ARM 处理器分支预测漏洞分析测评及新漏洞发现

王春露[1]，田瑞冬[1]，赵旭[2]，吕勇强[3]，汪东升[4]

（1. 北京邮电大学可信分布式计算与服务教育部重点实验室，100876，北京；2. 北京信息科技大学计算机学院，100192，北京；3. 清华大学北京信息科学与技术国家研究中心，100084，北京；4. 清华大学计算机科学与技术系，100084，北京）

摘要：针对 ARM 处理器上的分支预测漏洞研究不全面、不深入等问题，提出了一种分支预测漏洞测评方法。通过对分支预测漏洞的攻击过程进行研究，提炼了分支预测漏洞的 6 步骤攻击模型。根据分支预测漏洞攻击利用的微体系结构和针对的地址空间，将现有的分支预测漏洞分成 9 种类型。基于攻击模型和分类方法构建了 ARM 处理器上的分支预测漏洞测评方法。在主流的 3 种 ARMv8 架构处理器上对 9 种分支预测漏洞进行了测评，测评内容包括是否受漏洞影响、防御方法是否有效和新漏洞的发掘。实验结果表明，部分 ARM 处理器完全不受分支预测漏洞攻击影响，部分处理器只受 5 到 6 种分支预测漏洞影响。在防御方法方面，尚未存在一种防御方法能防御所有的分支预测漏洞，但可以通过不同防御方法的组合来构建完善的防御体系。在测评过程中，发现了一种 ARM 架构独有的预测执行漏洞——顺序预测漏洞，此漏洞能够泄露同一进程空间的任意数据。

关键词：ARM 处理器；分支预测；漏洞分类；漏洞测评；顺序预测漏洞

中图分类号：TP309.1 **文献标志码**：A

DOI：10.7652/xjtuxb202107008 **文章编号**：0253-987X(2021)07-0071-08

OSID 码

Evaluation of Branch Prediction Vulnerability and New Vulnerability Discovery on ARM Processors

WANG Chunlu[1], TIAN Ruidong[1], ZHAO Xu[2], LÜ Yongqiang[3], WANG Dongsheng[4]

(1. Ministry of Education Key Laboratory of Trustworthy Distributed Computing and Service, Beijing University of Posts and Telecommunications, Beijing 100876, China; 2. Computer School, Beijing Information Science and Technology University, Beijing 100192, China; 3. Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China; 4. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: Because of the lack of research about branch prediction vulnerability on ARM processor, a branch prediction vulnerability evaluation method is proposed to evaluate ARM processor security. Studying the attack process of branch prediction vulnerability, a six-step attack model of branch prediction vulnerability is extracted. According to the difference of microarchitecture and address space used by branch prediction vulnerabilities, the existing branch prediction vulnerabilities are divided into nine types. Based on the attack model and classification method, a branch prediction vulnerability evaluation method is designed. Nine kinds of branch prediction vulnerabilities are evaluated on three mainstream ARMv8 architecture processors. The evaluation content includes the influence of vulnerabilities, the effectiveness of defense methods, and the discovery of new vulnerabilities. The results show that some ARM processors are not

收稿日期：2021-01-14。 作者简介：王春露(1969—)，女，副教授，硕士生导师。 基金项目：国家自然科学基金资助项目(62072263)；国家重点研发计划资助项目(A09B01C02-201801D2)。

网络出版时间：2021-04-07 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20210407.1425.002.html

affected by branch prediction vulnerabilities at all, while the others are only affected by 5 or 6 branch prediction vulnerabilities. In terms of defense methods, there is not a single defense method that can prevent all branch prediction vulnerabilities, but a perfect defense system can be constructed via combination of different defense methods. During the evaluation process, a prediction execution vulnerability unique on ARM architecture, the sequential prediction vulnerability, is also found, which can expose arbitrary data in the same process space.

