软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(12):4476–4503 [doi: 10.13328/j.cnki.jos.006695]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

ETSG-SMT: 一种 SMT 时间信道安全问题描述模型*

岳晓萌[1,2], 杨秋松[1], 李明树[1]

[1](中国科学院 软件研究所 基础软件国家工程研究中心, 北京 100190)

[2](中国科学院大学, 北京 100049)

通信作者: 岳晓萌, E-mail: yxmsamg@live.com

摘 要: 同时多线程(simultaneous multi-threading, SMT)技术是现代高性能处理器的标配技术, 是提升线程级并行度的重要微架构优化技术之一. SMT 技术在带来性能提升的同时, 也引入了新的时间信道安全问题, 相对于跨核、跨处理器, SMT 技术下的时间信道安全问题更难应对和防护, 且陆续有新的安全问题出现. 当前缺少一种系统描述 SMT 环境下时间信道安全问题的方法. 从利用 SMT 技术产生时间信道的原理入手, 聚焦 SMT 环境下共享资源产生的时间信道及其攻击机理, 基于拓扑排序图(topological sort graph, TSG)模型, 结合数据流分析扩展得到一种适用于 SMT 环境下的时间信道安全问题描述模型——ETSG (extended topological sort graph, 扩展的拓扑排序图)-SMT. 首先介绍 SMT 环境下时间信道安全问题利用和防护的技术特点以及使用 TSG 模型分析 SMT 环境下时间信道安全问题的限制与不足; 然后在 TSG 模型基础上, 针对 SMT 技术特征及其安全问题的形式化描述特点, 结合数据流分析技术形成一套新的建模方法; 最后, 通过将 ETSG-SMT 模型应用到 SMT 环境下现有的攻击方法和防护案例推导中, 证明使用 ETSG-SMT 模型对 SMT 环境下时间信道原理分析和防护技术推导有很好的应用价值.

关键词: 同时多线程; 拓扑排序图; 数据流; 时间信道

中图法分类号: TP311

中文引用格式: 岳晓萌, 杨秋松, 李明树. ETSG-SMT: 一种 SMT 时间信道安全问题描述模型. 软件学报, 2022, 33(12): 4476–4503. http://www.jos.org.cn/1000-9825/6695.htm

英文引用格式: Yue XM, Yang QS, Li MS. ETSG-SMT: Description Model of SMT Timing Channel Security Problem. Ruan Jian Xue Bao/Journal of Software, 2022, 33(12): 4476–4503 (in Chinese). http://www.jos.org.cn/1000-9825/6695.htm

ETSG-SMT: Description Model of SMT Timing Channel Security Problem

YUE Xiao-Meng[1,2], YANG Qiu-Song[1], LI Ming-Shu[1]

[1](National Engineering Research Center for Fundamental Software, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Simultaneous multi-threading (SMT) technology is the standard of modern high-performance processor technology, which is important micro structure optimization technology to improve the thread level parallelism. Compared with cross-cores and cross-processors, the timing channel security in SMT technology is more difficult to deal with and protect, and new security problems have emerged successively. At present, there is no systematic method to describe the timing channel security problem in SMT environment. Starting from the principle of timing channel attack using SMT technology, focusing on the timing channel attack generated by shared resources in SMT environment and its attack mechanism, based on the topological sort graph (TSG) model and data flow analysis extension, a description model of timing channel security problem suitable for SMT environment, ETSG-SMT is proposed. Firstly, this study introduces the technical characteristics of the utilization and protection of timing channel under SMT environment and the limitation and deficiency of TSG model to analyze these security problems. Then, on the basis of TSG model for SMT technical

* 基金项目: "核高基"国家科技重大专项(2014ZX01029101-002); 中国科学院战略性先导科技专项(XDA-Y01-01, XDC05020200)

收稿时间: 2021-03-20; 修改时间: 2021-05-17, 2021-10-08; 采用时间: 2021-11-13

© 中国科学院软件研究所 http://www.jos.org.cn

characteristics and its security problems of formal description characteristics combined with data flow analysis technology, a set of new modeling method is formulated. Finally, by applying ETSG-SMT model to the existing time channel attack methods and protection cases in the SMT environment, it is proved that the ETSG-SMT model has a sound application value in the analysis of the principle of timing channel attack and the derivation of protection technology in the SMT environment.

Key words: SMT; TSG; dataflow; timing channel

从 SMT(simultaneous multi-threading, 同时多线程)技术商用开始, 研究人员就开始利用 SMT 技术进行安全问题的挖掘和研究. 因为 SMT 技术增加了很多处理器微架构安全问题利用的场景和机会, 因此有研究人员评价 SMT 技术是“廉价的硬件并行意味着廉价的安全性”[1].

Aciicmez[2]在 2009 年总结了当时已有的基于 SMT 技术的攻击方法和应用案例, 对 SMT 技术导致的安全问题进行了系统归纳, 也汇总了当时 SMT 环境下安全问题的缓解手段. 然而, SMT 技术带来的安全问题短时间并没有得到解决或明显缓解. 2018 年, Ge 等人[3]总结了 2002 年–2018 年期间各个系统层级由于资源共享导致的时间信道安全问题及缓解措施, 提出基于 SMT 技术产生的硬件线程级时间信道安全问题相对于跨核、跨处理器的时间信道安全问题更难应对和防护, 且当前已有的 SMT 环境下时间信道安全问题的防护或缓解方法大部分集中在缓存结构. Canella 等人[4]以及吴晓慧等人[5]总结和评估处理器中瞬态执行攻击和防护方法, 但没有进一步形成形式化的描述方法.

2017 年, He 等人发表了在侧信道攻击下你的缓存是否安全的论文[6], 该文针对已有的缓存结构受到的侧信道攻击方法以及已有的防护型缓存结构进行了整合和分析, 形成了一套概率信息流图模型来表现攻击的内在原理, 并且通过分析攻击者的攻击成功概率来总结已有防护型缓存架构的安全性. 2020 年, He 等人针对处理器微架构侧信道安全问题新提出一种模型——TSG (topological sort graph, 拓扑排序图)模型[7], 这是一种由顶点和顶点之间的边来形成序列的有向无环图, 能够描述处理器微架构产生时间信道基本操作和步骤, 提出的“安全依赖”概念从数据流和控制流的依赖关系出发, 从理论上证明了缺失安全依赖性和条件竞争等价, 是形成处理器微架构侧信道攻击的主要原因, 但是该模型的描述能力在 SMT 环境下存在一些漏洞和不足.

虽然研究人员对时间信道安全问题的认识已逐步形成成熟体系并逐渐有相关模型描述, 但是目前还没有 SMT 时间信道安全问题的描述模型, 也没有把 TSG 及类似形式化描述模型应用于 SMT 环境下进行时间信道安全问题建模的先例.

本文基于 SMT 环境下时间信道安全问题的底层机理及其防护方法的技术原理, 提出一种统一的模型描述方法——ETSG (extended topological sort graph, 扩展的拓扑排序图)-SMT, 形成具备 SMT 技术特点的攻击原理分析和防护方法设计的模型基础; 本文使用 ETSG-SMT 模型对 SMT 环境下时间信道安全问题进行攻击原理建模和防护方法分类, 通过对不同的 SMT 环境下攻击原理和防护方法的进一步抽象, 形成 SMT 环境下特有的攻击和微架构级防护策略的映射关系, 建立基于 ETSG-SMT 模型的安全问题微架构级防护方法的推导路线. 最后, 本文针对当前缺少微架构级防护方法的 SMT 执行端口共享的时间信道安全问题进行防护推导, 进而证明 ETSG-SMT 模型的应用价值. 主要贡献有如下 3 点.

1) 提出了一种形式化描述 SMT 环境下时间信道安全问题的模型;
2) 提出了 SMT 环境下线程间依赖的概念, 并证明了 SMT 环境下存在线程内部和线程间的数据流依赖关系, 依赖造成的模型内部条件竞争是产生 SMT 时间信道安全问题的原因;
3) 建立基于 ETSG-SMT 模型的 SMT 环境下时间信道防护方法研究路线, 并完成特定案例的防护推导.

1 背 景

SMT 技术可以使一个物理核中并行运行多个线程, 多个线程可以共享处理器微架构内部的大量数据结构和资源, 并且可以在同一时刻利用不同执行资源执行对应线程的指令.

如图 1 所示, SMT 技术区别于其他的多线程技术的核心是, 可以在同一个流水级执行来自几个不同线程的操作. 普通的超标量单核处理器同时只能运行一个线程, 在局部时间内资源还会处于空闲状态(例如取指令

© 中国科学院软件研究所 http://www.jos.org.cn

阶段); 对于普通的多核处理器, 其会使用多个物理核来运行多个线程, 是普通单核处理器的简单叠加; 粗粒度多线程技术使用时间片的方式分配资源使用, 资源会在一段时间内分配给独立的一个线程使用, 同一时间资源只会分配给一个线程使用, 且时间粒度较粗; 细粒度多线程技术也是使用时间片的方式分配资源使用, 区别于粗粒度多线程技术, 其时间粒度更细, 例如极端情况每个时钟周期都会切换资源的使用者; SMT 技术区别于粗/细粒度的多线程技术, 其可以灵活分配处理器内部资源, 大部分资源结构被多个线程共享, 且同一周期可以调度多个线程操作同时执行, 最大化地提高资源利用率.

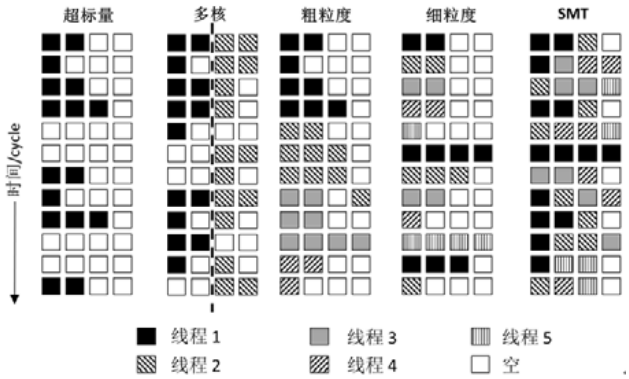

图 1 SMT 技术特征

以 Intel 的超线程技术[8]为例, SMT 在技术原理上主要包含以下 3 个微架构特点.

(1) 制定资源共享策略. SMT 技术可以使用资源复制、资源分割和资源共享的方式来实现数据结构资源的分配. 以图 2 为例, Intel SMT 技术中的 ITLB (instruction translation lookaside buffer, 指令转换缓冲)和 IP (instruction pointer, 指令指针)结构采用了资源复制的处理策略, Uop (micro operation, 微操作)Queue (微操作队列)结构采用了资源分割的处理策略, Trace Cache (追踪缓存)结构采用了资源共享的处理策略;

(2) 设置线程选择点. 以图 2 为例, Intel SMT 技术中两个队列(Queue 和 Uop Queue)都是线程选择点, 其作用是在共享资源的使用上进行两个线程的选择及切换. 合理的线程选择点设置和选择算法使用, 能够有效提高双线程共享资源的利用率以及双线程的公平性;

(3) 保证线程公平性. 以图 2 为例, Intel SMT 技术中在线程选择位置均使用了分割策略, 配合线程选择点的公平算法, 避免单一线程由于长时间得不到资源而饿死的情况出现, 进而保证线程公平性.

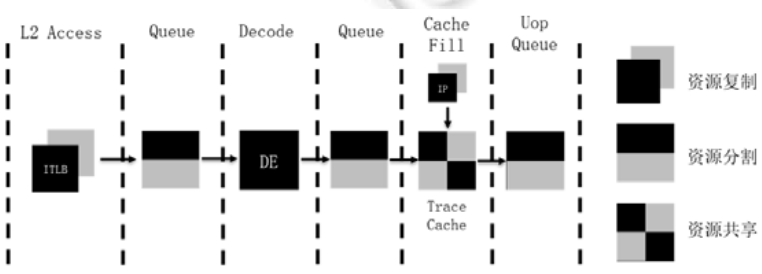

图 2 Intel SMT 前端微架构的处理方式

SMT 技术中资源复制和资源分割的处理方式使得两个线程在处理器内部会公平地使用自己的私有资源, 不会对其他线程的资源产生影响. 因此, 基于共享资源竞争的攻击方式对该类型的微架构组件无效. 对于资源共享的处理方式, 两个线程在处理器内部会产生竞争, 而且由于特点(2)和特点(3)的影响, 两个线程会“公平”地竞争处理器内部资源.

© 中国科学院软件研究所 http://www.jos.org.cn

SMT 环境下的攻击路径主要分为冲突、时间、提取这 3 个阶段: 冲突阶段主要是进行共享资源冲突场景的构建; 时间阶段主要是进行相关时间信道信息的度量; 提取阶段主要是对时间信息的分析, 将时间信息中的有效数据提取转化, 达到数据泄露的目的. 冲突和时间是构建 SMT 环境下时间信道的关键要素.

1.1 攻击案例

SMT 环境下利用时间信道产生的微架构攻击方法可以按照 SMT 技术使用产生的微架构共享资源和攻击方式两个维度进行分类. 2002–2020 年期间, SMT 环境下攻击方法涉及 SMT 技术下微架构组件包括: 分支预测器、指令缓存、指令转换缓冲、执行端口及执行单元、数据转换缓冲、数据缓存、二级缓存和两处缓存微架构特殊组件. 涉及的攻击利用方式涉及隐蔽信道攻击、侧信道攻击以及拒绝服务攻击这 3 种.

从微架构组件分布上, 分支预测器、执行端口及执行单元和数据缓存是研究人员最为关注的 SMT 技术共享的微架构组件; 从攻击利用方式上, 侧信道攻击是当前研究人员使用最多的攻击利用方式(见表 1).

表 1 SMT 环境下的时间信道构建方法

SMT 共享资源	攻击案例	攻击路径			攻击方式
		冲突	时间	提取	
分支预测器	[9–11]	BTB (branch target buffer, 分支目标缓冲)冲突	度量分支刷新延迟时间	RSA (rivest-shamir-adleman, 一种加密算法)	侧信道
	[12–14]	BHT (branch history buffer, 分支历史缓冲)冲突	度量分支刷新延迟时间	RSA	侧信道
指令缓存	[15,16]	缓存结构冲突	度量缓存缺失重载时间	RSA	侧信道
	[17]	多线程事件冲突	度量线程总执行时间		拒绝服务
指令转换缓冲	[18]	TLB (translation lookaside buffer, 转换缓冲)结构冲突	度量 TLB 缺失重载时间	EdDSA (Edwards-curve digital signature algorithm, 爱德华兹曲线数字签名算法)/RSA	侧信道
执行端口及执行单元	[1,19–21]	执行单元冲突	度量局部指令流执行时间	RNG (random number generator, 随机数生成器)/RSA	隐蔽信道 侧信道
	[22,23]	执行端口冲突	度量局部指令流执行时间	P-384 AES (advanced encryption standard, 高级加密标准)	侧信道
数据转换缓冲	[18]	TLB 结构冲突	度量 TLB 缺失重载时间	EdDSA/RSA	侧信道
数据缓存	[24]	缓存结构冲突	度量缓存缺失重载时间	RSA	隐蔽信道
	[25]			AES	侧信道
	[26]			AES	
	[27]			ECC (elliptic curve cryptography, 椭圆曲线加密)	
	[28]			AES	
	[29]	缓存 Bank 冲突	度量缓存 Bank 冲突访问总时间	RSA	侧信道
二级缓存	[24]	缓存结构冲突	度量缓存缺失重载时间	RSA	隐蔽信道
其他	[30]	LRU (least recently used, 最近最少使用)状态共享 目录结构共享	度量缓存缺失重载时间	通信场景 RSA	隐蔽信道 侧信道
	[31]				

1.2 防护方法

2005–2020 年期间, 利用 SMT 技术构建的时间信道从防护策略角度按照时间和冲突两个要素可以分成两个防护方向(见表 2); 从防护实现角度, SMT 技术的时间信道防护方法可以分为基于软件防护、基于软硬件结合防护以及基于微架构防护这 3 种(见表 3).

