И вот теперь Intel говорит нам, что TLB-буферы, входящие в состав MMU, должны «сбрасываться» на совершенно иной манер. Но даже сделав это, ошибки процессора, перечисленные в errata, не позволят создать стабильно работающей операционной системы.

Как уже говорилось, в обозначенном списке скрываются 20-30 ошибок, которые не могут быть преодолены программным путем на уровне операционной системы, и эти ошибки создают потенциальную угрозу для атаки на машину. Готов поставить на заклад кучу денег, что по крайней мере 2-3 из них реально способны на это.

Например, AI90 подходит для атаки на некоторые операционные системы (двоичные сборки OpenBSD в конфигурации по умолчанию к ним не относятся). В настоящий момент я бы не рекомендовал приобретать машины, построенные на базе Intel Core 2, до тех пор пока дефекты проектирования не будут исправлены (что, по моим подсчетам, займет больше года). Intel должна стать более «прозрачной» (а не зажимать технические детали, рассылая их только разработчикам BIOS и коммерческих операционных систем. – **Прим. автора**). Между тем мне хотелось бы отметить, что AMD с каждым днем становится все менее и менее полезной для Open Source-сообщества, поскольку количество ошибок, обнаруженных в ее процессорах, растет не менее стремительно.

тором помимо Intel имеются и другие игроки, впрочем, сталкивающиеся с теми же самыми проблемами.

Мир не совершенен, никто из нас не без греха. Обновлять микрокод ЦП, прошивку BIOS (а в критических случаях – и сами процессоры!) нужно так же регулярно, как накладывать заплатки на операционные системы и прочее программное обеспечение. Ах да, операционные системы… С них-то все и началось!

…и грянул гром

Ознакомившись с очередной errata на Core 2 Duo, Teo де Раадт (Theo de Raadt), ведущий разработчик операционной системы OpenBSD, славящейся своей надежностью и защищенностью, пришел в ярость, и набросился на производителей Core 2 Duo с обличительными заявлениями в стиле: «Как дальше жить и что нам делать?!», тут же подхваченными прессой и ставшими достоянием широкой общественности, постепенно начинающей осознавать, что помощи нет и не будет. Ситуация очень серьезна – достаточно большое число ошибок не только приводит к краху системы, но и (теоретически) допускает возможность удаленного захвата управления, поскольку некоторые инструкции при определенных обстоятельствах выполняются не так, как ожидалось, например, программа, написанная на JavaScript, интенсивно работающая с кэшем, может вызывать направленный удар по памяти, искажая ад-

рес возврата или другие указатели на функции.

Ряд ошибок процессора «сотрудничает» с ошибками программного переполнения. Широко распространена ошибка неявного приведения типов (кастинга) – signed int в unsigned int. Огромное количество программ, написанных на Си, хранят длину копируемого блока памяти в переменной типа signed int, передавая ее функции типа memcpy() и выполняя при этом проверку «if (len > MAX) return -1», забывая о том, что если len < 0, то данная проверка проходит на «ура», а вот в процессе передачи аргумента len функции memcpy() происходит неявное преобразование типов в unsigned int и, поскольку на x86-процессорах знаковый бит является старшим битом, то memcpy пытается скопировать по меньшей мере 2 Гб памяти (при 32-разрядном int). Вот именно, что «пытается». В ходе копирования функция непременно «врезается» в регион, недоступный на запись, и генерируется исключение, приводящее к аварийному завершению программы (хотя в Windows-системах есть надежда, что в процессе копирования удастся перезаписать адрес обработчика исключений до наступления исключения, перехватывая управления при его генерации, но техника SafeSEH предотвращает такой вариант развития событий). Однако в процессоре Core 2 Duo имеется целый класс ошибок, приводящих к преждевременному прерыванию копирования блока памяти, когда его размер превышает размеры адресного пространства. В нормальных программах такие ошибки никак не проявляются, поскольку никто из программистов не копирует блоки памяти по 4 Гб или более того, но вот хакеры… им преждевременное прерывание копирования очень на руку – отличное средство для «дозированного» переполнения буфера в условиях неявного преобразования signed int в unsigned int.

Разработчики OpenBSD попытались заткнуть самые крупные дыры, переписав код ядра так, чтобы исключить или хотя бы снизить вероятность событий, ведущих к проявлению ошибок, но вот остальные программистские коллективы, выражаясь образным языком, даже не почесались, и в первую очередь это относится к новомодному Server 2008, для которого заплаток нет и не предвидится. Что же тогда говорить о «морально устаревшем», но все еще работающем парке Windows 2000, XP, Server 2003?!