Keywords: ARM processor; branch prediction; vulnerability classification; vulnerability evaluation; sequential prediction vulnerability

传统高性能处理器通过指令流水线技术来提高指令执行并行度,从而提高处理器性能。但是,程序指令之间会存在数据、控制和资源相关性,这些相关性会造成流水线阻塞[1]。为此,现代高性能处理器采用了多种微架构优化技术来降低相关性对指令流水线的影响。然而,最新的研究表明,由于当前的处理器架构始终以性能优先作为导向,未对性能优化机制进行周密的安全性分析,导致当前处理器系统存在严重的安全漏洞,可能造成计算机机密信息的泄露,危害性极大[2-4]。

分支预测技术作为最基本的优化技术,广泛应用于 Intel、AMD 和 ARM 等主流的商业处理器中。在分支预测机制中,存在一些指令被提前执行,但最终被废弃,无法提交。虽然这些指令的执行不会改变体系结构的状态,不影响程序的正确执行,但这些被废弃的指令可能会在微体系结构中留下痕迹,从而产生被攻击的风险,此类漏洞被称为分支预测漏洞[5]。Kocher 等在 2018 年发表的 Spectre 是利用分支预测漏洞攻击的典型代表[2]。自从此后,学术界对 Spectre 及其变种进行了大量研究,先后提出了 SpectreRSB[3-4]、BranchScope[6]、NetSpectre[7] 和 SGXpectre[8-9] 等攻击变种,对其威胁程度和相关防御手段也进行了充分讨论,但目前的研究主要集中在 x86 架构上。与 x86 架构芯片主要服务于桌面和服务器市场不同,ARM 架构芯片主要集中在移动端和嵌入式领域,更追求低功耗,分支预测器的设计更为保守。因此,专门针对 ARM 架构处理器的分支预测漏洞研究具有重要意义。

本文首先总结了目前已发现的分支预测漏洞,在此基础上提出了一种适用于所有分支预测漏洞的攻击模型。其次,根据分支预测漏洞攻击过程中利用的微体系结构和针对的地址空间,将分支预测漏洞分为了 9 类。通过实验探究了 9 种漏洞对 ARM 处理器的影响。实验结果表明,部分低功耗的 ARM 处理器完全不受分支预测漏洞影响。性能较高的 ARM 处理器只受部分漏洞影响。在防御方面,现有的防御手段只能单独防御某一类分支预测漏洞,想要达到良好的防御效果需要多种防御方法的组合。除此之外,本文还发现了一种新的 ARM 预测执行漏洞——顺序预测漏洞。

1 分支预测漏洞

1.1 分支预测漏洞原理

当指令流在执行过程中遇到分支指令时,可能需要等待目的地址被解析之后才能确定下一步需要执行的指令。此时若使流水线停顿,等待目的地址解析完成再恢复流水线,会严重影响性能。为了提升性能,处理器会预测下一步需要执行的指令并提前执行[1]。预测执行的结果不会被提交而是暂时保存,当目的地址解析完成后,若预测正确,就提交执行结果。但是,如果预测不正确,处理器会丢弃预测指令的结果并使流水线恢复到预测执行前的状态。

现代处理器中设置了专门负责分支预测的微体系结构——分支预测器[10],分支预测器通过记录分支指令的历史执行信息,对分支指令的跳转方向和跳转地址进行预测。分支预测器如图 1 所示。现代分支预测器通常由一个二级自适应分支预测组件[11]和一个目标分支地址缓存区[12](BTB)组成。

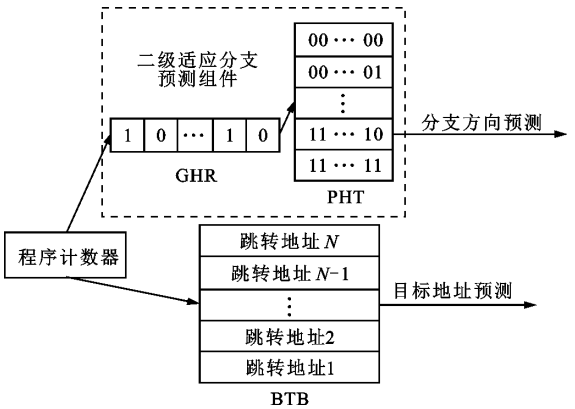


图 1 分支预测器

Fig. 1 Branch predictor

全局历史记录寄存器(GHR)记录了分支指令的历史跳转信息。每个 GHR 的数据可以和历史模式表(PHT)的一个条目相关联，PHT 根据 GHR 的数据做出是否跳转的判断。BTB 记录了分支指令的地址和跳转地址的映射。如果 PHT 预测将跳转，则从 BTB 中根据分支指令的地址查找跳转地址。