© 中国科学院软件研究所 http://www.jos.org.cn

表 2 时间要素防护案例防护策略与防护实现对比

防护案例	防护方法	防护策略	防护实现分类	适用的微架构组件	防护具体实现	缺点或不足
[32,33]	时间隔离	时间恒定值法	基于微架构	全部	通过固定访存时间实现	失去低级缓存快速命中的性能收益
[34]	时间隔离	时间阶段固定访问时间	基于微架构	全部	通过固定刷新本地状态实现	增加刷新次数,性能开销大
[35,36]	时间隔离	资源租赁方式分隔时间	基于软硬件结合	全部	通过软硬件结合的方式将共享资源作为被租赁对象来分隔使用时间	改动量大, 需要软硬件配合, 且资源浪费严重
[37–39]	增加全局时间噪声	使时间度量手段精度变差	基于软件	全部	修改时间度量指令或修改对应软件接口	修改时间度量手段会降低度量精度, 且会造成部分正常软件时钟机制无法使用或出现异常

表 3 冲突要素防护案例防护策略与防护实现对比

防护案例	防护方法	防护策略	防护实现分类	适用的微架构组件	防护具体实现	缺点或不足
[24,40]	硬件隔离	使冲突无法产生	基于软硬件结合	全部	通过软件端见 SMT 技术禁用	失去 SMT 技术的收益
[41–44]	硬件隔离	使冲突无法产生	基于微架构	缓存系统	进行缓存结构的隔离,使 SMT 双线程无法在缓存结构上构建冲突	仅能够解决缓存结构冲突导致的时间信道问题, 且会造成资源浪费
[45]	硬件隔离	使冲突无法产生	基于微架构	分支预测器	使用新型预测结构来隔离线程	硬件改动量大
[46]	硬件隔离	使冲突无法产生	基于软件	全部	基于软件需求动态关闭 SMT 技术	在使能安全软件需求时失去 SMT 技术的收益
[42, 47–49]	添加硬件结构噪声	使冲突效果不明显	基于微架构	缓存系统	进行缓存结构替换算法优化和改进	主要针对缓存系统,对 SMT 其他组件无效
[50]	添加硬件结构噪声	使冲突效果不明显	基于软硬件结合	缓存系统	软硬件结合进行替换算法优化	需要软件配合,且主要针对缓存系统
[51]	添加硬件结构噪声	使冲突效果不明显	基于微架构	缓存系统	冲突回滚和替换随机化	主要针对缓存系统,对 SMT 其他组件无效
[52]	添加硬件结构噪声	使冲突效果不明显	基于微架构	二级缓存	实现新型目录结构(独立踢出目录)	针对二级缓存,对 SMT 其他组件无效
[53]	添加硬件结构噪声	使冲突效果不明显	基于微架构	指令和数据转换缓冲	静态隔离和随机填充	针对数据缓冲结构,对 SMT 其他组件无效
[54]	添加硬件结构噪声	使冲突效果不明显	基于微架构	缓存系统	随机缓存映射	主要针对缓存系统,对 SMT 其他组件无效
[55]	审计	进行冲突检测并规避	基于微架构	分支预测器	识别冲突并进行硬件隔离	仅针对分支预测器
[56]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	限制投机访存指令执行	仅对数据访问相关数据结构有效
[57]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	有条件地设置投机指令	仅对数据访问相关数据结构有效
[58,59]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	暂停投机并进行投机提前解决	仅对数据访问相关数据结构有效
[60]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	装载限制和广播限制	仅对数据访问相关数据结构有效
[61]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	设置安全检查点进行审计和隔离	仅对数据访问相关数据结构有效
[62]	审计	进行冲突检测并规避	基于微架构	数据转换缓冲及数据缓存	设计 0 级缓存过滤结构,提前检测冲突并规避	仅对数据访问相关数据结构有效

1.3 TSG模型的限制

当前, 研究人员在进行处理器微架构时间信道安全问题研究过程中, 已逐步建立起成熟的原理模型研究体系和路径, 其中比较有代表性的是文献[6]提出的概率信息流图模型. 但是, SMT 技术下时间信道安全问题尚没有好的分析方法和描述手段. TSG 模型[63]目的是描述处理器微架构侧信道攻击的基本操作和步骤, 适用

© 中国科学院软件研究所 http://www.jos.org.cn

于当前基于瞬态攻击的攻击类型描述, 但在描述 SMT 环境下的时间信道安全问题时存在一定的限制和不足.

TSG 模型按照如下流程进行攻击流程的描述.

① STEP1-启动——攻击者建立一个信道传递环境并且进行一次非法访问;

② STEP2-延迟授权——处理器微架构中出现延迟授权状态进而触发投机指令的执行窗口;

③ STEP3-信息访问——攻击者非法访问私密信息;

④ STEP4-信息(使用或)发送——攻击者将私密信息存储在微架构状态中;

⑤ STEP5-信息接收——攻击者接收到私密信息.

从 TSG 模型的攻击流程的阶段划分可以得出如下规则.

规则 1. TSG 攻击流程的信道传递窗口建立由 STEP2-延迟授权开始, 即授权检查会延迟进行.

规则 2. TSG 攻击流程的私密信息访问建立在投机执行基础上.

规则 3. TSG 攻击流程的信息发送通过微架构状态, 需要具有存储能力的数据结构配合.

但是在 SMT 环境下, TSG 模型的这些局限性会影响 SMT 环境下时间信道安全问题的描述, 具体存在如下不足.

(1) TSG 模型的描述对象建立在处理器微架构设计产生的投机执行上, 针对投机执行产生的信道传递提供模型描述方法, 主要归纳了处理器投机执行中由于对私密数据的非法访问和信道传递; 而 SMT 环境下不仅可以通过投机执行构建信道传递条件, 还可以仅依赖共享数据结构冲突构建时间信道;

(2) TSG 模型的非法访问建立在授权检查的基础上, 而 SMT 环境下并不一定依赖授权检查, 因为两个线程可以同时执行, 在授权检查前, 攻击者线程依然有条件获取时间信道信息;

(3) TSG 模型的信息传递依赖微架构状态存储能力, 而 SMT 环境下时间信道的传递路径不一定依赖微架构状态存储, 在两个线程同时执行的状态下, 可以实时获取时间信道信息;

(4) TSG 模型对 SMT 技术的核心要素表达能力较弱, 在模型建立过程中无法区分 SMT 环境下两个线程的行为以及两个线程共享数据结构对模型的影响; TSG 模型的抽象粒度较粗, 无法精确描述 SMT 环境下两个线程形成可传递的信道环境的流程;

(5) TSG 模型整体偏向于安全问题描述, 相比于其他的瞬态攻击建模方式, 其优点在于安全问题描述较为完整, 但是其描述粒度无法达到程序语义级别, 缺少从 TSG 模型描述到程序语义描述的直接联系, 限制了其应用场景.

下面以 SMT 环境下执行端口攻击 PortSmash[22]为例, 说明使用 TSG 模型描述 SMT 环境下时间信道安全问题的具体限制(如图 3 所示).

(1) 图中延迟授权阶段顶点行为是调度执行操作通过执行端口进入执行单元, 同延迟授权流程阶段的描述不符;

(2) 信息访问和信息发送阶段的信道窗口描述仅仅是执行操作和产生冲突, 并没有在投机窗口下, 且该描述粒度无法准确描述执行端口时间信道产生的原理;

(3) 信息接收阶段, 冲突操作执行完毕后受害者进程操作是正常执行指令, 并没有进行相应授权检查;

(4) 信息接收阶段, 攻击者直接度量执行时间, 时间信道可以直接通过指令流实时采集(例如通过 rdtsc 指令), 并不依赖微架构状态的临时存储能力.

SMT 环境下, TSG 模型的最大问题是对于安全问题的判断会存在误判. 例如图 3 的 SMT 环境下执行端口冲突产生的时间信道建模中, 按照 TSG 模型对安全漏洞的定义是缺乏安全依赖, 判断安全依赖的条件是模型中存在条件竞争的顶点, 但是图 3 中并不存在条件竞争顶点, 因此, 按照 TSG 模型, 该模型下不存在安全漏洞. TSG 模型在 SMT 环境下除了存在描述能力的限制外, 还存在安全问题无法识别的不足.

因此, SMT 环境下时间信道安全问题并不适合完全通过 TSG 模型的既定攻击流程和方法进行建模, 而单纯使用 TSG 模型描述 SMT 环境下时间信道安全问题存在一定的限制.

© 中国科学院软件研究所 http://www.jos.org.cn

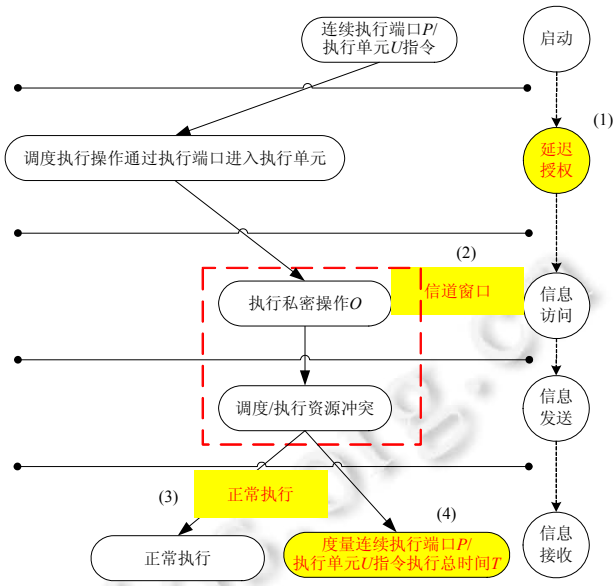

图 3 PortSmash 时间信道 TSG 模型

2 模型设计

本文基于 TSG 模型结合数据流分析进行一定的扩展, 形成一套新的适用于 SMT 技术特征的时间信道安全问题的描述模型——ETSG-SMT.

(1) 安全问题抽象和定义——ETSG-SMT 模型更加关注 SMT 技术产生时间信道的机制和原理, 从冲突和时间两个关键要素进行安全问题的抽象和相关定义的设计;

(2) 模型元素扩展——ETSG-SMT 模型的顶点设计基于 SMT 技术下双线程同时运行的特征将模型顶点扩展为攻击者线程顶点和受害者线程顶点两种, 并在建模方法上进行区分. 同时扩展了攻击者线程制造冲突环境, 进而影响到受害者线程的跨线程“边”. 与普通顶点间直接依赖关系的边不同, 跨线程边主要用于描述 SMT 环境下的双线程的间接依赖关系;

(3) 数据流设计——通过定义数据流块元素以及数据流描述解决模型在信道传递过程对安全问题行为描述粒度不足的缺陷, 可以更加清晰地描述 SMT 环境下两个线程同时运行时产生时间信道传递场景的原理, 建立模型描述到程序语义描述的直接联系;

(4) 攻击流程设计——ETSG-SMT 模型基于 SMT 环境下时间信道的特征将攻击流程重新设计, 配合数据流分析, 使其更加适合 SMT 环境下时间信道安全问题建模;

(5) 防护区域设计——ETSG-SMT 模型基于 SMT 环境下时间信道防护方法的特征, 针对冲突和时间两个要素将防护方法划分了不同的应用区域, 使基于 ETSG-SMT 模型可以快速进行防护方法的分类和新防护方法的推导;

(6) 定理扩展——ETSG-SMT 模型定理在 TSG 模型基础上针对 SMT 环境下时间信道的特征增加了具备冲突环境及时间信道传递条件两个条件竞争的前提, 可以更加准确地描述 SMT 特征.

2.1 形式化描述

ETSG-SMT 模型是一个区分双线程操作的由数据流顶点和顶点之间的边来形成序列的有向无环图, 图中顶点序列具备对应的 SMT 双线程数据流描述.

元素 1(顶点). 表示 SMT 环境下双线程执行的操作, 使用数据流块表示: 矩形块表示攻击者线程操作, 椭圆形块表示受害者线程操作. 数据流块中包含两个域——一个是操作域用于描述程序行为或标签, 一个是资

© 中国科学院软件研究所 http://www.jos.org.cn

源域用于描述微架构组件的资源依赖.

dataflow-block V={*op*,*res*} //数据流块包含操作域和资源域

rectangle-block V∈*attacker thread* //矩形数据流块属于攻击者线程

ellipse-block V∈*victim thread* //椭圆形数据流块属于受害者线程

first-block V=*V.input* //初始的数据流块被命名为 *input*(输入)

op={*clean_predictor*,*train_predictor*,*jmp*,*jcc*,…,*function label*/*operation*} //操作域包含的信息包括清空预测器、训练预测器、跳转、标签、执行操作等等

res={*BPU*(*branch prediction unit*,分支预测单元),*IFU*(*instruction fetch unit*,指令取指单元),*DE*(*decoder*,解码器),*CACHE*(缓存结构),*REG*(寄存器结构),*MOB*(*memory order buffer*,内存排序缓冲),*IU*(*int execution unit*,整型执行单元),*BU*(*branch execution unit*,分支执行单元),…,*micro-architecture components*} //资源域包含的信息包括分支预测器、取指单元、解码单元、缓存、寄存器、内存排序缓冲、执行单元、分支单元等微架构组件

元素 2(边). 表示两个顶点间的依赖, 有向边(带方向实线箭头的线)表示两个顶点间的直接依赖, 有向边(带方向虚线箭头的线)表示两个顶点间的间接依赖. 直接依赖产生于同一个线程内部, 间接依赖产生于两个线程间, 且间接依赖中两个线程有相同的微架构组件资源依赖. 如果两个顶点 *u* 和 *v* 之间有 *u* 到 *v* 的有向边, 表示 *u* 在 *v* 之前发生.