Но вернемся к исходному сообщению Тео де Раадта, опубликованному в конце июня 2007 года, когда обнаруженных ошибок было вдвое меньше, чем сейчас: http://marc.info/?l=openbsd-misc&m=118296441702631.

Сенсация или реальная угроза?

Интересно разобраться, что же так напугало Тео де Раадта и насколько велика вероятность атаки на Core 2 Duo, тем более, что за время, прошедшее с момента публикации, количество обнаруженных ошибок удвоилось. Анализировать ошибки мы будем в порядке, обозначенном де Раадтом, с учетом специфики операционных систем семейства NT (Windows 2000, XP, Server 2003/2008, Vista), Linux и линейки BSD – Free, Net и Open, приводя официальную информацию из errata со всеми выкладками и рассуждениями, а местами – сценариями реализации атаки.

AI65: Температурное прерывание не генерируется при выходе текущей температуры за пределы

- **Проблема:** когда DTS (Digital Thermal Sensor – цифровой температурный сенсор) достигает одного из установленных пороговых значений, процессор генерирует прерывание, протоколируя данное событие в журнале (IA32_THERM_STATUS MSR (019Ch) биты [9,7]). Вследствие конструктивного дефекта при достижении пороговой температуры (что индицируется MSR-регистром IA32_THERM_STATUS бит [31]) DTS не генерирует прерывания и не устанавливает биты журнала, даже если было пройдено одно из пороговых значений.
- **Последствия:** при выходе температуры кристалла за пороговые границы процессор не генерирует прерывания.
- **Решение:** не найдено.

В общем, не такой уж и страшный дефект, хотя… учитывая, что:

- 99% машинного времени процессор «спит», практически не нагреваясь;
- при активной работе одного или нескольких блоков процессора тепловыделение резко возрастет;
- для отвода тепла с крохотной площади приходится применять высокооборотные вентиляторы, характеризующиеся высоким уровнем шума, с которым очень сложно (и дорого!) бороться. Вот производители и перешли на адаптивную схему охлаждения, автоматически повышающую обороты вентилятора при нагреве кристалла и практически останавливающую лопасти во время процессорного сна.

Некоторые производители используют внешний термодатчик, но большая часть полагается на показания процессора – так и дешевле, и точнее, но если DTS не работает, то возникает прямая угроза перегрева кристалла, особенно если хакер загрузит его на полную мощность, что очень легко сделать Java-скрипитом или Flash-роликом. Кратковременный перегрев для ЦП в общем-то не опасен, но вот систематический «перекал» ведет к необратимой деградации кристалла. Первым, как правило, гибнет кэш, и система начинает выдавать критические ошибки приложений и выбрасывать голубые экраны смерти.

Таким образом, атаковать систему не нужно. Она и сама умрет через какое-то время.

AI79: Инструкции записи с префиксом REP при определенных обстоятельствах могут завесить ЦП

- **Проблема:** во время выполнения серии инструкций записи, предваренных префиксом REP (REP STOSx/MOVSx), содержимое промежуточного буфера может выгружаться в память до того, как туда поступят актуальные данные. Поведение процессора зависит от порядка выполнения инструкций, временных характеристик «спекулятивных» переходов и временных характеристик некэшируемого буфера записи. Этот дефект распространяется на все операции записи с префиксом REP.