现代处理器中通常使用返回堆栈缓冲区(RSB)来预测函数返回地址[4]。RSB 是一种特殊微体系结构，如图 2 所示。在函数调用时(ARM 中是 BL 指令)，将跳转指令的下一条指令的地址作为预测地址储存在 RSB。函数返回时(ARM 中是 RET 指令)，如果出现流水线阻塞，则将栈顶元素作为预测地址放入程序计数器。

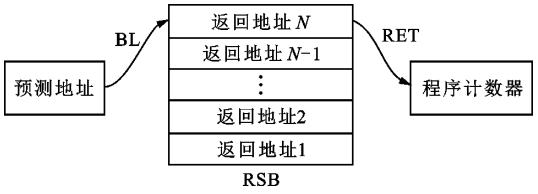


图 2　返回堆栈缓冲区

Fig. 2　Return stack buffer

现代分支预测器的正确率很高，但仍无法保证 100%的正确预测。当分支预测器预测失败时，为了保证最终结果的正确性，处理器会放弃分支预测指令的执行结果，使用者在体系结构层面无法感知到分支预测指令的执行。但是，分支预测指令可能仍会在微体系结构状态中留下痕迹，从而使得攻击者可以利用这些痕迹恢复分支预测指令的执行结果。这种设计上的漏洞导致了被攻击的可能。

1.2　分支预测漏洞攻击模型

分支预测漏洞攻击模型如图 3 所示，总结为 6 个步骤。

步骤 1　毒化分支预测器。根据分支预测器的运行原理可知，分支预测器各部件储存的历史信息会随着程序的执行而变化。攻击者可以设计特殊的程序以修改分支预测器中的内容，从而控制分支预测的预测结果。这一恶意训练的过程被称为毒化。

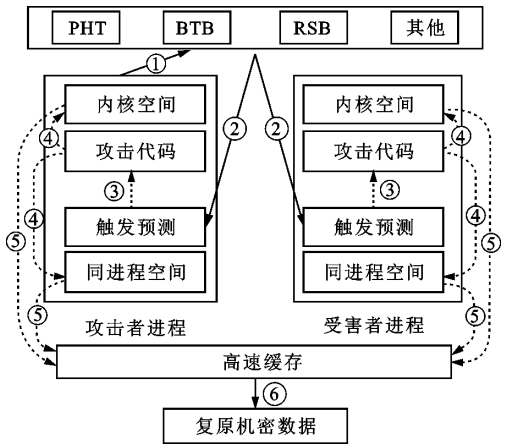
步骤 2　主动触发分支预测。在处理器需要读取数据时，首先会从高速缓存中读取，如果高速缓存中没有处理器需要的数据，处理器再从内存中寻找数据[13]。但是，从内存中读取数据要花费很多时钟周期，处理器不会停顿流水线等待数据读取完成，而是使用预测机制来继续执行。因此，可以通过将程序所需数据从高速缓存中驱逐来主动触发预测执行。

步骤 3　执行攻击代码。触发分支预测后，处理器将分支预测器的预测结果放入程序计数器，但被毒化的分支预测器的预测结果是由攻击者控制，攻击者将预测结果指向特定的攻击代码。

步骤 4　访问机密数据。分支预测的指令执行过程中不会进行权限检查，攻击代码可以直接访问那些受权限保护的地址。

步骤 5　提取机密数据到高速缓存。通过执行攻击代码将机密数据提取到高速缓存。当分支预测失败时，虽然处理器会丢弃分支预测指令执行的结果，但是攻击代码的执行结果仍然在高速缓存中。

步骤 6　复原机密数据。攻击者通常采用缓存侧信道的方式来获取分支预测在高速缓存中留存的信息[14-15]。缓存侧信道利用高速缓存的多级存储结构造成的数据访问延迟来推测高速缓存中的数据。常见的缓存侧信道攻击有 Flush＋Reload[16]、Evict＋Reload[17]和 Prime＋Probe[18]等。



虚线表示此步骤是在预测执行时完成；

实线表示此步骤是在正常的流水线执行时完成。

图 3　分支预测漏洞攻击模型

Fig. 3　Attack model of branch prediction vulnerability

2　ARM 处理器分支预测漏洞测评

2.1　分支预测漏洞分类

目前，曝光的分支预测漏洞攻击主要通过利用 PHT、BTB 和 RSB 这 3 个不同的分支预测组件[2,4,6]实现。

2.1.1　利用 PHT 实现分支预测攻击　图 4 是利用 PHT 的分支预测漏洞攻击代码。用合法的 x(x <array1_size)多次执行 if 语句，会使 PHT 认为程序下一次执行时使用的 x 也是合法的。之后，用一个恶意 x 执行这段代码，并在 if 语句执行时将 array1_size驱逐出缓存，阻塞流水线。尽管 x 大于 array1_size，但分支预测器预测 x 仍符合要求，后面