(1) 直接依赖边

- $V_a \in thread_x$ //线程 *x* 顶点 *a*
- $V_b \in thread_x$ //线程 *x* 顶点 *b*
- $E_d=(V_a \rightarrow V_b)$ //线程 *x* 直接依赖边 *d* 连接顶点 *a* 和顶点 *b*
- $E_{d.nxt}=(V_b \rightarrow V_{nxt})$ //线程 *x* 直接依赖边 *d* 的下一条边为顶点 *b* 到下一个顶点的直接依赖边
- $E_{d.adj}=(V_a \rightarrow V_c)$ //线程 *x* 直接依赖边 *d* 的相邻边为顶点 *a* 和顶点 *c* 的直接依赖边

(2) 间接依赖边

- $V_a \in thread_x$ //线程 *x* 顶点 *a*;
- $V_b \in thread_y$ //线程 *y* 顶点 *b*;
- $V_a.res=V_b.res$ //顶点 *a* 和顶点 *b* 具有相同的资源域
- $E_{ind}=(V_a \rightarrow V_b)$ //间接依赖边 *ind* 连接线程 *x* 顶点 *a* 和线程 *y* 顶点 *b*
- $E_{ind.nxt}=(V_b \rightarrow V_{nxt})$ //间接依赖边 *ind* 的下一条边为顶点 *b* 到下一个顶点的间接依赖边
- $E_{ind.adj}=(V_a \rightarrow V_c)$ //间接依赖边 *ind* 的相邻边为顶点 *a* 和顶点 *c* 的间接依赖边

元素 3(路径). 表示连接顶点间的连续有向边, 在 SMT 环境下, 因为双线程同时执行, 连续有向边可以既包含直接依赖顶点, 又包含间接依赖顶点

$P=(V_a \rightarrow V_b \rightarrow V_c \rightarrow \ldots \rightarrow V_n)$ //路径 *P* 表示连续有向边

元素 4(序列). ETSG-SMT 中, 顶点的序列可以由一个带有序列信息的顶点集合表示, 顶点集合间顶点的顺序描述顶点间的依赖关系

$S=[V_a,V_b,V_c,\ldots,V_n]$ //序列 *S* 表示顶点集合

定义 1(线程间依赖). 对于任意两个线程间的顶点操作 *u* 和 *v*, *v* 和 *u* 有线程间依赖指的是: 在 *u* 在 *v* 之前或同时执行时, 会产生线程间共享资源依赖的冲突.

定义 2(条件竞争). 具备线程间依赖及时间信道传递条件的顶点 *u* 和 *v* 之间, 如果有两个不同的有效序列 S_1 和 S_2: 在 S_1 中, *u* 在 *v* 之前; 在 S_2 中, *u* 在 *v* 之后. 则顶点 *u* 和 *v* 之间有条件竞争.

定义 3(安全依赖). 对于任意一对顶点操作 *u* 和 *v*, *v* 和 *u* 有安全依赖指的是: 在 *u* 必须在 *v* 之前完成, 且使得操作 *v* 不会产生线程间依赖, 或无法产生隐蔽信道, 进而可以保证不出现安全漏洞. 操作 *u* 可以表示为安全防护操作, 操作 *v* 表示对于操作 *u* 有安全依赖的操作.

ETSG-SMT 模型中存在条件竞争意味着有安全漏洞存在, 而安全防护方法的设计核心即消除条件竞争,

© 中国科学院软件研究所 http://www.jos.org.cn

通过 ETSG-SMT 模型的条件竞争的定义可以得知, 消除 ETSG-SMT 模型下条件竞争有如下方法.

(1) 消除 ETSG-SMT 模型的线程间依赖;

(2) 消除 ETSG-SMT 模型的时间信道传递条件;

(3) 使 ETSG-SMT 模型中任意两个顶点在不同的序列中不会出现依赖顺序的交叉.

ETSG-SMT 模型符合如下定理.

定理. 对于具备线程间依赖及时间信道传递条件的任意一对顶点 u 和 v, 当且仅当有一条有向路径连接 u 和 v 这两个顶点时, 顶点 u 和 v 则没有条件竞争.

证明:

首先进行必要性证明(采用反证法). 假设不存在一条顶点 u 和 v 的有向路径, 设计一条有效序列 S:

$$S=[s_1,s_2,\ldots,s_k,v,s_{k+1},\ldots,s_n] \tag{1}$$

$$S=[S_v,v,s_{k+1},\ldots,s_n] \tag{2}$$

S 是有效序列, 其中, $S_v=[s_1,s_2,\ldots,s_k]$, 表示序列 S 在顶点 v 前面的顶点序列, 将 S_v 分为 2 个在序列 S 中有相同顺序的子序列($k=k_1+k_2$):

$$S_1=[s_1^1,s_1^2,\ldots,s_1^{k_1}] \tag{3}$$

$$S_2=[s_2^1,s_2^2,\ldots,s_2^{k_2}] \tag{4}$$

S_1 包含的顶点有到顶点 v 的路径, S_2 包含的顶点没有到顶点 v 的路径. 假设顶点 u 没有到顶点 v 的路径, 那么顶点 $u\in S_2$, 可以进一步构造出新的序列:

$$S'=[S_1,v,S_2,s_{k+1},\ldots,s_n] \tag{5}$$

$$S'=[s_1^1,s_1^2,\ldots,s_1^{k_1},v,s_2^1,s_2^2,\ldots,s_2^{k_2},s_{k+1},\ldots,s_n] \tag{6}$$

声明 S'也是有效序列, 那么对于任意的 $s_1^{k_i}\in S_1$ 和 $s_2^{k_j}\in S_2$, 一定不存在连接两者的边($s_2^{k_j},s_1^{k_i}$), 否则, $[s_2^{k_j}\to s_1^{k_i}\to v]$成立, 与 S_2 的定义矛盾. 同理, 也一定不存在有效边($s_2^{k_j},v$).

进一步, 将边(z,x)的类型分为 3 种: $x\in S_1$, $x=v$ 和 $x\in S_2$, 在顶点 x 之前的顶点 z 在 S'序列中存在以下 3 种情况.

① 对于模型中的任意边(z,v), 顶点 z 只能在序列 S_1 中, 且顶点 z 在序列 S'中在顶点 v 之前;

② 对于模型中的任意边($z,s_1^{k_i}$), 顶点 z 只能在序列($s_1^{k_1},\ldots,s_1^{k_i-1}$)中, 且顶点 z 在序列 S'中在顶点 $s_1^{k_i}$ 之前;

③ 因为 S_2 是在序列 S'后面且 $s_{k+1},\ldots,s_n$ 无论是序列 S 还是序列 S'都是在相同的位置, 因此对于图中的任意边($z,s_2^{k_j}$), 顶点 z 在序列 S'中在顶点 $s_2^{k_j}$ 之前.

通过上述的情形①~情形③可以得出: 声明 S'是有效序列成立; 同时, 顶点 v 在序列 S'中在顶点 $u\in S_2$ 之前, 同顶点 u 在所有有效的序列中在顶点 v 之前矛盾. 因此, 该假设不成立, 且必须是只有一条顶点 u 连接顶点 v 的有向路径.

其次进行充分性证明. 定理的充分性是相对明显的, 在不失一般性的前提下, 假设存在顶点 u 和 v 的有向路径, 例如路径 $P=(u,w_1,w_2,\ldots,w_k,v)$, 那么对于任何有效的序列 S, 顶点 u 在顶点 w_1 之前, 顶点 w_1 在顶点 w_k 之前, 顶点 w_k 在顶点 v 之前, 那么顶点 u 在任何有效序列下都在顶点 v 之前.

2.2 数据流扩展

ETSG-SMT 中的数据流描述两个线程产生时间信道传递的程序级流程, 数据流以两个线程的 *input* 块开始, *output* 块结束. *input* 描述数据流输入, 包括线程执行依赖的变量和资源类型; *output* 描述数据流输出, 包括受害者线程结束操作和依赖的资源类型或攻击者获取时间信息 T 和依赖的资源类型.

ETSG-SMT 数据流块的连接关系符合程序语义, 具备正常程序的输入(*input* 块)输出(*output* 块), 同时也符合 ETSG-SMT 模型中顶点序列构成的攻击流程. 在攻击流程中, 每个流程环节根据程序行为特征可以分为一个或多个数据流块, 数据流块的线程通过矩形和椭圆形的“块”区分, 数据流块的连接关系由带方向实线箭头的线连接, 带方向实线箭头的线表示两个数据流块间的线程内执行的程序顺序, 带方向虚线箭头的线表示两

© 中国科学院软件研究所 http://www.jos.org.cn

个线程间数据流快的影响, 即 SMT 环境下一个线程程序执行过程中对另一个线程的影响, 形成线程间依赖.通过数据流的直观描述, 可以更加清晰地表示 SMT 环境下两个线程程序间的相互关系.

ETSG-SMT 数据流的示例如图 4 所示.

基于 ETSG-SMT 的数据流描述方法, 下面以 SMT 环境下特有的执行端口共享产生的攻击方式 SMoTherSpectre[23] 为例, 使用 ETSG-SMT 数据流描述其程序行为.

SMoTherSpectre 攻击采用同 PortSmash 类似的执行端口冲突构造指令来构建用于受害者泄露信息和攻击者度量时间信息的程序代码, 为了增加冲突构造的成功率, 其在冲突构建阶段使用了 Spectre 分支诱导的方式, 实现 SMoTherSpectre 攻击场景的一组程序行为及对应的 ETSG-SMT 数据流图如图 5 所示.

SMoTherSpectre 攻击的 ETSG-SMT 数据流中, 受害者输入私密信息 *secret*、指针地址 *pointer* 和函数标签 *mark*,程序执行依赖资源包括 BPU(分支预测器)、MO*B*(内存排序缓冲)、PORT(执行端口)、BU(分支单元),最后输出由 BU 产生刷新操作(flush)结束; 攻击者输入 *clean_predictor* 完成 BPU 数据结构初始化、诱导地址 *smother* 和函数标签 *loop*,程序执行依赖资源包括 BPU、IU(整型单元)、PORT、REG(寄存器,用于传递时间信息), 最后输出存储在寄存器内的时间信息 *T*.

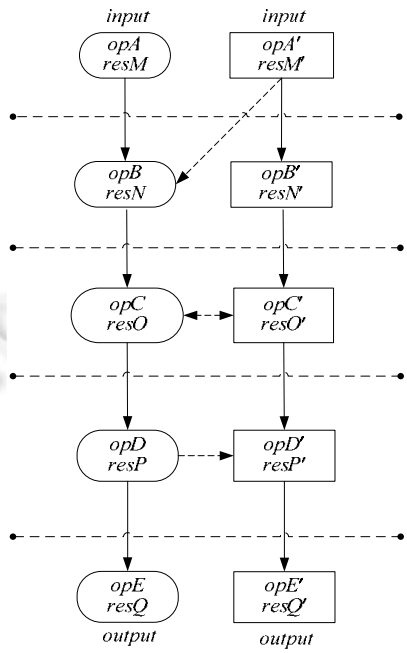

图 4 ETSG-SMT 数据流示例

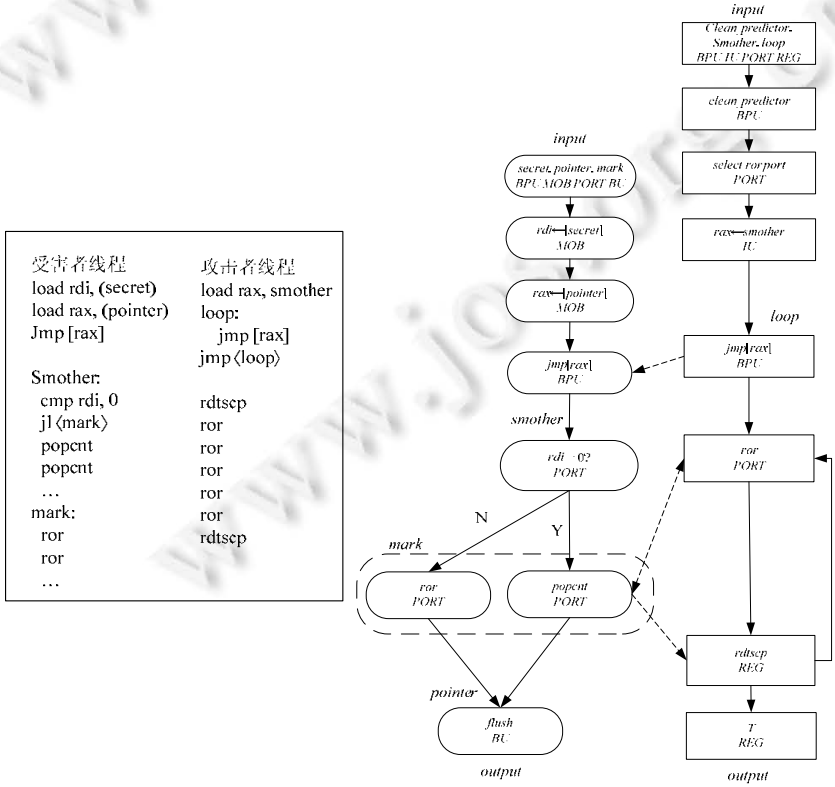

图 5 SMoTherSpectre 攻击的 ETSG-SMT 数据流

© 中国科学院软件研究所 http://www.jos.org.cn

从 SMoTherSpectre 攻击的数据流图可以得出, SMT 环境下 SMoTherSpectre 依赖了两个 SMT 技术下共享资源的冲突, 一个是 BPU, 通过 BPU 资源的冲突攻击者完成受害者分支的诱导; 另一个是 PORT, 通过 PORT 资源的冲突攻击者完成受害者执行路径的判断, 进而获取到受害者私密信息 *secret*.

完成 ETSG-SMT 的数据流设计后, 将具有数据流描述能力的顶点、边以及多个顶点形成的路径、序列按照 SMT 环境下构建时间信道的流程重新整理可以进一步形成 ETSG-SMT 时间信道建模和防护推导的模型.

2.3 模型示例

完整的 ETSG-SMT 模型示例如图 6 所示.针对 SMT 技术的特点, 将 ETSG-SMT 模型的 5 个阶段定义如下.

① STEP1-冲突环境准备——该阶段描述 SMT 环境下的攻击者线程行为, 用于攻击者利用数据结构的共享冲突建立信道传递环境;

② STEP2-冲突操作执行——该阶段描述 SMT 环境下的受害者线程行为, 用于受害者执行攻击者信道传递环境下的冲突操作, 进而触发条件竞争窗口;

③ STEP3-信息访问/执行——该阶段描述 SMT 环境下的受害者线程行为, 用于受害者访问私密信息或执行私密操作;

④ STEP4-时间信道传递——该阶段描述 SMT 环境下的受害者线程行为, 受害者在攻击者设计的冲突环境下被动将私密信息通过微架构状态传递出去;

⑤ STEP5-时间信道接收——该阶段描述 SMT 环境下的攻击者和受害者线程行为, 受害者线程在被动完成时间信道传递后恢复状态或从投机窗口刷新恢复, 同时攻击者线程通过解析时间信道获取私密信息.

基于 ETSG-SMT 模型中的数据流特征, 采用遍历搜索的方式设计 ETSG-SMT 模型条件竞争搜索算法如下.

算法 1. ETSG-SMT 条件竞争搜索算法.

输入: 攻击者线程初始顶点 v_a, 受害者线程初始顶点 v_v, 模型图所有边集合 *E_all*;

输出: 存在条件竞争的边集合 *S_vio*.

```
RaceConditionSearch(v_a,v_v){
  E_initial={e=(v_s,v_d)|e∈E_all &&(v ==v_a||v_s==v_v)};   //连接初始顶点的边
  E_candidate=E_initial;                                   //初始化候选边
  while (E_candidate is not empty) {                       //当候选边不为空时进行搜索
    e=pop E_candidate;                                     //e=(v_s,v_d)
    if (visited(e)==1) continue;                           //visited 为边的标记, 表示已访问过
    visited(e)=1;                                          //设置 visited 标记
    if (e 为安全依赖的边) {                                 //按照 ETSG-SMT 模型定义检查是否存在安全依赖
      if (e.v_s 与 e.v_d 之间存在条件竞争) {                 //按照 ETSG-SMT 模型定义检查是否存在条件竞争
        S_vio=S_vio ∪{e};                                  //将存在条件竞争的边放入边集合 S_vio
      }
    }
    E_candidate=E_candidate ∪{e'=(v_s',v_d')|e'∈E_all && v_s'==v_d}     //搜索后续的边
  }
}
```

基于图 6 的 TSG SMT 模型, 针对冲突和时间两个要素划分不同的防护策略使用区和对应的防护策略映射, 如图 7 所示.