Cores Duo/Solo Errata as of January 21, 2006	Classification / Impact
AE1	**x87/FPU/OS/Debugger**
FST (Floating Point Store--taking data out of the FPU and putting it into memory) with numeric and null segment exceptions may cause General Protection Faults to be missed and FP Linear Address mismatch	**Show-Stopper, but only observed by Intel so far**
AE2	**CPU**
Code Segment limit violation (when the CPU checks to see if the instruction pointer for the currently running program is valid) may occur on 4-Gbyte limit check when the code stream wraps around in such a way that one instruction ends at the last byte of a 4-Gbyte limit, and the next instruction begins at byte 0.	**Show-Stopper. Could be exploited by a virus--though unlikely. Only observed by Intel so far**
AE3	**CPU**
POPF and POPFD (pop flags and pop flags double, which restores the internal FLAGS register from the memory stack) instructions that set the Trap Flag (TF) bits in the EFLAGS register (causing the processor to enter Single-Step mode) may cause Unpredictable processor behavior.	**Software Workaround available, but requires knowledge of this errata**
AE4	**CPU**
REP MOVS (Repeat/Move a string from one memory location to another) operation in fast string mode continues in that mode when crossing into a page with a different memory type.	**Show-Stopper, but only observed by Intel so far**
AE5	**CPU/Memory**
Memory aliasing (using common memory "Page Table Entries") with inconsistent A (Accessed--meaning the page has been used) and D (Dirty--meaning the page has been updated) bits may cause processor deadlock (such as saying it's "D"irty without saying it's been "A"ccessed, which will only occur with an errant operating system or a hardware failure, such as bad memory).	**Show-Stopper, but only observed by Intel so far**
AE6	**CPU/OS**
VM (Virtual Mode) bit will be cleared on a Double Fault Handler. Following a task switch to a Double Fault Handler that was initiated while the processor was in virtual-8086 (VM86) mode, the VM bit will be incorrectly cleared in EFLAGS, taking the processor out of VM86 mode.	**Show-Stopper**
AE7	**CPU/OS/Memory**
Page with PAT (Page Attribute Table) set to USWC (Uncacheable, Speculative, Write Combine) while associated MTTR (Memory Type Range Register) is UC (Uncacheable) may consolidate to UC.	**Possible effect on performance. No data loss or corruption.**
AE8	**x87/FPU/Debugger**
FXSAVE after FNINIT without an intervening FP (floating point) instruction may save uninitialized values for FDP (x87 FPU Instruction Operand (Data) Pointer Offset) and FDS (x87 FPU Instruction Operand (Data) Pointer Selector).	**Would only affect an application being debugged, and the programmer should catch it visually--though probably be confused.**
AE9	**CPU/OS**
Under certain conditions, LTR (Load Task Register) instruction may result in a system hang. The conditions are: 1) Invalid data selector of the TR (Task Register) resulting in a #GP or #NP fault; 2) GDT (Global Descriptor Table) is not 8-bytes aligned; 3) Data BP (breakpoint) is set on cache line containing the descriptor data.	**Show-Stopper, but only observed by Intel so far. Also, any OS developer who codes like this deserves this one.**
AE10	**CPU/OS**
Invalid entries in the Page-Directory-Pointer-Table Register (PDPTR) may cause General Protection (#GP) exception if the reserved bits are set to one.	**The OS would catch this, and Intel makes it very clear that reserved bits are to be set to their specified values (typically a ZERO)**
AE11	**CPU/Memory**
REP MOVS operation in fast string mode continues in that mode when crossing into a page with a different memory type	**Possible effect on performance. No data loss or corruption.**
AE12	**x87/FPU**
FP inexact-result exception FLAG (bit 5 of the Status Word) may not be set if the #InexactResult occurs in any FPU instruction with one of these instructions occurring immediately after: FST/P m32/64, FSTP m80, FIST/P m16/32, FISTP m64. *Note: Inexact precision occurs when the FPU (Floating Point Unit) is unable to fully represent the result in a finite manner, such as 1/3. Since there are a finite number of bits in the FPU (one of 33, 65 or 80--depending on the FPU's current precision mode), not all numbers can be represented exactly. This exception is normally masked, which means it doesn't signal the operating system that an inexact precision has occurred. This is masked because most every instruction cannot be represented exactly, and a small, acceptable amount of rounding is usually tolerable for the high speed of the FPU.*	**Potentially catastrophic! If exact precision is required, the computation may fail because the handler has no idea why it occurred. All numeric exceptions are routed to #MF, which is Interrupt 16. Intel's workaround of inserting 2 NOPs (No Operation) between the offending instructions would *notably* degrade FPU performance.**

Рисунок 2. Сводный конспект основных ошибок Core 2 Duo, на который ссылается Theo de Raadt

- **Последствия:** проявления дефекта варьируются от зависания процессора или операционной системы.
- **Решение:** дефект можно обойти на уровне BIOS.

Да… тут есть о чем задуматься. Инструкции REP STOSx/MOVSx относятся к числу самых популярных. Функции инициализации и копирования памяти в большинстве случаев реализованы именно так, и работают они как на прикладном, так и на ядерном уровне. Трудно представить, что произойдет, если вместо записываемых данных в память будет выгружен мусор, случайно оказавшийся в буфере или… умышленно засунутый туда хакером («засунутый» в смысле оставшийся от предыдущей серии операций записи).

К сожалению, Intel не раскрывает технических деталей, и нам остается только догадываться, при каких именно условиях возникает преждевременный «выброс» данных в память. То, что дефект устраним на уровне BIOS, – это хорошо, но вовсе не факт, что за это не придется расплачиваться производительностью (есть подозрение, что BIOS просто изменяет диапазоны временных характеристик подходящим образом).

Использовать данный дефект для атаки теоретически возможно, но практически для этого необходимо научиться точно воспроизводить условия, при которых происходит преждевременный выброс данных в память. И тогда – хакер сможет с прикладного уровня проникнуть внутрь ядра любой операционной системы, и операционная система не в силах этому помешать. Вся надежда на BIOS.