的代码仍然会被执行。

```
1. if(x<array1_size)
2.    y=array2[array1[x]×4 096];
```

图 4 利用 PHT 的分支预测漏洞攻击代码

Fig. 4 Codes of branch prediction vulnerability attack with PHT

2.1.2 利用 BTB 实现分支预测攻击 图 5 是利用 BTB 的分支预测漏洞攻击代码。执行 func 函数会跳转到 funcA 函数执行。重复执行一定次数后，BTB 会建立 func 函数的虚拟地址 A 到 funcA 函数的虚拟地址 B 的映射。在受害者进程中，gadget 函数和 funcA 函数的虚拟地址相同，两个进程中的 func 函数地址也相同。受害者进程执行 func 函数时本应跳转到 funcB 函数。但是，如果发生了流水线阻塞，分支预测器则会用 func 函数的虚拟地址 A 在 BTB 中索引到虚拟地址 B，并将 B 作为分支预测的结果放入程序计数器，B 地址指向 gadget 函数而不是 funcB 函数。

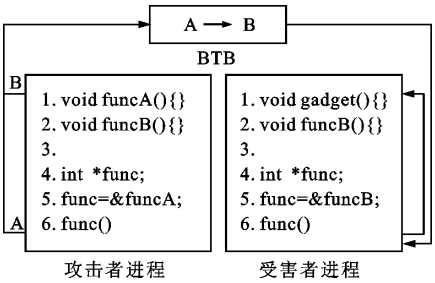


图 5 利用 BTB 的分支预测漏洞攻击代码

Fig. 5 Codes of branch prediction vulnerability attack with BTB

2.1.3 利用 RSB 实现分支预测攻击 图 6 是利用 RSB 的分支预测漏洞攻击代码。攻击者进程中第 6 行代码的虚拟地址和受害者进程中 gadget 函数的虚拟地址相同，都为 A。在攻击者进程中执行 funcA函数时，RSB 会将跳转指令的下一条指令的地址 A 入栈。在执行 funcA 函数时，sched 函数触发进程切换，指令流跳转到受害者进程的 sched 函数开始执行。在受害者进程中，如果在 return 执行时发生了流水线阻塞，RSB 会将之前攻击者进程填充的 A 作为分支预测的结果放入程序计数器，此时虚拟地址 A 指向 gadget 函数。

以上 3 种分支预测漏洞攻击方法都可以针对同进程地址空间和内核地址空间开展攻击，除此之外，还可以针对跨进程地址空间。这是因为不同的进程可以拥有相同的虚拟地址，并且由于分支预测器是

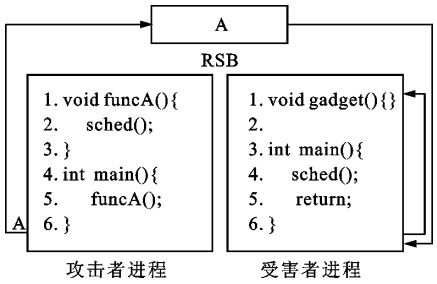


图 6 利用 RSB 的分支预测漏洞攻击代码

Fig. 6 Codes of branch prediction vulnerability attack with RSB

根据指令的虚拟地址来进行预测的，所以当攻击者进程恶意训练分支预测器时，也可以对其他进程的分支预测结果造成影响[10]。根据分支预测漏洞攻击利用的微体系结构和针对的目的地址的不同，将分支预测漏洞攻击分成了 9 类，如图 7 所示。