结合 ETSG-SMT 模型消除条件竞争的 3 个方法, 将防护区域设计为消除 SMT 线程间依赖、消除时间信道传递条件以及保持顶点操作依赖顺序唯一性 3 个区域.

© 中国科学院软件研究所 http://www.jos.org.cn

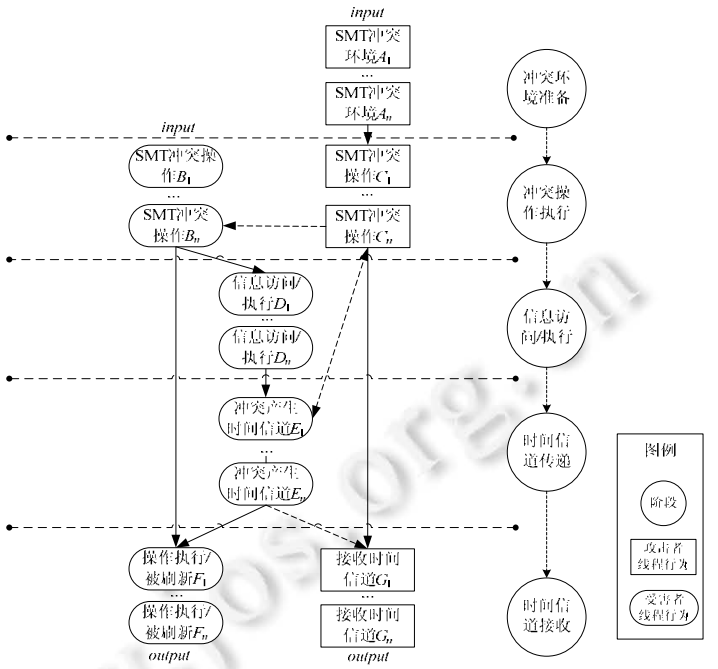

图 6 ETSG-SMT 示例

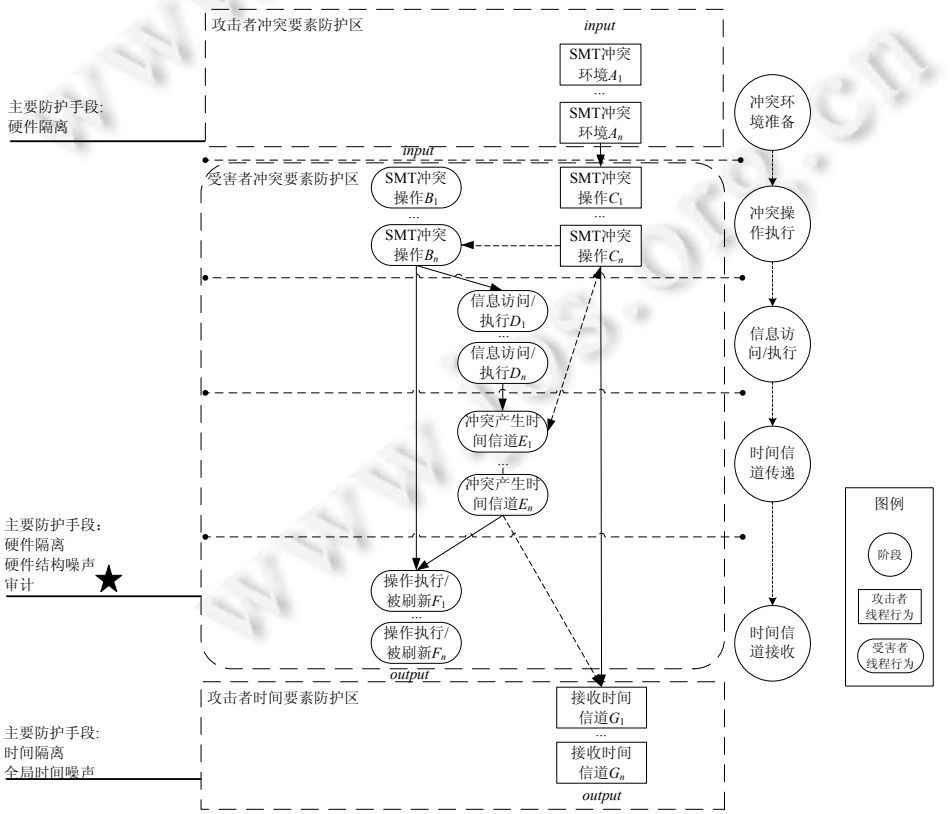

图 7 ETSG-SMT 防护策略映射

其中, 消除 SMT 线程间依赖环境和消除时间信道传递条件均面向攻击者线程, 因此定义为攻击者行为防

© 中国科学院软件研究所 http://www.jos.org.cn

护区, 分为冲突环境准备阶段的冲突要素防护区和时间信道接收阶段的时间要素防护区. 在冲突环境准备阶段, 要让攻击者线程在 SMT 环境下无法准备冲突环境, 只能通过硬件隔离的手段从根本上隔离开两个线程; 在时间信道接收阶段, 要让攻击者线程无法完成对时间信道的接收, 只能通过时间隔离或全局时间噪声的手段进行防护. 攻击者行为防护区对防护方法的难度要求不高, 但是无论是从根本上隔离两个线程还是采用时间隔离或全局时间噪声的方法, 在防护方法的功能或性能开销上比较大.

保持顶点操作依赖顺序唯一性消除条件竞争的方法主要面向受害者线程, 其主要用来切断受害者线程执行过程的条件竞争路径, 从而保证受害者线程的执行行为安全性. 目前业界主流的防护方法主要集中在受害者行为防护上, 重点约束受害者线程行为, 提供更加安全的执行环境. 因此, 该区域定义为受害者行为防护区. 受害者行为防护区主要针对冲突要素, 可以有硬件隔离、硬件结构噪声、审计等多种防护方法应用方向.

2.4 模型有效性分析

Disselkoen 等人[64]提出了一种基于多重偏序集的模型用于描述投机行为, 该模型抽象描述了处理器的微架构组件, 比如缓存结构和流水线, 进而描述以 Spectre 和 PRIME+ABORT 类型的攻击方式并验证其缓解方法的有效性. 但是该模型只关注投机信息的访问, 并没有设计用于描述完整的信道行为, 包括不会描述投机信息如何通过缓存隐蔽信道传输等, 在描述能力上存在明显欠缺. Guarnieri 等人[65]提出了使用投机不干扰属性来验证程序的行为, 主要目的是用于验证在投机流水线环境下投机执行和非投机执行是否会完全相同. 该模型的主要设计目的是提出一种投机执行是否安全的定义, 用于严格形式化推理防护设计的有效性. 该文献还设计实现了 Spectector 工具来基于符号执行算法证明投机不干扰属性的作用. Cheang 等人[66]提出了追踪属性依赖观察机制(trace property-dependent observational determinism, TPOD)来验证不同的执行路径的区别, 该机制不会直接描述攻击或防护方法, 但是可以说明防护方法是否起效果. TPOD 可以通过公式化精确描述瞬态执行的漏洞, 并且具有一定的适用性. 上述两个形式化模型的最主要缺陷是仅仅针对投机执行作为研究对象, 对于 SMT 环境下, 其和 TSG 模型一样, 不具备 SMT 环境下基于微架构数据结构共享产生时间信道安全问题的有效描述能力. 下面从模型描述粒度、模型描述范围、SMT 环境适用性、是否可以工具化和自动化等方面比较几类模型的描述能力, 具体见表 4.

表 4 时间信道描述模型对比

模型名称	描述粒度	描述范围	SMT 环境支持	工具化/自动化
TSG[7]	非形式化行为级	完整攻击和防护流程	部分支持	无法工具化/自动化
Cache[6]	概率信息流图	仅缓存相关	部分支持	有
Pomsets based[64]	形式化行为级描述	仅投机信息利用	不支持	无
Spectector[65]	形式化行为级描述	仅部分投机场景	不支持	有
TPOD[66]	形式化行为级描述	仅部分投机场景	不支持	无
ETSG-SMT	程序数据流级描述	完整攻击和防护流程	支持	暂无, 但有工具化/自动化能力

使用 ETSG-SMT 模型(其模型建立如图 8 所示)可以快速地对 SMT 环境下时间信道进行描述, 下面以 SMT 环境下最为典型的执行端口共享产生的时间信道安全问题“PortSmash”为例说明 ETSG-SMT 模型具有安全问题描述能力以及通过数据流角度分析有效防护需要对模型数据流产生哪些影响, 进而证明模型描述攻击和防护方法的有效性.

PortSmash 攻击[22](如图 9 所示)利用 SMT 技术开启后执行端口由于端口分配和调度策略会产生资源竞争的原理, 在冲突构建阶段, 攻击者创造执行端口冲突的场景, PortSmash 针对 Intel Skylake 的微架构设计了 3 组冲突指令流, 分别适用于仅端口 1 执行, 仅端口 5 执行以及可以在端口 0156 同时执行的冲突构建. 仅端口 1 或端口 5 执行的冲突构建指令选用长延迟指令, 端口 1 选用整型乘法(INT MUL)执行单元执行的 crc32 指令, 端口 5 选用向量交织(VEC SHU)执行单元执行的 vpermd 指令. 可同时在端口 0156 同时执行的指令选用单周期指令, 使用最常见的算术逻辑(INT ALU)执行单元执行的 add 指令. 攻击者线程和受害者线程同时执行, 攻击者使用 rdtsc/rdtscp 指令作为时间度量的手段, 通过在攻击者程序冲突指令执行前后使用 rdtsc/rdtscp 指令可以得出冲突指令的执行时间, 从而获取冲突程度或冲突次数的信息.

© 中国科学院软件研究所 http://www.jos.org.cn

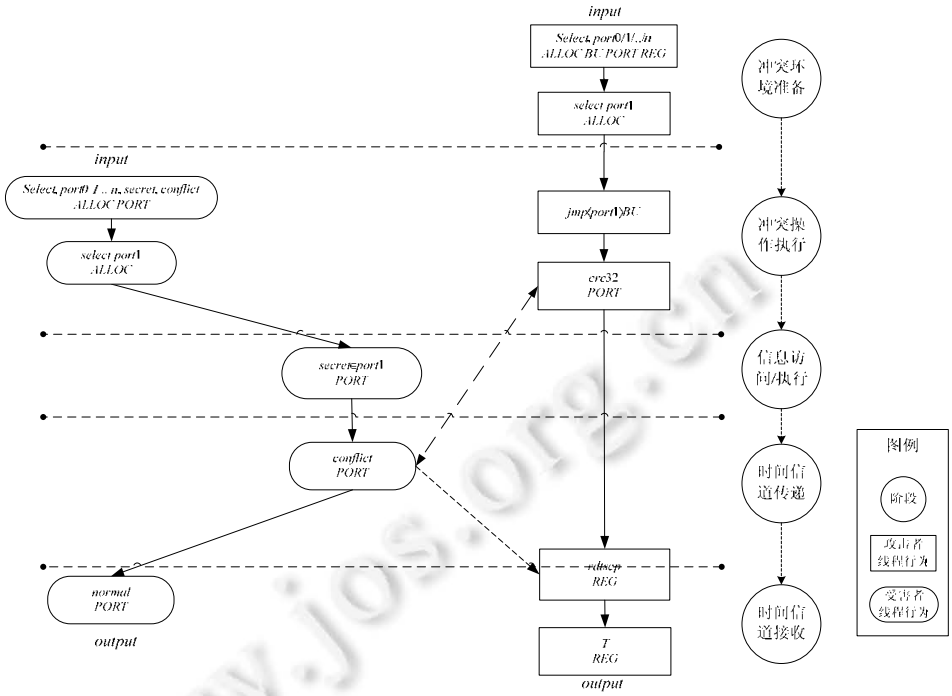

图 8 执行端口 PortSmash 时间信道 ETSG-SMT 模型

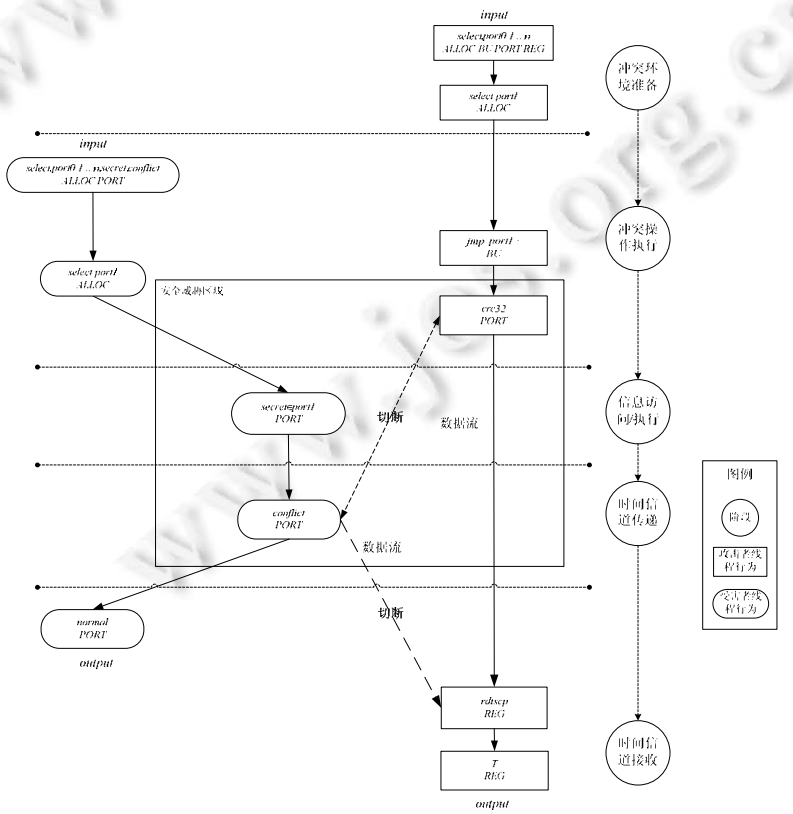

图 9 PortSmash 时间信道模型

© 中国科学院软件研究所 http://www.jos.org.cn

基于该模型可以得知, PortSmash 攻击中攻击者利用了 ALLOC (allocation, 分配资源), BU、PORT 和 REG 共享资源, 攻击者通过不同的攻击函数的调用从而选择不同的端口构建持续的端口冲突场景, 如选择端口 1 则连续执行 crc32 指令. 在此期间使用了 BU 资源进行不同端口的选择, 不同端口的选择同时影响 ALLOC, PORT 资源, REG 资源被攻击者用于存储 TSC 时间信道度量后的结果.从数据流角度, PortSmash 攻击的数据流的主要传递方式是 rdtsc 的返回值, 通过连续指令执行前后的 rdtsc 值的统计获取时间信道信息. 基于 ETSG-SMT 模型, PortSmash 攻击的核心区域和攻击者数据流传递方式如图 9 所示.

3 攻击建模

本节使用 ETSG-SMT 模型对 SMT 环境下部分数据结构的时间信道进行建模描述.