AI43: Параллельные записи в «чистые» страницы памяти на многопроцессорных системах ведут к неопределенному поведению

- **Проблема:** когда логический процессор осуществляет запись в «чистую» (т.е. ранее не модифицированную) страницу памяти, а другой логический процессор в это же самое время либо осуществляет запись в эту же страницу, или явным образом устанавливает бит модификации в соответствующем поле страничного каталога, «внутриусобные» войны между логическими процессорами, заключенными в один физический кристалл, при определенных обстоятельствах приводят к неопределенному поведению системы.
- **Последствия:** данный дефект приводит к непредсказуемому поведению системы и возможному зависанию.
- **Решение:** дефект может быть устранен на уровне BIOS.

Ох, столько мудрости в этих словах «неопределенное поведение». Ин-

женеры их очень любят. Инженеры вообще любят неожиданности, подстерегающие их в непредвиденных местах. Но чего тут гадать. Все предельно ясно.

Флаг модификации, являясь разделяемым ресурсом, весьма придирчив к порядку, и все операции с ним должны быть упорядочены теми или иными механизмами синхронизации. Даже не сколько сам этот бит, а его окружение. Модифицировать отдельные биты процессор не обучен и он оперирует машинными словами (длина которых не обязательно равна двум байтам, в данном контексте – это минимальная порция обмена с памятью, которая, на Core 2 Duo по одним данным составляет 16, по другим – 32, а по третьим – 64 бита). Процессор сначала читает машинное слово целиком в свой внутренний буфер, взводит/сбрасывает один или несколько бит, после чего записывает его обратно.

А теперь представим, что процессоров у нас два и они одновременно обращаются к одному и тому же слову. Тогда повторная установка уже установленного бита приведет к его сбросу! Операционная система будет считать, что данная страница не была модифицирована и потому при нехватке памяти не станет выгружать ее в файл подкачки, что приведет к необратимой потери данных!

Как это можно использовать для атаки?! Завесить систему – это ерунда, при желании можно разрушить дисковый кэш, для этого нужно организовать операции записи одних и тех же файлов из двух (или более) разных потоков, параллельно с этим «съедая» всю свободную оперативную память. На NTFS самым главным файлом является $MFT, хранящий информацию обо всех остальных файлах тома. Прямое обращение к нему операционная система блокирует, но вот косвенное – создание/удаление/изменение атрибутов одних и тех же файлов из разных потоков – допускает даже с гостевыми правами. А крах дискового тома – это пострашнее зависания.

Кстати говоря, автору уже не первый раз приходилось восстанавливать диски, разрушенные именно таким образом. Причем очень необычным образом. Обычно, если внезапно отключается питание или зависает система, то содержимое кэш-буфера теряется целиком, в результате чего том остается в более или менее работоспособном состоянии, но вот когда теряется лишь часть изменений (равная, как ни странно 4 Кб – т.е. размеру одной страницы), то это дает все основания для заключения, что конструктивный дефект в процессорах проявляется намного чаще, чем этого следовало ожидать.

Cores Duo/Solo Errata as of January 21, 2006	Classification / Impact
AE13	**CPU/Memory**
IFU/BSU deadlock may cause system hang.	**Show-Stopper, but only observed by Intel so far**
AE14	**CPU/Debugger**
MOV with debug register causes debug exception.	**Would only affect an application being debugged. A software workaround is available, but the programmer would need to know of this errata.**
AE15	**CPU/Memory**
INIT does not clear global entries in the TLB.	**Serious, but BIOS writers would know of this errata and code for it.**
AE16	**CPU/Memory**
Use of memory aliasing with inconsistent memory type may cause system hang.	**Show-Stopper, but only observed by Intel so far. Also, any OS developer who codes like this deserves this one.**
AE17	**CPU/Memory**
Machine check exception may occur when interleaving code between different memory types	**Show-Stopper, but only observed by Intel so far**
AE18	**CPU**
Processor Digital Thermal Sensor (DTS) Readout stops updating upon returning from C3/C4 state. **This is the only errata Intel currently has planned to fix.**	**Potentially catastrophic! If returning from C3/C4 state and this fails, and then the HSF fails, the machine could observe catastrophic failure--though unlikely.**
AE19	**CPU/Core Duo only**
Data Prefetch Event Monitor (EMON) Events can only be enabled on a single core.	**Only affects Core Duo, and the EMON Events are still registered, just by one core at a time.**
AE20	**CPU/SMP Environment**
LOCK# asserted during a special cycle shutdown transaction may unexpectedly deassert.	**Potentially catastrophic! In a cooperative, shared CPU resource environment, if one CPU needed to be rebooted, it could potentially leave the other CPUs in an indeterminate state, thereby causing a full server failure--though unlikely.**
AE21	**CPU/Core Duo only**
Disable "Execution-Disable" bit (IA32_MISC_ENABLES[34]) is shared between cores.	**Potentially catastrophic! The "Execution-Disable" bit was added to help keep hackers from launching viruses. Since the "Disable" setting is shared across both cores, it could be such that a state exists where hackers could gain access -- though unlikely.**
AE22	**CPU**
Last Branch Records (LBR) updates may be incorrect after a task switch. *Note: When a task switch occurs within a CPU, such as running two or more programs at the same time, information related to the task are stored by the processor. In this case, where it came "From" may be assigned to where it's going "To".*	**Would only affect an application being debugged or intercepted in some way, and would greatly confuse the programmer for a time.**
AE23	**CPU/Memory**
Address reported by Machine-Check Architecture (MCA) on single-bit L2 ECC errors may be incorrect.	**Would only affect an OS trapping such errors. Intel's workaround is to *"not rely"* on the MCi_ADDR address.**
AE24	**CPU**
Disabling of single-step On Branch Operation may be delayed following a POPFD (Pop FLAGS Double) instruction.	**Only affects applications being debugged. A software workaround exists.**
AE25	**CPU/OS/Chipset**
Performance monitoring counters that count external bus events may report incorrect values after processor power state transitions.	**Should not affect anything significantly. Performance monitoring is a monitor, not an active resource--except for applications monitoring performance.**