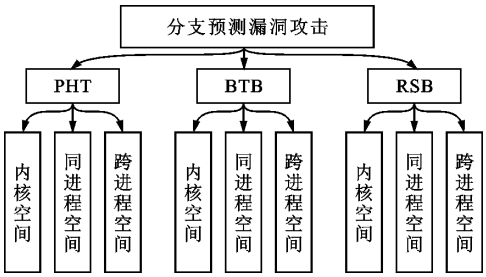


图 7 分支预测漏洞攻击分类

Fig. 7 Classification of branch prediction vulnerability

2.2 ARM 处理器分支预测漏洞测评结果

依据图 7 所示的分类，本文分别为 9 种分支预测漏洞攻击方法编写了可运行在 ARMv8 架构上的攻击代码。选择了 3 种主流的 ARMv8 架构处理器作为实验平台：处理器 A 是一款常用的 ARM 低功耗处理器，其性能也较低；处理器 B 是一款高性能 ARM 处理器；处理器 C 是最新的高性能 ARM 处理器。代码运行环境为 linux-4.15。评判攻击是否成功的标准是攻击者能否获得受害者存储在不同地址空间中的数据。

文献[19]的研究已经表明，9 种攻击方法在主流的 x86 架构都可以攻击成功。但是，本文研究发现，这 9 种攻击方法在 ARM 处理器上有不同的表现。处理器 A 不受任何分支预测漏洞攻击的影响，因为这款处理器属于低功耗处理器，会适当减少复杂寄存器的使用，所以分支预测的功能有限。处理器 B 的测试结果如表 1 所示。可以看出，由于处理器 B 性能较高，其分支预测功能更加完善，更容易受到分支预测漏洞的影响，但仍有 3 种攻击无法在

处理器 B 上起作用，原因可能是 ARM 处理器在进程切换和用户-内核态切换时会对分支预测器的内容做出调整。处理器 C 的测试结果如表 2 所示。可以看出，除了在处理器 B 无法攻击成功的 3 种方法外，利用 BTB 针对跨进程空间的攻击也无法成功，原因是这款处理器已经兼顾了安全性设计，集成了硬件防御措施。

表 1　处理器 B 分支预测漏洞测评结果

Table 1　Evaluation of branch prediction vulnerability on processor B

利用的微体系结构	目标地址	攻击效果
PHT	内核地址	成功
	同进程空间	成功
	跨进程空间	成功
BTB	内核地址	失败
	同进程空间	成功
	跨进程空间	成功
RSB	内核地址	失败
	同进程空间	成功
	跨进程空间	失败

表 2　处理器 C 分支预测漏洞测评结果

Table 2　Evaluation of branch prediction vulnerability on processor C

利用的微体系结构	目标地址	攻击效果
PHT	内核地址	成功
	同进程空间	成功
	跨进程空间	成功
BTB	内核地址	失败
	同进程空间	成功
	跨进程空间	失败
RSB	内核地址	失败
	同进程空间	成功
	跨进程空间	失败

3　防御方法分析和测评

目前，ARM 处理器上的防御手段主要有指令屏障、无效缓冲区和内核页表隔离 KPTI，本小节对这 3 种防御手段进行分析和测评。

3.1　主要防御方法

3.1.1　*指令屏障*　分支预测漏洞最简单的防御方法就是在处理器读取机密信息时禁止分支预测。可以通过指令屏障来实现这一功能。指令屏障强制指令屏障之后的指令等待其之前的指令执行完毕再执行。在 ARMv8 架构中，可以使用 DSB SY＋ISB 的指令组合和 CSDB[20] 来实现指令屏障。用户可以在重要的数据操作前插入指令屏障来防止分支预测指令对这些数据的访问。

3.1.2　*无效微体系结构缓冲区*　除指令屏障外，另一种可行的防御方法是破坏分支预测漏洞攻击中的毒化过程。通过消除攻击者对分支预测相关微体系结构的影响，避免分支预测器引导处理器执行攻击者准备的攻击代码，从而起到防御的作用。这种无效微体系结构的方法通常都插入在进程切换和内核-用户态切换的过程中。用户可以通过 linux cmdline 中的 nospectre_v2 参数使操作系统在进程切换时刷新微体系结构缓冲区。

3.1.3　*内核页表隔离*　内核页表隔离[21] 是一种 Linux 内核功能，将用户空间页表和内核空间页表完全分开，在用户模式下使用的页表包含用户空间的副本和部分的内核空间，用户程序在执行时无法直接访问内核数据，但对于目标为窃取另一个进程用户空间数据的分支预测攻击，KPTI 没有效果。用户可以通过 linux cmdline 中的 kpti 参数来开启/关闭 KPTI。