3.1 分支预测器

目前 SMT 环境下分支预测器的攻击方法中, 文献[9–11]在攻击路径上采用了分支目标缓冲结构构造冲突, 文献[12–14]在攻击路径上利用了分支历史表结构及 L2 BPU (level2 branch prediction unit, 二级分支预测器)预测结构构造冲突, 这几种方法均通过度量分支刷新延迟时间来获取时间信道信息, 进而实现信息提取.

在 SMT 环境下使用分支预测器构建时间信道有两种方式, 一种是以 Spectre[14]为代表的瞬态攻击类型, 该类攻击在 SMT 环境下可以使用分支诱导组合数据访问的方式传递信息, 攻击者首先通过训练 BPU 来达到诱导受害者进入错误执行路径. 受害者在错误执行路径的投机窗口下完成访存行为对缓存系统产生的影响, 攻击者通过对缓存系统中的缓存行的访问时间进行度量从而达到信道传递目的.

另一种是 SMT 环境下特有的实时进行分支路径分析的攻击类型, 其核心原理是通过观测分支预测错误引起的惩罚(错误预测后分支刷新引来的延迟)来对程序路径进行分析. 可以在适当的位置执行分支指令, 用以选择性地踢出 BTB/PHT 中的条目, 而后度量受害者程序执行时间进而获取受害者的执行路径. 最终通过数据分析的方式来获取受害者线程的完整执行路径信息.

利用 SMT 环境下分支预测器实时路径分析构建时间信道的 ETSG-SMT 模型建立如图 10 所示.

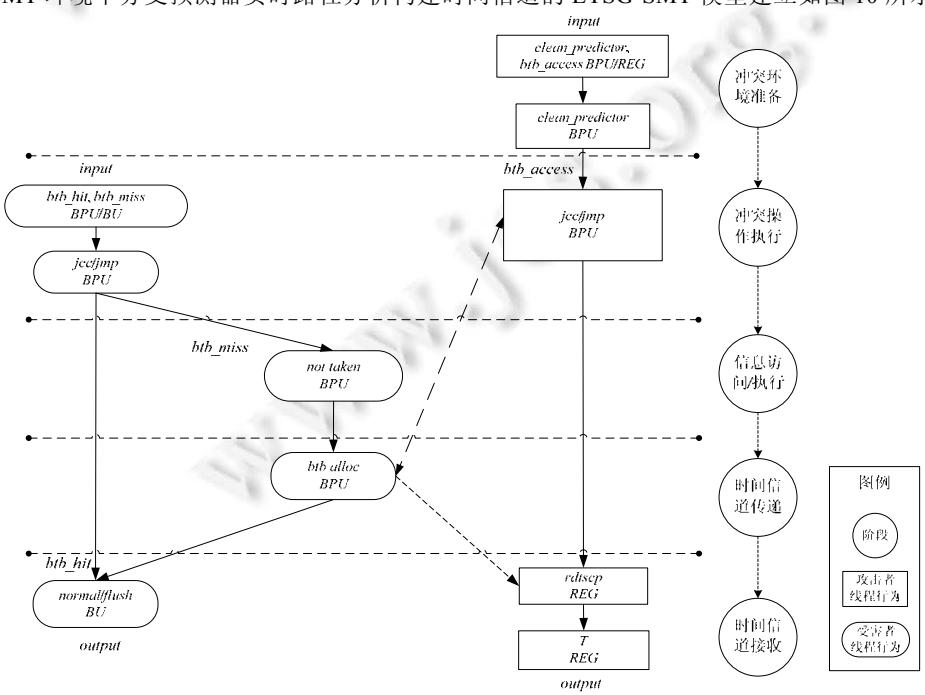

图 10 分支预测单元实时路径分析时间信道 ETSG-SMT 模型

© 中国科学院软件研究所 http://www.jos.org.cn

利用 SMT 环境下分支预测器瞬态投机窗口构建时间信道的 ETSG-SMT 模型建立如图 11 所示.

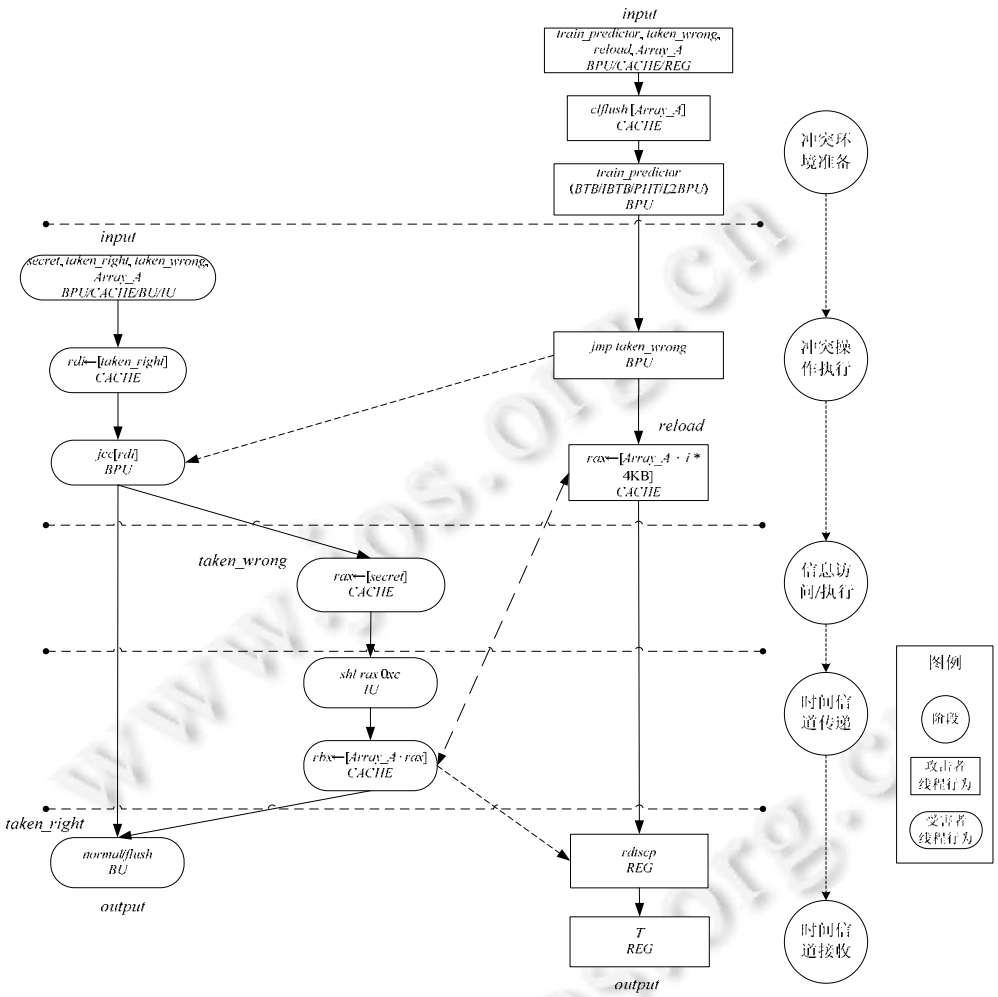

图 11 分支预测单元利用投机窗口的时间信道 ETSG-SMT 模型

3.2 执行端口

文献[22,23]采用双线程共享的执行端口构造冲突, 均通过度量局部指令流执行时间来获取时间信道信息, 进而实现隐蔽信道和侧信道攻击场景, 达到提取受害者关键信息的效果.

SMoTherSpectre[23]攻击是一种新型“Spectre”类型攻击, 其使用 SMT 环境下执行端口的资源竞争构建了投机代码重用的攻击场景, 可以从受害者线程获取私密信息. 比较特殊的是, 该攻击在利用 SMT 环境下执行端口的资源竞争的同时, 还使用了分支预测器诱导的方式控制受害者线程执行错误路径代码, 即组合了分支预测器资源冲突和执行端口资源冲突这两个微架构组件的 SMT 技术特征, 具有更强的适应性和攻击准确性.

文献[23]利用 SMT 环境下执行端口构建时间信道的 ETSG-SMT 模型建立如后文图 12 所示.

3.3 数据缓存

目前 SMT 环境下数据缓存的攻击方法中, 文献[24–28]在攻击路径上采用了缓存结构构造冲突, 通过度量缓存缺失重载时间来获取时间信道信息, 进而实现信息提取.

© 中国科学院软件研究所 http://www.jos.org.cn

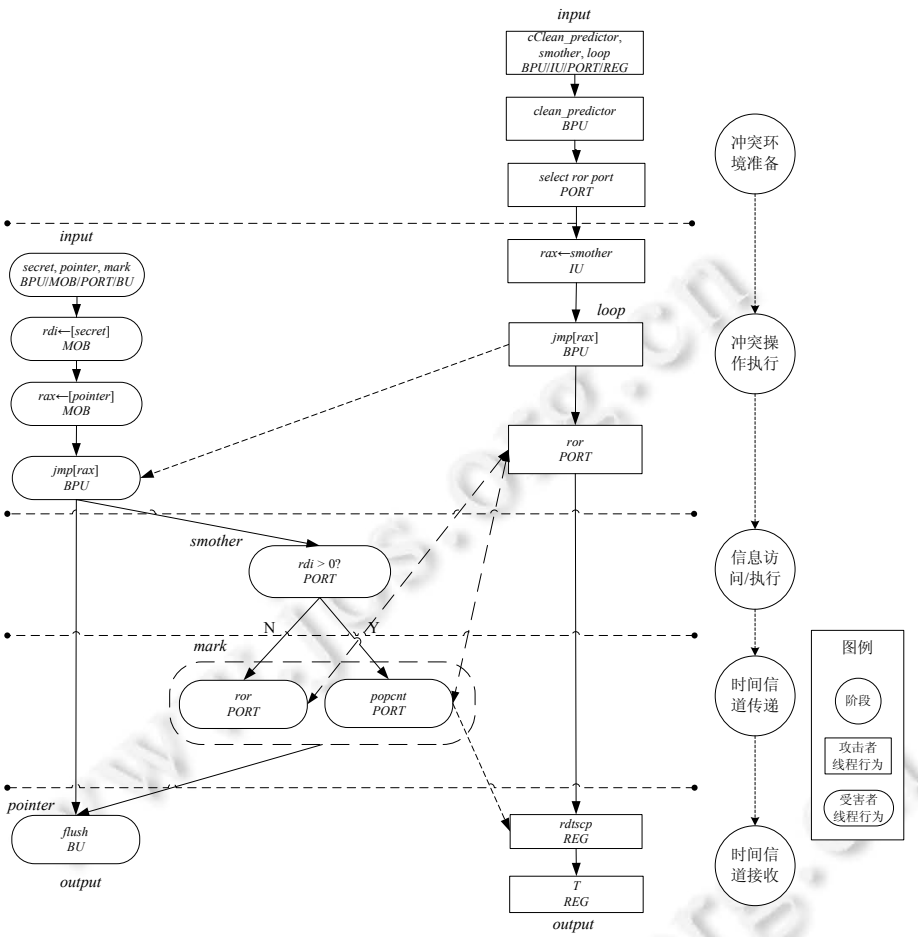

图 12 执行端口 SMoTherSpectre 时间信道 ETSG-SMT 模型

利用 SMT 环境下数据缓存的缓存结构构建时间信道可以使用多种不同的技术手段, 目前在 SMT 环境下可以使用的已知技术原理包括:

(1) “Prime+Probe”, 攻击者首先填充缓存结构, 然后等待受害者执行代码产生缓存结构踢出, 最后攻击者通过访问缓存进行访问时间的度量来获取信息;

(2) “Flush+Reload”, 攻击者首先刷新缓存结构, 然后等待受害者执行代码产生缓存结构装载, 最后攻击者通过访问缓存进行访问时间的度量来获取信息;

(3) “Flush+Flush”, 攻击者首先刷新缓存结构, 然后等待受害者执行代码产生缓存结构装载, 最后攻击者通过刷新缓存指令的执行时间的度量来获取信息;

(4) “Evict+Reload”, 攻击者首先通过缓存加载踢出掉其他缓存结构, 然后等待受害者执行代码产生缓存结构装载, 最后攻击者通过访问被踢出掉的缓存空间进行访问时间的度量来获取信息.

下面针对不同的攻击技术手段对利用 SMT 环境下数据缓存构建时间信道 ETSG-SMT 模型(如图 13–图 16 所示).

