НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

## МАГИСТЪРСКИ ФАКУЛТЕТ

# ДЕПАРТАМЕНТ "ИНФОРМАТИКА"

**ПРОГРАМА „Софтуерни технологии в Интернет”**



### Курсова работа

**КУРС: INFM152 Компютърна сигурност в Интернет**

**ТЕМА: Слабостите на модерните процесори - Spectre и Meltdown**

**НА СТУДЕНТА** : Георги Иванов Георгиев

**Фак. №:** f88683

**Учебна година:** 2017/2018

**Есенен семестър**

**Дата:** 14/01/2018 г.

**Преподавател:**

/ доц. В.Къдрев / …………………

София

*В началото на месец януари 2018 година на публичността бе съобщена информация за потенциален бъг в архитектурата на съвременните процесори, който може да доведе до изключително много проблеми свързани със сигурността на данните на потребителите. Тези проблеми са известни с имената “Meltdown” и “Spectre”.*

*Настоящата курсова работа има за цел да представи същността на проблемите; начина по който те биха могли да бъдат експлоатирани; начина по който се очаква потребителите да бъдат защитени и конкретния прогрес по отношение на защитата на отделните операционни системи и платформи.*

*Курсовата работа се състои от съдържание, увод, изложение в три глави и заключение.*

**СЪДЪРЖАНИЕ**

1. *Увод*
2. *Проблемите “Meltdown” и “Spectre” – архитектурната грешка на последното десетилетие*
3. *Как да се предпазим*
4. *Заключение*

**Увод**

Силиконовата долина беше разтресена от новината за проблемите „Meltdown“ и “Spectre”. Тя бе направена публично достояние в началото на месец януари и бе репортувана първоначално от Google в края на 2017 г.

Сами по себе си тези проблеми представляват пропуски в архитектурата на процесори от Intel, AMD и ARM, които се възползват от слабост в базовата защита на тези чипове и потенциално биха могли да доведат до кражба на ценна информация от зловредни страни, като например пароли, ключове за криптиране, информация на трети страни(приложения) и др.

Едно от основните неща, които трябва да знаем е, че почти всеки персонален компютър, лаптоп, таблет и смартфон е засегнат от този пропуск в сигурността – независимо коя компания е сглобила устройството или коя операционна система се използва. И това не е всичко. Проблемът не засяга само персонални машини, но и сървъри и машини в data центрове и масивни облачни платформи. При правилно подбрани условия Meltdown и Spectre могат да се използват, за да се краде информация между потребители на облачни услуги въпреки факта, че ресурсите им са изолирани. Важно е обаче да се спомене, че уязвимостта не е лесна за откриване и експлоатиране – изисква се специфичен набор от обстоятелства, включително на машината да има вече инсталиран зловреден код.

Въпреки, че потенциални решения за проблема вече излизат под формата на обновления за различните операционни системи, те често водят със себе си различни проблеми за потребителите – особено за онези с по-стари чипове, където може да се отбележи спад в производителността до 30%.

**Проблемите “Meltdown” и “Spectre” – архитектурната грешка на последното десетилетие**

В своя основен доклад Google откриват, че синхронизацията по отношение на кеш на данни от процесора може успешно да бъде експлоатирана с цел изтичане на информация. Вариантите на проблема са три основни видa: CVE-2017-5753 и CVE-2017-5715, наречени още “Spectre” и CVE-2017-5754 наречен още “Meltdown”. Всички познати варианти на проблема се възползват от т.нар. спекулативно изпълнение – част от механизма за оптимизиране начина на работа на модерните процесори. По време на тяхното изследване Google създават 4 концепции:

1. Първа концепция – демонстрира основните принципи на вариант 1 Spectre в среда на обикновен потребител. Използваният хардуер e съставен от чипове Intel Haswell Xeon CPU, AMD FX CPU, AMD PRO CPU и ARM Cortex A57. Концепцията има за цел да тества способността да се чете частна информация, предоставена в следствие на грешно спекулативно изпълнение, в рамките на един процес, без да се нарушават никакви права за достъп.
2. Втора концепция – демонстрира принципно изпълнение на вариант 1 Spectre в среда на потребител с обикновени права в рамките на модерен Linux kernel. Проблемът позволява неоторизирано четене на виртуалната памет на kernel-а на чипове Intel Haswell Xeon CPU и AMD PRO CPU.
3. Трета концепция – демонстрира принципно изпълнение на вариант 2, където root потребител в рамките на Kernel Virtual Machine (KVM) има възможност да чете kernel паметта на реалната машина.
4. Четвърта концепция – демонстрира принципно изпълнение на вариант 3, където потребителм с нормални права има възможност да чете kernel памет на Intel Haswell XEON чип, при наличието на няколко предварително подготвени обстоятелства, основното от които е kernel паметта да е налична в т.нар. L1D кеш.