Рисунок 3. Сводный конспект основных ошибок Core 2 Duo, на который ссылается Theo de Raadt (продолжение)

Грубым решением проблемы является переход в однопроцессорный режим, что осуществляется добавлением ключа /ONECPU в boot.ini). Конечно, производительность при этом падает, но если целостность данных превыше производительности – это не такая уж и безумная мера. А надеяться на производителей BIOS… где гарантия, что они действительно справятся с ошибкой?!

AI39: Запрос кэш-линейки L1-кэша из одного ядра разрушает модифицированную кэш-линейку другого ядра, приводя к непредсказуемому поведению системы

- **Проблема:** когда запрос данных из Ядра 1 приводит к «промаху» L1-кэша, запрос перенаправляется к L2-кэшу. Если же данная кэш-линейка уже находится в L1-кэше Ядра 2 и была им модифицирована, при определенных обстоятельствах Ядру 1 возвращается неверный результат.
- **Последствия:** непредсказуемое поведение системы.
- **Решение:** BIOS может справиться с этой проблемой.

Как известно, многоядерные процессоры имеют разделяемый (один на всех) L2-кэш и «индивидуальные» L1-кэши, фактически являющиеся частью ядра. В грубом приближении мы имеем обыкновенную многопроцессорную систему с разделяемой внешней памятью и внутрипроцессорной кэш-памятью. Для поддержания памяти в согласованном состоянии используются специальные когерентные протоколы, разработанные десятки лет тому назад. Казалось бы, в чем проблема?!

А в том, что создатели Core 2 Duo или забыли о когерентности (ну это уж вряд ли), либо, что более вероятно, реализовали ее неправильно. Насколько часто разные ядра работают с одними и теми же порциями данных? Очень часто! Ведь при переключении контекстов потоки, стартовавшие на одном ядре, рано или поздно оказываются на другом! Последние версии операционных систем линейки NT, Linux и BSD наконец-то научились отличать ядра от физических процессоров (в многопроцессорных системах) и потому стараются по возможности избегать переброски потоков с одного ядро на другое, поскольку при этом придется заново заполнять L1-кэш. Есть даже специальные API-функции для закрепления потоков за определенным процессором (ядром), но на практике они не используются, и программисты предпочитают доверять системному планировщику, алгоритм которого оптимизируется разработчиками операционной системы с учетом «характера» процессоров, имеющихся на рынке. Но в распоряжении планировщика – сотни потоков и обычно только два (редко четыре) ядра. Так что избежать «передислокаций» потоков невозможно, точнее, возможно, но уж слишком неэффективно.

Как можно использовать данный дефект для атаки? Самое простое – ничего не делать с системой. Она и сама упадет. Разрушение дисковых данных достаточно маловероятно, хотя и не исключено на 100%, а вот критические сбои приложений и голубые экраны смерти – вполне ожидаемое явление. Действительно, если программа что-то записала в память (на самом деле в кэш, но это уже не важно), а при последующем чтении не обнаружила никаких изменений, тут может произойти все что угодно.

Автору удалось реализовать разрушение кучи (динамической памяти) путем циклического выделения/освобождения крошечных блоков в Java-скрипте, следствием чего явился крах браузера. Чисто теоретически возможно разрушить кучу так, чтобы передать управление на shell-код, но для этого необходимо учитывать множество специфичных деталей, неизвестных удаленному атакующему, хотя в принципе достаточно предсказуемых.