© 中国科学院软件研究所 http://www.jos.org.cn

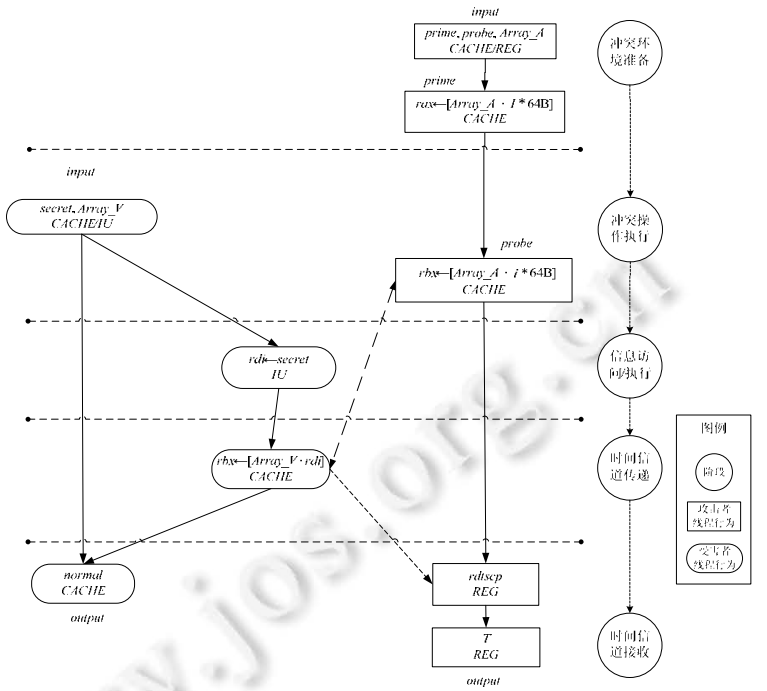

图 13 “Prime+Probe” ETSG-SMT 模型

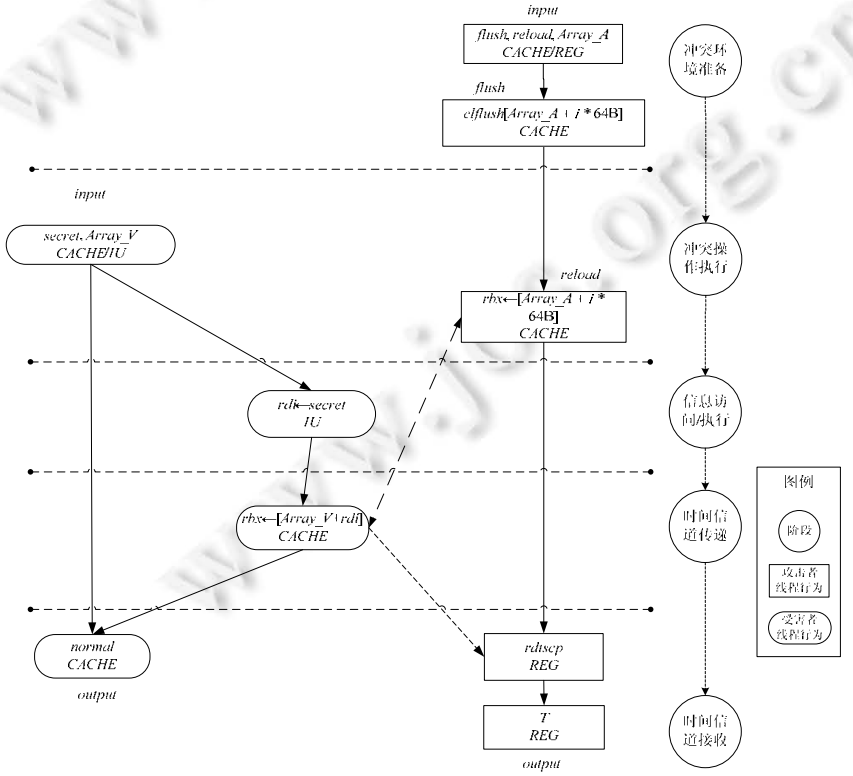

图 14 “Flush+Reload” ETSG-SMT 模型

© 中国科学院软件研究所 http://www.jos.org.cn

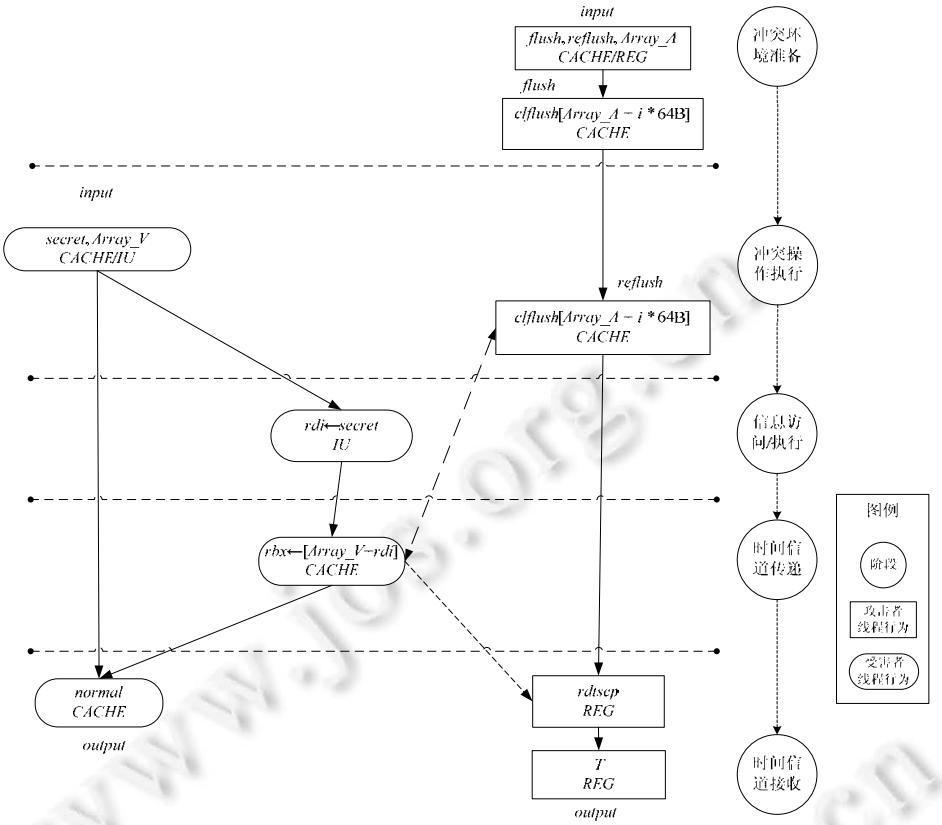

图 15 “Flush＋Flush” ETSG-SMT 模型

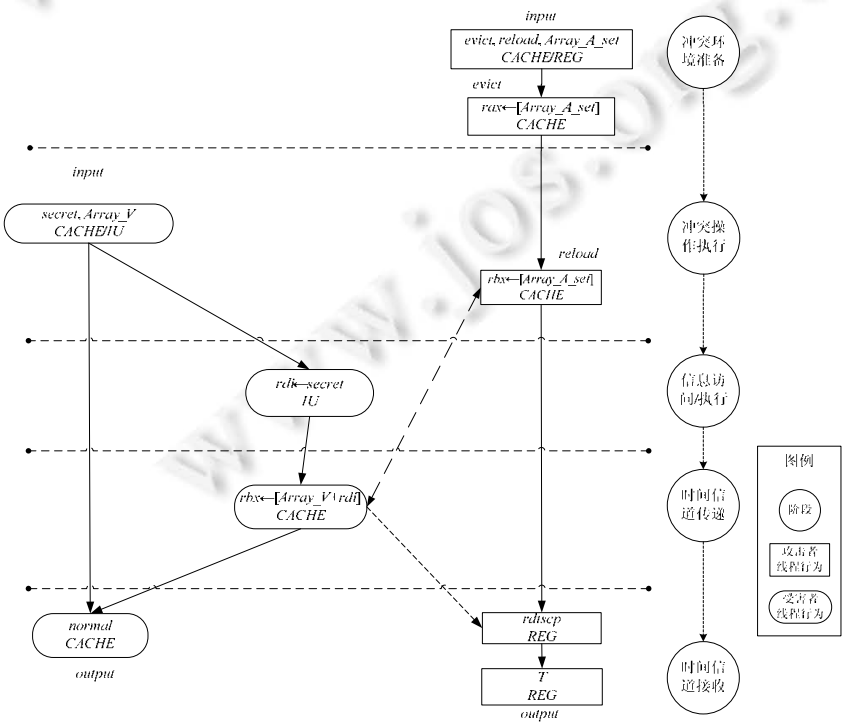

图 16 “Evict+Reload” ETSG-SMT 模型

© 中国科学院软件研究所 http://www.jos.org.cn

4 防护推导

从防护方法角度, SMT 技术的时间信道防护方法按照时间和冲突两个要素可以分成两个防护方向. ETSG-SMT 模型中的防护区域分为攻击者冲突要素防护区、攻击者时间要素防护区和受害者冲突要素防护区 3 个部分.

从时间要素角度, 主要应用在攻击者线程上, 主要限制攻击者对时间要素的获取, 在 ETSG-SMT 模型上主要应用在攻击者时间要素防护区. 从冲突要素角度, 既可以应用在攻击者线程, 又可以应用在受害者线程, 前者在 ETSG-SMT 模型上主要应用在攻击者冲突要素防护区, 目的是切断攻击者可以制造冲突的环境, 隔离两个线程; 后者在 ETSG-SMT 模型上主要应用在受害者冲突要素防护区, 目的是约束受害者行为或者提供给受害者更加安全的执行环境.

下面以 SMT 环境下分支预测器共享导致攻击者诱导受害者进行攻击为例, 从 ETSG-SMT 模型分析攻击所使用的安全依赖及可以采用的防护策略.

图 17 和图 18 是针对 SMT 环境下分支预测器时间信道进行时间阶段和冲突阶段防护方法的 ETSG-SMT 建模.

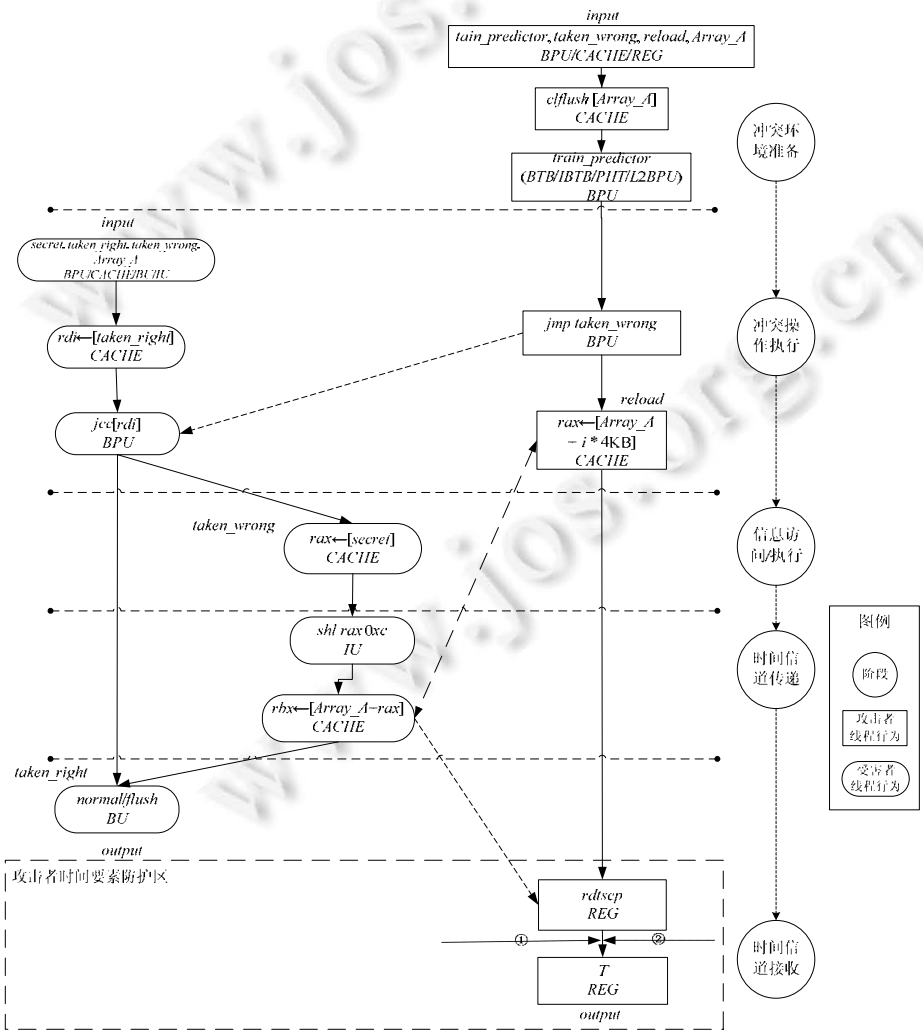

图 17 针对时间要素防护 SMT 分支预测器时间信道的 ETSG-SMT 模型

© 中国科学院软件研究所 http://www.jos.org.cn

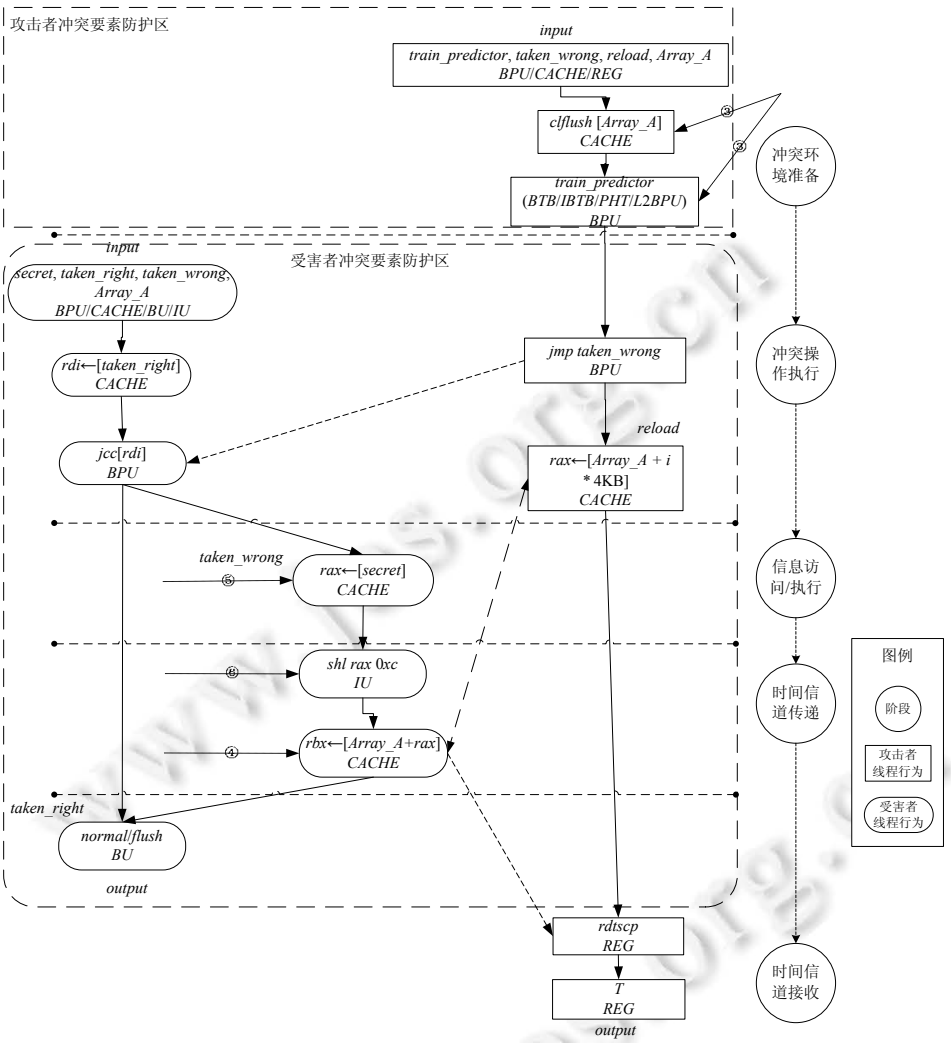

图 18　通过冲突要素防护 SMT 分支预测器时间信道的 ETSG-SMT 模型

采用时间阶段防护主要采用两种方法, 一种是: ① 固定刷新所有数据结构使时间信息固定, 破坏度量时间[34]; 另一种是: ② 修改时间度量指令, 固定或随机返回值[37−39,67].

根据 ETSG-SMT 模型定理, 破坏时间信道传递条件后可以使得条件竞争消除, ①和②的作用都是使得攻击者线程度量时间固定或度量得到的时间不准确, 虽然没有直接破坏 ETSG-SMT 模型的顶点路径来消除条件竞争, 但是使破坏了 ETSG-SMT 模型的时间信道传递条件, 依然可以消除条件竞争, 建立安全依赖环境.

采用冲突阶段防护主要有 4 种方法, 分别是应用于攻击者冲突要素防护区的③隔离双线程的预测结构[24,40,45]/缓存结构[24,40]和应用于受害者冲突要素防护区的④禁止或随机化修改缓存结构操作[41,47−51,53], ⑤表示增加访问私密数据前的审计和检查[55], ⑥表示审计投机数据的使用[55].

根据 ETSG-SMT 模型定理, 破坏线程间依赖后可以使得条件竞争消除, ③的作用是使得攻击者线程无法通过 SMT 环境下数据结构的冲突来构建影响受害者线程的冲突环境, 进而破坏条件竞争的条件. 根据 ETSG-SMT 模型定理, 防护手段④–⑥均是通过约束 ETSG-SMT 模型中条件竞争的顶点路径来达到破坏条件竞争的目的.

在具备线程间依赖和时间信道传递条件的前提下, 具有条件竞争的顶点有两组, 一组是受害者线程内部

© 中国科学院软件研究所 http://www.jos.org.cn

的, 两个顶点分别是受害者线程顶点 $V_{v(rax\leftarrow[secret])}$和受害者顶点 $V_{v(normal/flush)}$; 另一组是受害者线程和攻击者线程之间的, 两个顶点分别是攻击者线程顶点 $V_{a(rax\leftarrow[Array_A+i*4KB])}$和受害者顶点 $V_{v(rbx\leftarrow[Array\_A+rax])}$.