Специфика на тестваните процесори:

* Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz ("Intel Haswell Xeon CPU")
* AMD FX(tm)-8320 Eight-Core Processor ("AMD FX CPU"
* AMD PRO A8-9600 R7, 10 COMPUTE CORES 4C+6G ("AMD PRO CPU")
* An ARM Cortex A57 - Google Nexus 5x ("ARM Cortex A57")

***Вариант 1 – Преминаване на проверките за достъп***

В тази секция ще изясня теоретичния подход към проблем номер 1 от тип Spectre. Преди да направим кратко изложение на проблема е важно да споменем какво се има предвид под бранч предикшън (или предвиждане на изпълнението). Основната документация на Intel, свързана с оптимизации на техните чипове, казва следното:

„Предвиждане на изпълнението е техника, която позволява на процесора да започне изпълнението на инструкции по даден клон много преди да се знае пътя, по който този клон ще бъде изпълнен“

С други думи процесорите много често изпълняват всички зададени към тях инструкции независимо от това дали те се равняват на true или false. Това важи също така за информацията съхранявана в т.нар. L1D кеш:

* Зареждането може да се случва спекулативно – преди изпълнението на самите клонове;
* Т.нар. кеш пропуски се запълват по начин, по който информацията може да се пресича;

Освен тези два основни оптимизационни елемента, от които зависи проблема, съществува и още един наречен имплицитно кеширане. То се случва, когато някой елемент от паметта е създаден с цел да бъде кеширан, въпреки че изпълнението на процеса създал паметта може никога да не се нуждае от него. Този процес е следствие от горе споменатото спекулативно изпълнение и начина по който се изгражда възстановяването на кеш пропуските.

Нека вземем за пример следния код:

struct array {

 unsigned long length;

 unsigned char data[];

};

struct array \*arr1 = ...;

unsigned long untrusted\_offset\_from\_caller = ...;

if (untrusted\_offset\_from\_caller < arr1->length) {

 unsigned char value = arr1->data[untrusted\_offset\_from\_caller];

 ...

}

Ако arr1->length е некеширано, процесорът има възможността спекулативно да зареди данните от arr1->data[untrusted\_offeset\_from\_caller]. Това само по себе си е прочит на памет извън привилегиите. В конкретния пример това не е проблем, защото процесорът ще върне състоянието на изпълнението, когато започне изпълнението на самия клон и никоя спекулативно изпълнена инструкция няма да бъде взета предвид.

Нека обаче обърнем внимание на следното разширение на примера:

struct array {

 unsigned long length;

 unsigned char data[];

};

struct array \*arr1 = ...; /\* small array \*/

struct array \*arr2 = ...; /\* array of size 0x400 \*/

/\* >0x400 **(OUT OF BOUNDS!)** \*/

unsigned long untrusted\_offset\_from\_caller = ...;

if (untrusted\_offset\_from\_caller < arr1->length) {

 unsigned char value = arr1->data[untrusted\_offset\_from\_caller];

 unsigned long index2 = ((value&1)\*0x100)+0x200;

 if (index2 < arr2->length) {

   unsigned char value2 = arr2->data[index2];

 }

}

Тук се забелязва следния проблем. Ако arr1->length, arr2->data[0x200] и arr2->data[0x300] не са кеширани, но всичката друга необходима памет е, и предикатите на клона се изпълняват процесорът може да направи следното спекулативно изпълнение преди да зареди arr1->length:

* Зарежда value = arr1->data[untrusted\_offset\_from\_caller]
* Започва зареждането на зависим offset в arr2->data, като зарежда отговарящите редове от кеш паметта в L1 кеш-а;

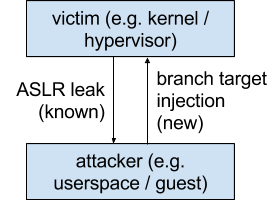
След като изпълнението на кода се върне към неспекулативната си част процесорът забелязва, че untrusted\_offset\_from\_caller е по-голямо от arr1->length, но съхранения в L1 кеш arr2->data[index2] не се зачиства. Чрез измерване времето необходимо за зареждане на arr2->data[0x200] и arr2->data[0x300], злонамерен атакуващ може да определи дали стойността на index2 по време на спекулативното изпълнение е била 0x200 или 0x300 – което предопределя дали arr1->data[untrusted\_offset\_from\_caller]&1 е 0 или 1.