Шторм запросов к любому драйверу (например, шторм IP-пакетов) также потенциально способен вызывать крах системы, но, учитывая, что шторма зачастую вызывают BSOD даже на «правильных» процессорах (из-за ошибок синхронизации, допущенных разработчиками драйверов), очень трудно определить, с каким дефектом мы имеем дело: с программным или аппаратным. Тем не менее хакерский потенциал у атак данного типа весьма весомый, что главным образом обуславливается простотой их реализации.

Самое непонятное в этой истории – какое отношение имеет BIOS с «разборкам» внутри процессора?! Единственное разумное предположение – BIOS просто заливает обновленный микрокод, предоставленный Intel. По-другому справиться с проблемой навряд ли получится.

AI90: Атрибут доступа к странице памяти может быть установлен до генерации исключения

- **Проблема:** если лимит сегмента кода установлен близко к концу кодовой страницы, то при определенных обстоятельствах следующей за ней странице памяти процессор может назначить атрибут доступа до генерации исключения общего нарушения защиты или выхода за пределы лимитов кодового сегмента.
- **Последствия:** при проявлении дефекта страница памяти, следующая за последней страницей кодового сегмента, получает атрибут доступа, даже если изначально она была недоступна для чтения.
- **Решение:** пагубное влияние дефекта может быть нейтрализовано путем установки сторожевой страницы (отсутствующей в памяти или неисполняемой), отделяющей кодовый сегмент от последующего за ним семена.

Вот мы и добрались до дефекта, особо отмеченного Тео де Раадтом в качестве подходящего кандидата для атаки на операционные системы (за исключением OpenBSD в конфигурации по умолчанию). Что же это за операционные системы такие?!

В NT лимиты сегментов кода, данных и стека «распахнуты» на все адресное пространство, нижнюю половину которого занимает прикладной код, верхнюю – операционная система. Так что за пределами кодового сегмента вообще ничего нет. Да и чтобы добраться до его конца, необходимо сначала как-то попасть на уровень ядра, где можно делать что угодно и без всяких дефектов процессора (правда, в Core 2 Duo есть еще один интересный дефект, обнаруженный

автором и не описанный в последней errata – при выполнении кусочка инструкции, находящегося в самом конце кодового сегмента, декодер, определяя длину инструкции по первым байтам, пытается считать ее продолжение, которого нет, в результате чего происходит «заворот» в начало сегмента, первой странице которого также присваивается атрибут доступа, после чего генерируется исключение, которое хакеру необходимо отловить, чтобы система не свалилась в BSOD, но что это даст?!

Младшие страницы адресного пространства специально сделаны недоступными в NT для отлова обращения по нулевым указателям, свидетельствующих о том, что программа обращается к невыделенной области памяти, и Windows передает ей исключение, которая та может обработать тем или иным образом, но чаще всего обработчик конструктивно непредусмотрен, и тогда система завершает работу приложения в аварийном режиме.

Установка атрибута доступа на первую страницу приведет к подавлению исключения, и программа упадёт не в момент обращения к нулевому указателю, а чуть позже – когда попытается обработать считанный оттуда мусор. Не слишком-то большое достижение, к тому же легко реализуемое на прикладном уровне без обращения к дефектам процессора).

Правда, 16-разрядные приложения имеют свои собственные 16-разрядные лимиты, и теоретически благодаря ошибке в процессоре атакующий может получить доступ к данным, которые ему читать не положено. В смысле по атрибутам страниц не положено, а так… если атакующий может запускать приложения, то отформатировать диск или удалить все файлы – быстрее и надежнее.

А вот из 32-разрядного приложения вызвать 16-разрядное, ну не то чтобы невозможно, но все-таки достаточно сложно (имеется в виду вызов из shell-кода, исполняющегося в 32-разрядном приложении). Другими словами, на Windows этот дефект не оказывает ровным счетом никакого воздействия.