基于 ETSG-SMT 模型定理, 要保证没有条件竞争, 需要破坏上述产生条件竞争路径上的顶点, 已有的防护方法的作用如下.

- ④作用于 $V_{v(rbx\leftarrow[Array\_A+rax])}$, 缓存结构的修改被禁止或者随机化使得装载地址这个数组的新地址无法对缓存行分布产生影响, 该顶点不会产生到受害者顶点 $V_{v(rdtscp)}$度量时间的路径, 进而消除条件竞争;
- ⑤作用于 $V_{v(rax\leftarrow[secret])}$, 增加数据访问的审计和检查环节, 使得无法访问私密数据, 该顶点被破坏后不会产生到条件竞争顶点 $V_{v(rbx\leftarrow[Array_A+rax])}$的路径, 进而消除条件竞争;
- ⑥作用于 $V_{v(shl\ rax\ 0xc)}$, 增加投机数据使用的审计和检查环节, 使得私密数据获取后无法使用, 该顶点被破坏后不会产生到条件竞争顶点 $V_{v(rbx\leftarrow[Array_A+rax])}$的路径, 进而消除条件竞争.

通过上述的模型分析可以得知, 针对受害者冲突要素防护区的防护方法均是针对冲突要素通过直接消除模型中的路径条件竞争来达到防护作用, 是当前研究主流的防护手段.

基于ETSG-SMT模型, 当前SMT环境下执行端口共享导致的时间信道安全问题已有的防护手段中, 使用SMT 禁用或动态关闭的硬件隔离防护方法[24,40,46]应用于攻击者线程冲突要素防护区, 直接破坏攻击者线程构建冲突环境形成线程间依赖的条件来解除条件竞争; 使用修改时间度量指令增加全局时间噪声[37−39,67]或资源租赁[35,36]进行时间隔离的防护方法应用于攻击者线程时间要素防护区, 直接破坏攻击者线程时间信道传递条件来解除条件竞争.

禁用或动态关闭 SMT 技术会直接损失 SMT 技术的性能收益, 即使是以 DDM[46]为例的软硬件防护手段可以有效防护 SMT 环境下执行端口及执行单元攻击, 相比于直接关闭 SMT 技术具有一定的性能优势, 但软件级的策略修改需要用户使用对应的系统软件, 如果不直接集成到主流系统软件中也无法达到真正意义上的安全, 且当攻击者线程存在时, SMT 技术会处于关闭状态, 依然无法体现 SMT 技术的性能优势. 增加全局时间噪声的前提是修改指令语义, 对正常功能会造成影响, 软硬件资源租赁的方法需要完整执行上下文切换, 其性能开销比关闭 SMT 技术的损失还要大, 往往用于对安全性要求较高而不考虑性能的应用场景. 因此, 针对 SMT 环境下执行端口共享导致的时间信道安全问题, ETSG-SMT 模型中的攻击者冲突要素防护区和时间要素防护区的防护方法均对性能有较大影响. 而目前还没有针对受害者线程冲突要素防护区的防护方法出现, 目前主流的基于微架构的防护手段基本上都针对受害者线程的冲突要素防护区进行防护机制的设计, 且在性能开销上相对其他防护方向有较大的优势, 是利用 ETSG-SMT 模型进行防护技术应用推导的研究重点.

4.1 PortSmash攻击防护推导

根据 ETSG-SMT 模型定理, 目前 SMT 环境下执行端口时间信道安全问题缺少针对受害者冲突要素防护区的防护手段, 即没有通过约束 ETSG-SMT 模型中条件竞争的顶点路径来达到破坏条件竞争目的的有效防护方法.

在不破坏线程间依赖和时间信道传递条件的前提下, 按照 ETSG-SMT 条件竞争搜索算法.

$RaceConditionSearch(V_{a(input)},V_{v(input)})$

$S\_vio=\{E_{ind}(V_{a(crc32)}\rightarrow V_{v(conflict)})\}$

$V_{race}=V_{v(conflict)}$

基于 ETSG-SMT 模型定理, 要达到消除条件竞争的目的, 可以从 $V_{v(secret\in port1)}$(③)和 $V_{v(conflict)}$(④)进行防护机制设计. 只要破坏这两个顶点的路径 $P(V_{v(secret\in port1)}\rightarrow V_{v(conflict)})$即可以达到防护目的, 如图 19 所示, ③④是可基于 ETSG-SMT 模型推导的新防护方法.

基于ETSG-SMT模型推导, SMT环境下共享执行端口 PortSmash 攻击新防护方法研究可以从图 19 中③和④两个顶点位置进行防护机制设计, 具体的防护方法推导如下.

- ③作用于 $V_{v(secret\in port1)}$, 目的是使得 SMT 环境下两个线程在分配端口时无法分配到相同的端口, 即双线程执行端口之间在执行过程中是互相隔离的. 在 SMT 环境下, 执行端口是双线程共享的资源, 如

© 中国科学院软件研究所 http://www.jos.org.cn

果要达到该目的, 需要在 SMT 环境下将双线程共享的执行资源隔离开, 即采用 SMT 分割的资源设计策略;

- ④作用于 $V_{v(conflict)}$, 目的是使得 SMT 环境下两个线程在调度执行阶段不产生冲突, 可以通过审计和硬件隔离相结合的方式, 首先进行冲突状态的检测和记录, 然后在调度算法和端口绑定策略上进行双线程执行端口和执行资源的隔离, 消除资源冲突.

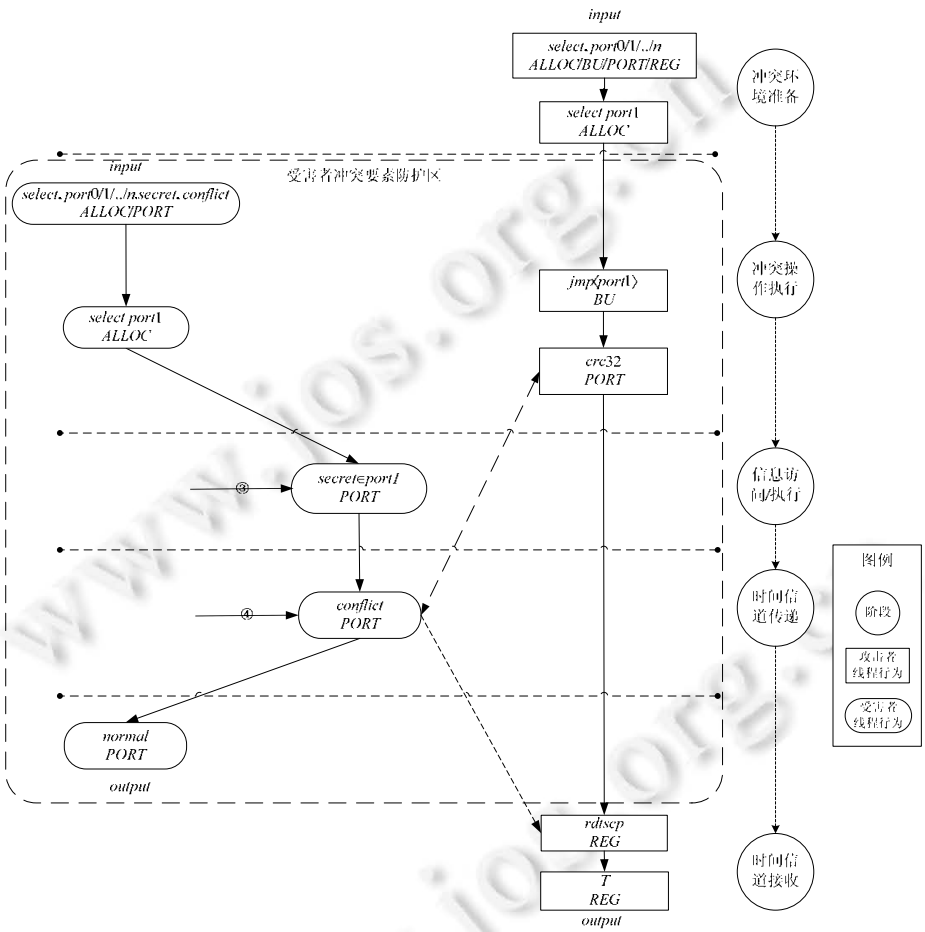

图 19 基于 ETSG-SMT 模型的 PortSmash 攻击防护推导

4.2 SMoTherSpectre攻击防护推导

目前没有通过约束 ETSG-SMT 模型中条件竞争的顶点路径来达到破坏 SMoTherSpectre 模型中条件竞争的有效防护方法.

在不破坏线程间依赖和时间信道传递条件的前提下, 按照 ETSG-SMT 条件竞争搜索算法.

$RaceConditionSearch(V_{a(input)},V_{v(input)})$

$S\_vio=\{E_{ind}(V_{a(jmp[rax])}\rightarrow V_{v(jmp[rax])}),$

$E_{ind}(V_{a(ror)}\rightarrow V_{v(ror)})$ or $E_{ind}(V_{a(ror)}\rightarrow V_{v(popcnt)}),$

$E_d(V_{v(jmp[rax])}\rightarrow V_{v(rdi>0?)})$ and $E_d(V_{v(jmp[rax])}\rightarrow V_{v(flush)})\}$

$V_{race}=[V_{v(jmp[rax])},V_{v(ror)}$ or $V_{v(popcnt)},V_{v(rdi>0?)}]$

可以看出, SMoTherSpectre 区别于 PortSmash 的关键在于 SMoTherSpectre 具有多组条件竞争顶点, 其组合使用了分支预测器共享造成的投机时间窗口进行瞬态攻击, 端口冲突指令只执行在投机时间窗口下, 该特

© 中国科学院软件研究所 http://www.jos.org.cn

征导致产生条件竞争的顶点不再是正常执行的指令, 而是后续会被分支刷新的投机代码, 被投机执行的冲突指令并不属于被攻击者执行的正常指令流. 基于 ETSG-SMT 模型定理, 要达到消除条件竞争的目的, 可以从 $E_d(V_{v(jmp[rax])}\rightarrow V_{v(rdi>0?)})$(④),$V_{a(ror)}$或 $V_{v(popcnt)}$(⑤)和 $V_{v(flush)}$(⑥)进行防护机制设计. 如图 20 所示, ④–⑥是可基于 ETSG-SMT 模型推导的新防护方法.

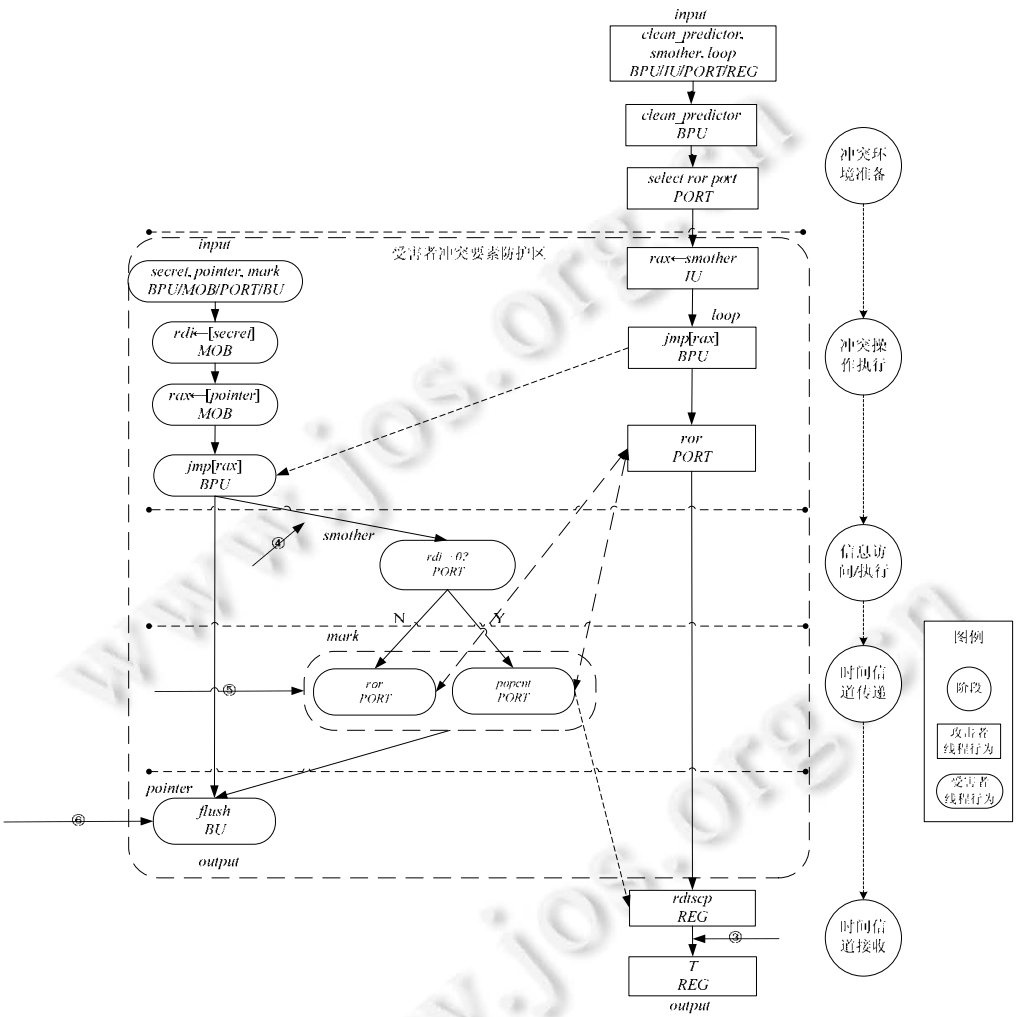

图 20 基于 ETSG-SMT 模型的 SMoTherSpectre 攻击防护推导

SMoTherSpectre 在利用冲突指令时, 仅仅使用了非常小的冲突窗口, 且有冲突产生和没有冲突产生对于攻击者来说都可以获取有效信息, 这样就造成了仅仅使用冲突检测进而动态调整资源使用防护策略无法阻止时间信道产生. 基于 ETSG-SMT 模型推导, 作用于 3 个顶点的具体防护方法推导如下.

- ④作用于 $E_d(V_{v(jmp[rax])}\rightarrow V_{v(rdi>0?)})$, 目的是当带有私密信息的分支指令执行时, 相对于该分支指令更年轻的操作无法产生时间信道, 这样分支指令后的顶点路径则被切断;
- ⑤作用于 $V_{a(ror)}$或 $V_{v(popcnt)}$, 目的是当检测到分支指令时, 在调度算法和端口绑定策略上进行双线程执行端口和执行资源的隔离, 消除资源冲突;
- ⑥作用于 $V_{v(flush)}$, 目的是使得 SMT 环境下当一个线程调度分支指令触发分支刷新时, 记录该分支刷新的线程到对应分支指令记录表中, 作为触发防护机制的使能条件.

© 中国科学院软件研究所 http://www.jos.org.cn

5 总 结

本文通过研究 SMT 环境下时间信道安全问题的底层机理及其防护方法的技术原理, 提出一种 SMT 环境下统一的模型描述方法——ETSG-SMT. SMT 环境下时间信道安全问题被提出以来, 一直缺少有效的描述方式, ETSG-SMT 模型针对 SMT 技术特征和其安全问题的形式化描述特点, 通过一套全新的模型描述方法形成 SMT 技术特点下的攻击原理分析和防护方法设计的模型基础.