3.2　测评结果

针对本文中几种成功的攻击方式，在处理器 B 和处理器 C 上进行了防御措施的有效性测评。判断防御措施是否有效的依据是开启防御措施后攻击者能否获得受害者存储在不同地址空间中的数据，若无法得到预期的数据，则判断防御措施生效。本文实施防御措施的方法是：在受害者进程访问机密数据的代码段前插入指令屏障，并通过 nospectre_v2 和 kpti 两个 cmdline 参数开启无效缓冲区功能和 KPTI，结果如表 3 和表 4 所示。

从表 3 和表 4 可以看出，指令屏障可以防御所有的分支预测攻击，但这需要事先对运行程序进行代码分析，实现难度较大。在实际的应用中，往往只在内核代码插入指令屏障。无效缓冲区一般在用户-内核态切换和进程切换时执行，因此只能防御针对内核空间和跨进程空间的攻击。KPTI 只隔离了内核空间，因此只对针对内核空间的攻击有效。

从本小节可知，3 种防御方法都无法独自防御所有的分支预测漏洞攻击。因此，为了实现有效防御，通常结合多种防御措施来搭建有效的防御体系。当 3 种防御方法全部打开时，3 种 ARM 处理器都能够有效防御针对分支预测漏洞的攻击。

表 3 处理器 B 防御措施有效性测评

Table 3 Evaluation of branch prediction vulnerability defense on processor B

利用的微体系结构	目标地址	防御效果		
		使用指令屏障	开启无效缓冲区	开启 KPTI
PHT	内核空间	成功	成功	成功
	同进程空间	成功	失败	失败
	跨进程空间	成功	成功	失败
BTB	同进程空间	成功	失败	失败
	跨进程空间	成功	成功	失败
RSB	同进程空间	成功	失败	失败

表 4 处理器 C 防御措施有效性测评

Table 4 Evaluation of branch prediction vulnerability defense on processor C

利用的微体系结构	目标地址	防御效果		
		使用指令屏障	开启无效缓冲区	开启 KPTI
PHT	内核空间	成功	成功	成功
	同进程空间	成功	失败	失败
	跨进程空间	成功	成功	失败
BTB	同进程空间	成功	失败	失败
RSB	同进程空间	成功	失败	失败

4 新型 ARM 处理器预测执行漏洞

ARM 处理器在执行 RET 指令时，会触发一种 ARM 处理器独有的预测执行——顺序预测。顺序预测同样有可能造成处理器数据泄露。

4.1 顺序预测漏洞原理

在 ARM 体系结构中，当指令流水线执行到 RET 指令时，如果函数返回地址不在高速缓存中，处理器就需要花费较长的时间从内存中提取数据，这会造成处理器流水线阻塞。此时，ARM 处理器的顺序预测就会被触发，处理器会使用 RET 指令紧邻的下一条指令来预测执行。当处理器发现预测错误时，会回滚处理器中各个寄存器的状态，保证计算机结果的正确性，与分支预测相同，顺序预测同样会在处理器高速缓存中留下可观察的痕迹。

顺序预测原理如图 8 所示。函数 F1 在 RET 指令返回时，按照顺序预测的执行路径，会直接执行函数 F2 的指令，而不是跳转到函数 main 继续执行。

顺序预测和 RSB 都依靠 RET 指令来作为触发

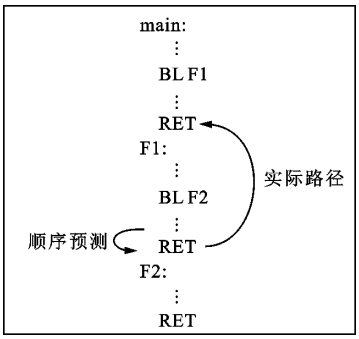


图 8 顺序预测原理

Fig. 8 Principle of sequential prediction vulnerability

指令，不同的是 RSB 依靠微体系结构缓冲区来决定预测执行的下一条指令，但目前还未发现顺序预测需要借助任何寄存器或缓冲区。

本文尚未在 x86 架构中发现顺序预测。这是由于 x86 注重于高性能，而 ARM 更追求低功耗。两者在预测执行上的设计会有不同。实际应用中，顺序预测大部分情况下都是失败的，本文猜测顺序预测是 32 位 ARM 架构遗留下来的机制，但由于 ARM 没有公开这方面的具体信息，很难考证。之后会进行进一步研究。