А что на счет BSD и Linux? Штатные ядра также исповедуют плоскую

Cores Duo/Solo Errata as of January 21, 2006	Classification / Impact
AE26	**CPU/OS/Chipset**
VERW/VERR/LSL/LAR instructions may unexpected update the Last Exception Record (LER) MSR.	**Should not affect anything significantly. A software workaround exists.**
AE27	**CPU/Memory**
I originally had this text: "General Protection (#GP) fault may not be signaled on data segment limit violation above 4 Gigabyte limit. *Note: This cannot occur on a 32-bit CPU, which tells us that Yonah will eventually become 64-bit enabled with EM64T."*	**I originally had this text: "Should not affect anything in 32-bit code."**
Note: This yellow/green text has been replaced with that immediately below	
Since then, Stephane Hockenhull pointed out my error. It is updated as follows: "General Protection (#GP) fault may not be signaled on data segment limit violation above 4 Gigabyte limit when operating in a flat 32-bit model, with a base of 0 and a limit of 4 Gigabytes. This means two or more bytes begin at an address close to 4 Gigabytes, and then would extend beyond the 4GB limit for the last byte."	**In those rare circumstances where this occurs, the system may overwrite data beginning at 0x0, which could prove significant.**
AE28	**CPU/x87/FPU**
Performance Monitoring Events for retired floating point operations (C1h) may not be accurate.	**Should not affect anything significantly. See AE25.**
AE29	**CPU/Debugger**
DR3 address match on MOVD/MOVQ/MOVNTQ memory store instruction may incorrectly increment performance monitor count for saturating instructions executed (Event B1h). *It sounds to me like Intel needs to run their SIMD engine through a few more design meetings.*	**Only affects applications being debugged or trapped. Should not affect any production software.**
AE30	**CPU**
Global Pages in the Data Translation Look-Aside Buffer (DTLB) may not be flushed by RSM instruction before restoring the architectural state from SMRAM.	**Potentially catastrophic! Since the Core Duo is designed to be operated on a notebook, and since notebooks typically enter hibernation modes routinely, any software using global paging in SMM mode, upon returning to normal mode, will be loading data from the wrong memory location!**
AE31	**CPU/Debugger**
Data Breakpoint/Single Step on MOV SS or POP SS (update the Stack Segment/Selector) may be lost after entry into SMM. *Note: It is uncommon for any software to use these instructions outside of task switch or debugging operations. For this reason, debugging is usually disabled for one instruction immediately following MOV SS or POP SS. This is observable in a single-step debugger by executing those instructions. The instruction immediately after will be executed before trapped back to the debugger.*	**Only affects applications being debugged or trapped, and under an extremely rare set of occurrences.**
AE32	**CPU**
CS limit violation on RSM may be served before higher priority Interrupts/Exceptions.	**Should not affect anything significantly; however many interrupt service handlers rely on an order of precedence, such that interrupt A of high priority is known not to be an issue when/if interrupt B of lower priority is triggered. It could become significant in certain circumstances.**
AE33	**CPU/OS/Chipset**
Hardware Prefetch Performance Monitoring Events may be counted inaccurately.	**See AE25.**
AE34	**CPU/Debugger**
Pending x87 FPU Exceptions (#MF) following STI may be served before higher priority interrupts.	**See AE32.**
Note: This chart will be updated from time to time based on Intel's scheduled updates, which are: Jan 18, Feb 15, Mar 15, Apr 19, May 17, Jun 14, Jul 19, Aug 16, Sep 13, Oct 18, Nov 15, Dec 13.	

Рисунок 4. Сводный конспект основных ошибок Core 2 Duo, на который ссылается Theo de Raadt (продолжение)

модель. Кодовый сегмент находится в самом конце адресного пространства, и ничего интересного за ним нет. Некоторые защитные пакеты, разработанные еще в ту далекую эпоху, когда процессоры поддерживали атрибут, «исполняемый» только на уровне селекторов, а необходимость защиты стека, кучи и данных от исполнения заброшенного хакером туда shell-кода уже назрела – произошло вот что: нестандартные ядерные расширения урезали лимиты стека и сегмента данных (расположенных в начале) и отобрали у них атрибут «исполняемый». А за ними расположили сегмент кода. Вот если бы они поступили в обратном порядке (сначала код, потом стек и данные), то у хакеров была бы потенциальная возможность досту-

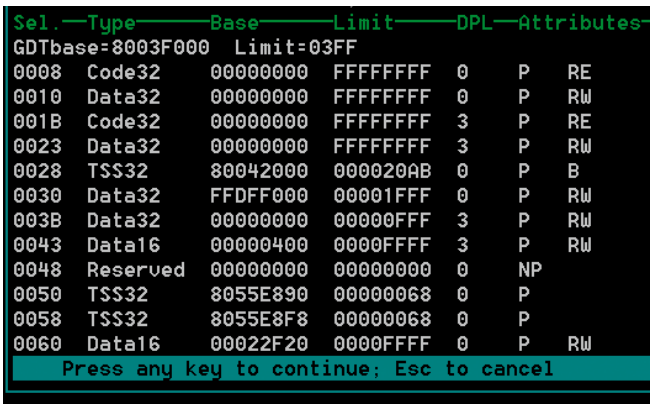

Рисунок 5. Windows использует «плоскую» модель памяти с «распахнутыми» границами сегментов

па к недоступным данным, вот только… стек и куча доступны и так… без всяких дефектов процессора.