За да се възползва наистина от това поведение, атакуващият трябва да може да предизвика изпълнението на такъв тип зловреден код в определен контекст, използвайки out-of-bounds индекс. За това е необходимо кодът да е наличен в рамките на съществуващ процес или да се изплозват т.нар. интерпретатори или JIT engine с цел да се създаде уязвимостта. За сега не съществуват данни за програми или интерпретатори, които да се възползват от тази зловредна възможност.

***Вариант 2 – Подмяна на целта на клона на изпълнение***

Следва кратко изложение на теорията зад концепцията на Spectre вариант 2. Този вариант на проблема разглежда използването на root привилегии във виртуална машина създадена върху Intel Haswell Xeon CPU.

В своята основа хипотезата приема, че е възможно код от различен защитен контекст да влияе на друг контекст и неговото предварително изпълнение. Процесът се изпълнява като изпълнението на код се пренасочва в атакувания контекст (с други думи се създава вмешателство в изпълнението на кода към даден контекст – атакуван, от външна страна – атакуващ).

[](https://3.bp.blogspot.com/-O6-JHlkvq5U/Wk064bpm2qI/AAAAAAAACPk/rjpYr72vyF0jYASgT-w4NBFQXC3hgIZZwCEwYBhgL/s1600/image2.png)

Основната идея зад вмешателството е да се атакува код от изпълнението, който съдържа индиректен клон, чиито контекст работи с адрес зареден от кернел паметта, след което да се изтрие кеш частта съдържаща адреса от основната памет. Когато CPU-то достигне до изпълнението на индиректния клон, то няма да знае истинския адрес на паметта и няма да може да го калкулира докато не завърши възстановяването на кеша – процес който отнема няколко стотин цикъла. Като резултат, атакуващият има прозорец от около 100 цикъла, през които би могъл да се възползва от предоставеното описание на проблем Spectre вариант 1, тъй като ще има достъп до спекулативното изпълнение на код намиращ се в чужд за него котекст.

В своята същност вариант 2 е изключително сложен за прилагане, за разлика от вариант 1. Причините за това са различните алгоритми, чрез които реално се осъществява т.нар. предварително изпълнение. Те са различни и непрекъснато променящи се.

***Вариант 3 – Зареждане на кеша с измамни данни***

Най-общо казано тази атака прави опит да прочете kernel паметта от потребителски софтуер без да отклонява изпълнението на kernel кода. Това става чрез използването на шаблонния код представен в предните две секции, но изпълнен в рамките на потребителски софтуер. Основна идея е, че проверката за достъп до адрес от паметта може да се окаже не достатъчна спирачка за това атакуващия да прочете данни от паметта в регистър, където проверката за достъп би имала значително отражение върху производителността. Вместо това четенето от паметта може да доведе до предоставяне на резултата от четенето предварително, като проверката за достъп започва да се изпълнява асинхронно и като резултат вдига флаг в буфера за пренареждане на командите, който от своя страна вдига изключение ако проверката се провали. Възможно е също така да се изпълнят инструкции преди изпълнението на клон с голяма латенция, като оптимизационна техника с цел предотвратяване на провал при зареждане на страница. Това отваря още по-голям прозорец за спекулация по време на възникналия интервал между четенето от kernel паметта и вдигането на изключение за невалидни потребителски права.

**Как да се предпазим**

Въпреки голямата драма, която медиите създадоха около проблемите Meltdown и Spectre, пред потребителите съществуват няколко възможности, за да предотвратят нежелани атаки към своите машини. Трябва да имате предвид обаче, че долу изброените начини да се предпазите трябва да се направят съвкупно, а не избирателно. Не можете да очаквате, че само един от тях ще ви предпази напълно от бъдещи атаки.