4.2 顺序预测漏洞攻击方法

依据顺序预测的原理，本文实现了利用顺序预测的攻击方法。由于顺序预测的预测方向是固定的，攻击者无需毒化特定的微体系结构组件，其攻击方法比分支预测漏洞更加简单。总体而言，顺序预测攻击仍然符合本文提出的攻击模型。

本文编写的攻击代码如图 9 所示。顺序预测会执行 RET 指令紧邻的下一条指令，因此顺序预测的预测方向是固定的，只需设计特殊的代码布局，在 RET 指令之后插入本文的攻击代码。当 speculation 函数返回时，可以利用 flush 函数将函数返回地址从高速缓存中驱逐，触发预测执行，处理器会在

```
1. Function speculation(){
2.     flush();
3.     RET;
4. }
5. Function gadget() {
6.     memory_access();
7. }
8. Function main() {
9.     speculation();
10.  side_channel();
11. }
```

图 9 顺序预测漏洞攻击代码示例

Fig. 9 Example of codes for sequential prediction vulnerability attack

接下来的预测执行中执行 gadget 函数。当程序执行完成后，处理器发现预测错误并回滚，但还是在高速缓存中留下了执行 gadget 函数的相关信息，本文就可以通过 Flush+Reload 将高速缓存中的信息取出，成功获取机密信息。

本文在处理器 C 上运行了漏洞利用代码，结果表明利用顺序预测的攻击可以成功泄露同一进程的用户空间的任意数据，但是由于顺序预测不借助任何中间微体系结构，因此顺序预测无法对跨进程空间产生影响。另一方面，由于顺序预测的预测执行窗口不够大，顺序预测攻击无法泄露内核空间的数据。多数情况下 RET 指令仍然是按照 RSB 提供的函数返回地址来预测执行。

4.3　顺序预测漏洞防御方法分析

由于顺序预测不借助任何中间缓冲区，因此常用的无效缓冲区的方法无法防止利用顺序预测的攻击。KPTI 只能防止针对内核地址空间的攻击，但目前本文尚未成功利用顺序预测漏洞攻击内核地址。利用指令屏障可以有效防止这类攻击，可以在每次 RET 指令之后插入指令屏障来防止利用顺序预测漏洞的攻击，并可以将此功能集成到编译器当中，但对性能有较大影响。

5　结　论

本文对分支预测漏洞进行了深入研究，总结了分支预测漏洞的攻击模型，并提出了一种新的分类方法。系统性地研究了目前已经曝光的分支预测漏洞攻击在 ARM 上的危害程度，并评估了目前 ARM 处理器上的防御措施的有效性。在 ARM 处理器上，发现了一种新的预测执行漏洞——顺序预测漏洞。本文得出的结论如下。

(1)现有的分支预测漏洞攻击方法在 ARM 处理器上的表现与 x86 不同，一些攻击方法无法对 ARM 处理器造成影响。

(2)在 ARM 处理器上，没有一种防御方法可以防御所有的分支预测漏洞，需要多种方法的组合才能完全防御分支预测漏洞攻击。

(3)由于架构的不同，ARM 处理器上存在其特有的预测执行漏洞，有必要专门针对 ARM 处理器进行研究。

参考文献：

[1] 张晨曦. 计算机体系结构 [M]. 2 版. 北京：高等教育出版社，2005：152-157.

[2] KOCHER P，HORN J，FOGH A，et al. Spectre attacks：exploiting speculative execution [C]// Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). Piscataway，NJ，USA：IEEE，2019：1-19.

[3] KORUYEH E M，KHASAWNEH K N，SONG C Y，et al. Spectre returns!：speculation attacks using the return stack buffer [C/OL]// Proceedings of the 12th USENIX Workshop on Offensive Technologies. [S. l.]：USENIX，2018.[2021-01-01]. http：//www.cs. ucr. edu/～nael/pubs/woot18. pdf.

[4] MAISURADZE G，ROSSOW C. Ret2spec：speculative execution using return stack buffers [C]// Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York，USA：ACM，2018：2109-2122.

[5] 吴晓慧，贺也平，马恒太，等. 微架构瞬态执行攻击与防御方法 [J]. 软件学报，2020，31(2)：544-563.
WU Xiaohui，HE Yeping，MA Hengtai，et al. Microarchitectural transient execution attacks and defense methods [J]. Journal of Software，2020，31(2)：544-563.

[6] EVTYUSHKIN D，RILEY R，ABU-GHAZALEH N C A E，et al. BranchScope [J]. ACM SIGPLAN Notices，2018，53(2)：693-707.