Конечно, можно предположить, что где-то есть операционная система, в которой сначала идут пользовательские сегменты кода и данных, а потом расположен сегмент операционной системы, хранящий данные, защищенные от чтения, поскольку, будучи прочитанными, они позволят повысить уровень своих привилегий, например, но… если такая операционная система и есть, то это не Linux, не Windows, не BSD и не QNX… а какая-то студенческая поделка. Вопрос: какой интерес ее атаковать?! Короче, на этот дефект процессора можно вообще не обращать внимания. Ну разве что при разработке новых операционных систем, радикально отличающихся от уже существующих.

AI99: Обновление атрибутов директории кодовых страниц без «инвалидации» TLB может привести к некорректной обработке исключения #PF

- **Проблема:** исключение #PF (Page Fault – страничный отказ) обычно обрабатывается с другими исключениями, перечисленными в порядке увеличения приоритета: #DB (Debug Exception – отладочное исключение), Segment Limit Violation (исключение выхода за пределы Сегмента), #GP (General Protection Fault – общее нарушение защиты). Вследствие дефекта проектирования, #PF исключение обрабатывается некорректно, при наступлении одного из следующих событий:
  - ☑ PDE (Page Directory Entry – запись каталога страниц) модифицируется без «инвалидации» (to invalidate – делать недействительным) соответствующей записи TLB (Translation Look-aside Buffer – буфер обратной трансляции);
  - ☑ транзакция исполнения машинной инструкции находится на стыке двух страниц, причем выполняется одно из следующих условий:
    - ♦ линейный целевой адрес соответствует модифицированной PDE;
    - ♦ PTE (Page Table Entry – запись таблицы страниц) для целевого линейного адреса имеет сброшенный бит доступа;
  - ☑ транзакции предшествует одно из двух следующих исключений:
    - ♦ #DB и #PF;
    - ♦ #GP и #PF.
- **Последствия:** программное обеспечение должно либо отслеживать некорректное #PF исключение, предшествующее #GP исключению, так же как #PF исключению перед #DB.
- **Решение:** не найдено.

Наконец-то мы добрались до дефекта, который Тео де Раадт характеризует как «ну вот, теперь нам Intel говорит, что TLB нужно обрабатывать совершенно иным путем, не похожим на старый». Ох и врет! Intel ничего такого не говорит, признавая за собой дефект и расписываясь в бессилии найти приемлемое обходное решение. Дефект, конечно, серьезный, и остается только удивляться, как существующие операционные системы ухитряются работать на Core 2 Duo, не падать каждые шесть-семь минут (хотя, возможно, они и падают, но не так часто).

Выход только один – ждать новой ревизии процессора, с исправленной ошибкой обработки TLB (после чего все будет «как при бабушке»), либо же модифицировать (причем весьма значительно) ядро операционной системы, чтобы оно стабильно работало и на дефектных ЦП. Microsoft выпуском такой заплатки не озаботилась, хотя похоже, что при ее манерах обращения с TLB обозначенный дефект не особо и мешает. А вот де Раадт доработал ядро OpenBSD, застраховав систему от возможных «сюрпризов» со стороны процессора. Естественно, ядро исправить (да еще таким необычным способом) – это не просто две строчки кода местами поменять, так что его можно понять. Кому хочется отдуваться за чужие ошибки?! А между тем ошибок в процессорах много…

Заключение

Разработчики популярных операционных систем за последние годы существенно усилили их защиту, и золотое время атак на переполняющиеся буферы закончилось. Основные лазейки уже закрыты, и поиск реально «работающих» дыр требует все больших и больших усилий. Но усилило ли это общую безопасность?! Едва ли. Это только раззадорило хакеров, спровоцировав активный поиск принципиально новых методов атак.

Нетронутую целину дефектов железа хакеры только-только начали осваивать. До сих пор не представлено ни одного proof-of-concept exploit и не зафиксировано случаев вторжения, основанных на ошибках ЦП, однако… в атмосфере определенно что-то происходит. Что-то такое.., не предвещающее ничего хорошего.

Впрочем, не будем изображать из себя пророков, предоставив событиям развиваться своим чередом, а сами тем временем приложим максимум усилий по обеспечению собственной безопасности, скачивая свежие прошивки BIOS и выбирая продукцию тех производителей, которым можно верить и которые действительно борются с дефектами процессоров, отмечая свои достижения в сопроводительных файлах. Что же касается операционных систем, то в линейке BSD – OpenBSD несомненный лидер, Linux все на одно лицо, и особой разницы между ними нет (у одних одни недостатки, у других – другие). Microsoft похоже бороться в ошибками процессоров вообще не собирается, а это большой минус.