1. Обновете своята операционна система до последна версия – повечето компании разработващи операционни системи вече пуснаха пачове предоставящи защита срещу проблемите Spectre и Meltdown. Техните решения са по-скоро заобикаляне на проблема отколкото пълното му отстраняване. Необходимо е също така да се знае, че обновленията за Windows ОС, конкретно, водят след себе си до няколко нежелани проблема. На първо място е инсталирането на .NET framework 4.7, което би могло да доведе до промяна в начина на работа на вашите приложения и евентуална невъзможност за тяхното пълноценно функциониране. На второ място е намаляване в производителността на Intel чиповете. Колкото по-стар е вашия процесор, толкова по голям е спада в производителността при извършването на определени процеси (главно операции по писане и четене).
2. Свалете обновления от производителите на вашия лаптоп, компютър или друг вид устройство – производителите на засегнатите устройства следва да предоставят на своите клиенти най-новата версия от драйвери за чиповете в техните устройства. За хората, които са сглобили сами своята конфигурация е препоръчително да потърсят новата версия на firmware от сайта на производителя на техния чип.
3. Уверете се, че уеб агентът ви е обновен до последна версия – по последни данни браузърите са основният начин, чрез който е възможно да се извършват атаки по вашата система. Повечето компании разработващи уеб агенти вече пуснаха обновления защитаващи своите потребители. Google Chrome е един от последните, които ще получат такова, въпреки факта, че той е един от най-употребяваните софтуери за уеб браузване. Очакваната дата за новата версия е 23 януари.
4. За потребителите на Епъл продукти - към момента Епъл предоставя защита на своите потребители по отношение на бъга Meltdown. Това става чрез приложението Mac App Store и предоставените от него обновления. Към момента компанията не разполага с данни за устройства засегнати от бъговете от тип Spectre и не е предвидила извънредни обновления за атаки от този характер.

**Заключение**

В заключение можем да обобщим, че проблемите Spectre и Meltdown са едно доста неприятно стечение на обстоятелствата свързано със стремежа на компаниите производители на процесорни чипове да оптимизират начина, по който те оперират, с цел постигане на по-голяма производителност. Като резултат имаме изключително голям брой устройства потенциални мишени за хакери. Големите доставчици на сървърни машини, както и клауд услуги, от своя страна трябва по най-бързият начин да обновят софтуерите на своите устройства, тъй като те са изцяло отговорни за сигурността на данните на своите клиенти. Като за капак предлаганите софтуерни решения към настоящия момент водят след себе си множество негативни последици, основната, от които е спад в производителността с до 30% при определени обстоятелства. Всичко това доведе до голям удар по пазарната стойност на компаниите производители, като най-пострадала се оказа Intel. Това допълнително може да усложни ситуацията поради необходимостта от ресурс за правилното и цялостното отстраняване на проблема в световен мащаб.

Едно е ясно – тези три пропуска в сигурността са следствие на некоректно взето решение по отношение на оптимизацията, предизвикано от стремеж на компаниите да предоставят експоненциално нарастваща производителност без кой знае каква хардуерна оптимизация. По мнението на много от големите потребители на процесорни чипове, най-доброто и единствено реално решение на проблема е хардуерна подмяна на засегнатите устройства – решение, което в никакъв случай не е приложимо и допустимо в контекста на световния пазар.

**Използвана треминология**

***Кеширани данни*** – данни намиращи се в някое от различните нива на кеш на централния процесор

***Некеширани данни*** – данни, които се намират само в главната памет и не са налични в никое от нивата на кеш на централния процесор. Зареждането на некеширани данни обикновено отнема около 100 цикъла CPU време

***Спекулативно (предварително) изпълнение*** – процесорът може да изпълни клон с инструкции без да знае дали те ще бъдат повикани. Ако се окаже, че резултатът от инструкциите е грешен или ненужен, процесорът може да изхвърли данните без да са необходими архитектурни промени и да продължи с изпълнението на процеса.

***Спекулативен прозорец*** – времето през което процесорът изпълнява спекулативно даден клон с ненужни инструкции и все още не е предвидил факта, че те няма да бъдат изпълнени.

**Използвана литература**

1. <http://uk.businessinsider.com/intel-chip-bug-meltdown-and-spectre-explained-2018-1?r=US&IR=T>
2. <https://googleprojectzero.blogspot.ch/2018/01/reading-privileged-memory-with-side.html>
3. [Intel Optimization Reference Manual](https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf)
4. <https://cyber.wtf/2017/07/28/negative-result-reading-kernel-memory-from-user-mode/>
5. <https://meltdownattack.com/meltdown.pdf>
6. <https://spectreattack.com/spectre.pdf>
7. <http://uk.businessinsider.com/meltdown-spectre-3-ways-to-protect-yourself-against-the-chip-flaws-2018-1?r=US&IR=T>