[7] SCHWARZ M，SCHWARZL M，LIPP M，et al. NetSpectre：read arbitrary memory over network [C]// Proceedings of the 24th European Symposium on Research in Computer Security. Cham，Germany：Springer，2019：279-299.

[8] HUO Tianlin，MENG Xiaoni，WANG Wenhao，et al. Bluethunder：a 2-level directional predictor based side-channel attack against SGX [J]. IACR Transactions on Cryptographic Hardware and Embedded Systems，2020(1)：321-347.

[9] CHEN Guoxing，CHEN Sanchuan，XIAO Yuan，et al. SgxPECTRE：stealing intel secrets from SGX enclaves via speculative execution [C]// Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P). Piscataway，NJ，USA：IEEE，2019：142-157.

[10] EVERS M，CHANG P Y，PATT Y N. Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches [J]. ACM SIGARCH Computer Architecture News，1996，24(2)：3-11.

[11] YEH T Y，PATT Y N. Two-level adaptive training branch prediction [C]// Proceedings of the 24th Annu-

al International Symposium on Microarchitecture. New York, USA: ACM, 1991: 51-61.

[12] ACIIÇMEZ O, KOÇ Ç K, SEIFERT J P. On the power of simple branch prediction analysis [C]// Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security. New York, USA: ACM, 2007: 312-320.

[13] 唐朔飞. 计算机组成原理 [M]. 2 版. 北京: 高等教育出版社, 2008: 109-117.

[14] OSVIK D A, SHAMIR A, TROMER E. Cache attacks and countermeasures: the case of AES [C]// Proceedings of the 2006 Cryptographers Track at the RSA Conference. Cham, Germany: Springer, 2006: 1-20.

[15] KOCHER P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems [C]// Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. Cham, Germany: Springer, 1996: 104-113.

[16] YAROM Y, FALKNER K. FLUSH + RELOAD: a high resolution, low noise, *L*3 cache side-channel attack [C]// Proceedings of the 23rd USENIX Security Symposium. [S. l.]: USENIX, 2014: 719-732.

[17] GRUSS D, SPREITZER R, MANGARD S. Cache template attacks: automating attacks on inclusive last-level caches [C]// Proceedings of the 2015 USENIX Security Symposium. [S. l.]: USENIX, 2015: 897-912.

[18] LIU Fangfei, YAROM Y, GE Qian, et al. Last-level cache side-channel attacks are practical [C]// Proceedings of the 2015 IEEE Symposium on Security and Privacy. Piscataway, NJ, USA: IEEE, 2015: 605-622.

[19] CANELLA C, BULCK J V, SCHWARZ M, et al. A systematic evaluation of transient execution attacks and defenses [C]// Proceedings of the 28th USENIX Security Symposium. [S. l.]: USENIX, 2018: 249-266.

[20] GRISENTHWAITE R. Cache speculation side-channels [EB/OL]. [2020-10-01]. https://developer.arm.com/-/media/Files/pdf/Cache_Speculation_Side-channels.pdf.

[21] GRUSS D, LIPP M, SCHWARZ M, et al. KASLR is dead: long live KASLR [C]// Proceedings of the 9th International Symposium on Engineering Secure Software and Systems. Cham, Germany: Springer, 2017: 161-176.

[本刊相关文献链接]

高榕,张良,梅魁志. 基于 Caffe 的嵌入式多核处理器深度学习框架并行实现. 2018,52(6):36-41+113. [doi:10.7652/xjtuxb201806007]

李杨,王劲林,叶晓舟,曾学文. 面向嵌入式处理器的优化 Montgomery 模乘算法. 2017,51(2):47-52+127. [doi:10.7652/xjtuxb201702008]

王强,董小社,王恩东,朱正东. 基于 I/O 受限进程识别的虚拟处理器调度机制. 2015,49(4):53-60. [doi:10.7652/xjtuxb201504009]

崔继岳,梅魁志,刘冬冬,李博良. 面向 OpenCL 的 Mali GPU 仿真器构建研究. 2015,49(2):20-24+68. [doi:10.7652/xjtuxb201502004]

丑文龙,梅魁志,高增辉,李博良. ARM GPU 的多任务调度设计与实现. 2014,48(12):87-92. [doi:10.7652/xjtuxb201412014]

(编辑 陶晴)