SMT 环境下安全问题在攻击方法挖掘和防护技术研究之间出现断层, 以执行端口共享产生的时间信道安全问题为例, 相关攻击方法提出以来还没有出现针对其安全问题的微架构级防护方法和解决方案. 本文提出 SMT 环境下线程间依赖的定义, 通过线程内部和线程间依赖的条件竞争分析可以建立 SMT 攻击原理和防护方法设计的直接联系. 基于 ETSG-SMT 模型对 SMT 环境下时间信道安全问题进行攻击建模和防护描述, 形成针对执行端口共享产生时间信道安全问题的微架构级的有效防护推导. 文中推导的防护方案在后续研究中也已形成基于冲突检测和冲突过滤两种微架构级防护方法的设计, 并在 CPU 模拟器中完成了有效性评估和验证, 这些相关成果也证明了 ETSG-SMT 模型在 SMT 环境下时间信道安全问题防护方法设计中的有效性.

ETSG-SMT 模型作为一种新的 SMT 环境下时间信道安全问题描述模型, 可以有效地描述当前攻击原理和防护策略, 在此基础上还可以扩展应用到更多的安全问题分析和防护方法设计中, 具有一定的研究和应用价值.

References:

[1] Aciicmez O, Seifert JP. Cheap hardware parallelism implies cheap security. In: Proc. of the Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE Computer Society, 2007. 80−91.

[2] Aciicmez O, Koç ÇK. Microarchitectural Attacks and Countermeasures. Cryptographic Engineering. Boston: Springer, 2009. 475−504.

[3] Ge Q, Yarom Y, Cock D, *et al*. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. Journal of Cryptographic Engineering, 2016, 8(1): 1−27.

[4] Canella C, Bulck JV, Schwarz M, *et al*. A systematic evaluation of transient execution attacks and defenses. In: Proc. of the 28th USENIX Security Symp. (USENIX 2019). 2019. 249−266.

[5] Wu XH, He YP, Ma HT, *et al*. Microarchitectural transient execution attacks and defense methods. Journal of Software, 2020, 31(2): 544−563. http://www.jos.org.cn/1000-9825/5979.htm [doi: 10.13328/j.cnki.jos.005979]

[6] He Z, Lee RB. How secure is your cache against side-channel attacks? In: Proc. of the 50th Annual IEEE/ACM Int'l Symp. on Microarchitecture. Cambridge: Association for Computing Machinery. 2017. 341−353.

[7] He Z, Hu GY, Lee R. New models for understanding and reasoning about speculative execution attacks. In: Proc. of the IEEE Int'l Symp. on High-Performance Computer Architecture (HPCA), 2021. 40−53.

[8] Marr DT, Binns F, Hill DL, *et al*. Hyper-threading technology architecture and microarchitecture. Intel Technology Journal, 2002, 6(1).

[9] Acıçmez O, Gueron S, Seifert JP. New branch prediction vulnerabilities in OpenSSL and necessary software countermeasures. In: Proc. of the IMA Int'l Conf. on Cryptography and Coding. Berlin, Heidelberg: Springer, 2007. 185−203.

[10] Acıçmez O, Koç ÇK, Seifert J-P. On the power of simple branch prediction analysis. In: Proc. of the ACM Symp. InformAtion, Computer and Comm. Security (ASIACCS 2007). 2007. 312−320.

[11] Acıiçmez O, Seifert J-P, Koç ÇK. Predicting secret keys via branch prediction. In: Proc. of the Cryptographers' Track at the RSA Conf.. Berlin, Heidelberg: Springer, 2007. 225−242.

[12] Evtyushkin D, Riley R, Abu-Ghazaleh N, *et al*. BranchScope: A new side-channel attack on directional branch predictor. SIGPLAN Not., 2018, 53(2): 693−707.

[13] Huo TL, Meng XN, Wang WH, *et al*. Bluethunder: A 2-level directional predictor based side-channel attack against SGX. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2019, 321−347.

[14] Kocher P, Genkin D, Gruss D, *et al*. Spectre attacks: Exploiting speculative execution. Commun. ACM, 2020, 63(7): 93−101.

[15] Aciiçmez O. Yet another MicroArchitectural attack: Exploiting I-cache. In: Proc. of the 2007 ACM Workshop on Computer Security Architecture. Fairfax: Association for Computing Machinery. 2007. 11−18.

[16] Acıçmez O, Brumley BB, Grabher P. New Results on Instruction Cache Attacks. 2010. 110−124.

© 中国科学院软件研究所 http://www.jos.org.cn

[17] Grunwald D, Ghiasi S. Microarchitectural denial of service: Insuring microarchitectural fairness. In: Proc. of the 35th Annual ACM/IEEE Int'l Symp. on Microarchitecture. Istanbul: IEEE Computer Society Press, 2002. 409–418.

[18] Gras B, Razav K, Herbert B, *et al*. Translation leak-aside buffer: defeating cache side-channel protections with TLB attacks. In: Proc. of the 27th USENIX Conf. on Security Symp. Baltimore: USENIX Association, 2018. 955–972.

[19] Cyber FA. https://cyber.wtf/2016/09/27/covert-shotgun

[20] Wang Z, Lee RB. Covert and side channels due to processor architecture. In: Proc. of the 22nd Annual Computer Security Applications Conf. IEEE Computer Society, 2006. 473–482.

[21] Evtyushkin D, Ponomarev D. Covert channels through random number generator: Mechanisms, capacity estimation and mitigations. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: Association for Computing Machinery, 2016. 843–857.

[22] Aldaya AC, Brumley BB, Hassan S, *et al*. Port contention for fun and profit. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). 2019. 870–887.

[23] Bhattacharyya A, Sandulescu A, Neugschwandtner M, *et al*. SMoTherSpectre: Exploiting speculative execution through port contention. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: Association for Computing Machinery, 2019. 785–800.

[24] Percival C. Cache missing for fun and profit. In: Proc. of the BSDCan. 2005. 1–13.

[25] Osvik DA, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES. In: Proc. of the 2006 the Cryptographers' Track at the RSA Conf. on Topics in Cryptology. San Jose: Springer-Verlag, 2006. 1–20.

[26] Acıiçmez O, Schindler W, Koç ÇK. Cache based remote timing attack on the AES. In: Proc. of the 7th Cryptographers' Track at the RSA Conf. on Topics in Cryptology. San Francisco: Springer-Verlag, 2007. 271–286.

[27] Brumley B, Hakala R. Cache-timing template attacks. In: Proc. of the Int'l Conf. on Advances in Cryptology-asiacrypt. Berlin, Heidelberg: Springer, 2009. 667–684.

[28] Tromer E, Osvik DA, Shamir A. Efficient cache attacks on AES, and countermeasures. Journal of Cryptology, 2010, 23: 37–71.

[29] Yarom Y, Genkin D, Heninger N. CacheBleed: A timing attack on OpenSSL constant-time RSA. Journal of Cryptographic Engineering, 2017, 7(2): 99–112.

[30] Xiong W, Szefer J. Leaking information through cache LRU states. In: Proc. of the 2020 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). 2020. 139–152.

[31] Yan M, Sprabery R, Gopireddy B. Attack directories, not caches: Side channel attacks in a non-inclusive world. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). 2019. 888–904.

[32] Bernstein D. Cache-timing attacks on AES. Technical Report, 2005. 1–37. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[33] Bernstein DJ, Lange T, Schwabe P. The security impact of a new cryptographic library. In: Proc. of the 2nd Int'l Conf. on Cryptology and Information Security in Latin America. Santiago: Springer-Verlag, 2012. 159–176.

[34] Zhang Y, Reiter MK. Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer and Communications Security. Berlin: Association for Computing Machinery, 2013. 827–838.

[35] Tiwari M, Li X, Wassel HMG, *et al*. Execution leases: A hardware-supported mechanism for enforcing strong non-interference. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture. New York: Association for Computing Machinery, 2009. 493–504.

[36] Tiwari M, Oberg JK, Li X, *et al*. Crafting a usable microkernel, processor, and I/O system with strict and provable information flow security. In: Proc. of the 38th Annual Int'l Symp. on Computer Architecture. San Jose: Association for Computing Machinery, 2011. 189–200.

[37] Vattikonda BC, Das S, Shacham H. Eliminating fine grained timers in Xen. In: Proc. of the 3rd ACM Workshop on Cloud Computing Security Workshop. Chicago: Association for Computing Machinery, 2011. 41–46.

[38] Martin R, Demme J, Sethumadhavan S. TimeWarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In: Proc. of the 39th Annual Int'l Symp. on Computer Architecture. Portland: IEEE Computer Society, 2012. 118–129.

[39] Oren Y, Kemerlis VP, Sethumadhavan S. The spy in the sandbox: Practical cache attacks in JavaScript and their implications. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: Association for Computing Machinery, 2015. 1406–1418.

[40] Marshall A, Bugher GHM. Security best practices for developing windows azure applications. Microsoft Corp, 2010, 42: 12–15.

© 中国科学院软件研究所 http://www.jos.org.cn

[41] Page D. Partitioned cache architecture as a side-channel defence mechanism. IACR Cryptology ePrint Archive, 2005.

[42] Wang Z, Lee RB. New cache designs for thwarting software cache-based side channel attacks. In: Proc. of the 34th Annual Int'l Symp. on Computer Architecture. San Diego: Association for Computing Machinery, 2007. 494–505.

[43] Domnitser L, Jaleel A, Loew J, *et al*. Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks. ACM Trans. on Archit. Code Optim., 2012, 8(4): 1–21.

[44] Colp P, Zhang JW, Gleeson J, *et al*. Protecting data on smartphones and tablets from memory attacks. In: Proc. of the 20th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Istanbul: Association for Computing Machinery, 2015. 177–189.

[45] Vougioukas I, Nikoleris, N, Al-Hashimi BM, *et al*. BRB: Mitigating branch predictor side-channels. In: Proc. of the 2019 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). 2019. 466–477.

[46] Zhang Y, Zhu ZY, Meng D. DDM: A demand-based dynamic mitigation for SMT transient channels. 2019. 614–621.

[47] Liu FF, Lee RB. Random fill cache architecture. In: Proc. of the 47th Annual IEEE/ACM Int'l Symp. on Microarchitecture. Cambridge: IEEE Computer Society, 2014. 203–215.

[48] Yan MJ, Gopireddy B, Shull T, *et al*. Secure hierarchy-aware cache replacement policy (SHARP): Defending against cache-based side channel attacks. In: Proc. of the 44th Annual Int'l Symp. on Computer Architecture. Toronto: Association for Computing Machinery, 2017. 347–360.

[49] Kiriansky V, Lebedev I, Amarasinghe S, *et al*. DAWG: A defense against cache timing attacks in speculative execution processors. In: Proc. of the 51st Annual IEEE/ACM Int'l Symp. on Microarchitecture. Fukuoka: IEEE Press, 2018, 974–987.

[50] Kong JF, Acıiçmez O, Seifert JP. Hardware-software integrated approaches to defend against software cache-based side channel attacks. In: Proc. of the 15th IEEE Int'l Symp. on High Performance Computer Architecture. 2009. 393–404.

[51] Saileshwar G, Qureshi MK. CleanupSpec: An "Undo" approach to safe speculation. In: Proc. of the 52nd Annual IEEE/ACM Int'l Symp. on Microarchitecture. Columbus: Association for Computing Machinery, 2019. 73–86.

[52] Yan MJ, Wen JY, Fletcher CW, *et al*. SecDir: A secure directory to defeat directory side-channel attacks. In: Proc. of the 46th Int'l Symp. on Computer Architecture. Phoenix: Association for Computing Machinery, 2019. 332–345.

[53] Qureshi MK. New attacks and defense for encrypted-address cache. In: Proc. of the 46th Int'l Symp. on Computer Architecture. Phoenix: Association for Computing Machinery, 2019. 360–371.

[54] Aga S, Narayanasamy S. InvisiPage: Oblivious demand paging for secure enclaves. In: Proc. of the 46th Int'l Symp. on Computer Architecture. Phoenix: Association for Computing Machinery, 2019. 372–384.

[55] Tan Y, Wei JZ, Guo Wei. The micro-architectural support countermeasures against the branch prediction analysis attack. In: Proc. of the 13th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications. IEEE Computer Society, 2014. 276–283.

[56] Yan MJ, Choi J, Skarlatos D. InvisiSpec: Making speculative execution invisible in the cache hierarchy. In: Proc. of the 51st Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO). 2018. 428–441.

[57] Li PN, Zhao LT, Hou R, *et al*. Conditional speculation: An effective approach to safeguard out-of-order execution against spectre attacks. In: Proc. of the 2019 IEEE Int'l Symp. on High Performance Computer Architecture (HPCA). 2019. 264–276.

[58] Barber K, Bacha A, Zhou L, *et al*. SpecShield: Shielding speculative data from microarchitectural covert channels. In: Proc. of the 28th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT). 2019. 151–164.

[59] Barber K, Bacha A, Zhou L, *et al*. Isolating speculative data to prevent transient execution attacks. IEEE Computer Architecture Letters, 2019, 18(2): 178–181.

[60] Weisse O, Neal I, Loughlin K, *et al*. NDA: Preventing speculative execution attacks at their source. In: Proc. of the 52nd Annual IEEE/ACM Int'l Symp. on Microarchitecture. Columbus: Association for Computing Machinery, 2019. 572–586.

[61] Yu JY, Yan MJ, Khyzha A, *et al*. Speculative taint tracking (STT): A comprehensive protection for speculatively accessed data. IEEE Micro, 2020, 40(3): 81–90.

[62] Ainsworth S, Jones TM. MuonTrap: Preventing cross-domain spectre-like attacks by capturing speculative state. In: Proc. of the 47th ACM/IEEE Annual Int'l Symp. on Computer Architecture (ISCA). 2020. 132–144.

[63] He ZC, Hu GY, Lee R, *et al*. New models for understanding and reasoning about speculative execution attacks. In: Proc. of the IEEE Int'l Symp. on HPCA. 2021. 40–53.

[64] Disselkoen C, Jagadeesan R, Jeffrey A, *et al*. The code that never ran: Modeling attacks on speculative evaluation. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). 2019. 1238–1255.

© 中国科学院软件研究所 http://www.jos.org.cn

[65] Guarnieri M, Köpf B, Morales JF, *et al*. Spectector: Principled detection of speculative information flows. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (SP). 2020. 1–19.

[66] Cheang K, Rasmussen C, Seshia SA, *et al*. A formal approach to secure speculation. In: Proc. of the 32nd IEEE Computer Security Foundations Symp. (CSF). 2019. 288–303.

[67] Hu WM. Reducing timing channels with fuzzy time. Journal of Computer Security, 1992, 1(3–4): 233–254.

附中文参考文献:

[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. 软件学报, 2020, 31(2): 544–563. http://www.jos.org.cn/1000-9825/5979.htm [doi: 10.13328/j.cnki.jos.005979]

岳晓萌(1989－), 男, 博士, 高级工程师, 主要研究领域为操作系统, 计算机架构, 系统安全.

杨秋松(1977－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为操作系统, 软件工程, 系统安全.

李明树(1966－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为操作系统, 软件工程, 分布式系统.

© 中国科学院软件研究所 http://www.jos.org.cn