针对 SGX 的攻击与防御综述

张亚晖[1],赵 敏[2],韩 欢[1]
(1.陆军工程大学,重庆 400035;
2.陆军工程大学,江苏 南京 210007)

摘 要: 如何防止恶意攻击者窃取用户数据或隐私,是当前信息安全领域研究的热难点问题。2013 年,Intel 公司在 HASP 会议上提出了新的处理器安全技术 SGX(software guard extensions,软件保护扩展) ,能够在计算平台上提供一个可信的隔离空间 enclave,用于保障用户代码和数据的机密性和完整性。SGX 是信息安全领域突破性的研究成果,对于个人用户和云计算平台用户都具有重大意义。但同时针对 SGX 的攻击的威胁模型非常强大,SGX 的攻击面不断地被发掘,其防御技术也不断更新。为更好地研究针对 SGX 的攻击与防御技术,介绍了针对 SGX 攻击的威胁模型,总结归纳了针对 SGX 的攻击类型,分析了针对 SGX 攻击的防御措施,并对未来可能的针对 SGX 的攻击与防御技术进行了探讨。

关键词: SGX; 威胁模型; 可信计算基; 攻击类型; 防御措施
中图分类号: TP311 **文献标识码**: A **文章编号**: 1673-629X(2020) 11-0104-07
doi: 10.3969/j.issn.1673-629X.2020.11.020

Survey of Attacks and Defenses against SGX

ZHANG Ya-hui[1],ZHAO Min[2],HAN Huan[1]
(1.Army Engineering University of PLA,Chongqing 400035,China;
2.Army Engineering University of PLA,Nanjing 210007,China)

Abstract: How to deal with the increasingly severe information security situation and prevent malicious attackers from stealing user data or privacy is a hot and difficult problem in the current information security field. In 2013,a new processor security technology SGX (software guard extensions) is proposed by Intel at the HASP conference,which can provide a trusted zone named enclave on the computing platform to protect the confidentiality and integrity of user code and data. SGX is a breakthrough research achievement in the field of information security,which is of great significance to individual users and tenants of cloud computing platforms. However,the threat model of SGX is quite strong,the attack surface against SGX is constantly being explored,and its defense technology is constantly updated. For in-depth research on SGX attack and defense technologies,we introduce the threat model of SGX,summarize the attack types against SGX,analyze the defense measures against SGX attacks,and discuss future possible attack and defense techniques against SGX.

Key words: SGX; threat model; trusted computing base; attack vectors; countermeasures

0 引 言

计算机和互联网技术的飞速发展已极大地改变了人们的生活方式,但在此过程中的计算机安全防护和隐私数据保护一直是学术界和工业界研究的热难点问题。目前无论是个人计算机还是第三方云计算平台,它们所采用的都是基于分层构建的安全模型,即从下到上分别为物理硬件(如 CPU、内存) 、特权软件(如操作系统、hypervisor[1,2]) 、软件堆栈(如数据库、网络协议) 、用户程序等。分层的安全模型要求特权级软件可以访问用户程序,而用户程序只能有限地调用特权软件开放的接口。其目的就是为了保护特权软件免受用户程序(通常被视为不受信任代码) 的攻击,但这种模型的负作用就是用户的隐私信息不能免受特权软件的访问。因此,用户的隐私信息在个人计算平台或者第三方云计算平台中,通常基于以下假设: (1) 计算平台所提供的软件和硬件都是可信的; (2) 计算平台的工作人员都是可信的; (3) 计算平台所处的司法地域隐私保护法是可信的。

收稿日期: 2019-12-25 **修回日期**: 2020-04-27
基金项目: 江苏省自然科学基金(BK20180080)
作者简介: 张亚晖(1992-) ,男(回) ,硕士研究生,研究方向为可信计算、系统软件安全; 赵 敏,副教授,研究方向为智能终端安全、系统软件安全和移动目标防御。

2013 年,Intel 提出的处理器安全技术 SGX[3-5] 是 Intel 实现可信执行环境(TEE) 的方式,目的是为在所有特权软件都可能是恶意的计算机上执行的安全敏感计算提供完整性和机密性保护。文中对目前针对 SGX 的攻击与防御技术进行了梳理,文章组织结构为: 第 1 部分介绍了 SGX 的威胁模型,第 2 部分总结分析了针对 SGX 的攻击类型,第 3 部分指出了针对 SGX 攻击的防御措施,第 4 部分探讨了针对 SGX 的攻击与防御技术可能的发展方向,最后对文章进行了总结。

1　SGX 的威胁模型

SGX 攻击的目标是破坏运行在 enclave 中应用的机密性和完整性,攻击者来自 non-enclave 部分,包括应用程序和系统软件。系统软件包括操作系统,hypervisor,SMM,BIOS 等特权级软件。

针对 SGX 的攻击通常假设攻击者已经获取到了除 enclave 包以外的所有资源,如硬件设备、完全访问 OS 资源、ROOT 权限等; 同时攻击者可以安装任意的内核模块并配置计算机的启动参数。攻击者利用获得的 ROOT 权限和其他资源可以分析程序的源代码或二进制执行程序,从而知道目标 enclave 程序可能的控制流; 任意时刻、次数地中断目标 enclave 程序的执行; 为目标 enclave 程序的执行指定特定的物理核心; 访问配置其他的硬件资源或性能统计监视器等等。

2　SGX 攻击

SGX 的攻击者通过破坏 enclave 的机密性可以获得存储在 enclave 中的用户隐私代码或数据,此类攻击方式主要有侧信道攻击、代码重用攻击和硬件漏洞攻击。而攻击者破坏 enclave 的完整性则只能进行拒绝服务攻击。其中侧信道攻击主要利用了 CPU 缓存来获取 enclave 中的控制流和数据流,基于 Cache 的侧信道攻击要比其他类型的侧信道攻击粒度更小,最小粒度可以做到一个 Cache line,从而获取更多的信息。代码重用攻击主要利用了现代编程中不可避免的软件漏洞来获取隐私数据,相比其他类型攻击,代码重用攻击更加隐蔽。硬件漏洞攻击主要利用了 CPU 硬件中的漏洞,该类攻击虽然危害性更大但通常可以及时消除。拒绝服务攻击则主要利用了 SGX 本身的安全机制,虽然该类攻击不能获取用户的隐私数据,但会给公共云提供商带来极大威胁。

2.1　侧信道攻击

2.1.1　基于 cache 的攻击

在现代计算环境中,硬件资源的普遍共享使得并行计算得到了广泛的应用,但这却带来了新的安全隐患——基于 cache 的侧信道攻击。Intel SGX 依赖于硬件,即使操作系统和其他软件堆栈是恶意的,SGX 也可以做到有效的隔离保护,但它却无法防御基于 cache 的侧信道攻击。基于 cache 的侧信道攻击通常采用 Prime+Probe[6-7] 攻击技术,攻击成功与否的关键在于降低攻击过程中的噪声。Prime+Probe 攻击的方法为: Prime,攻击者用预先准备的数据填充特定的多个 cache 组; Trigger,攻击者等待目标进程响应服务请求,将 cache 数据更新; Probe,攻击者重新读取 Prime 阶段填充的数据,度量并记录各个 cache 组读取时间,如果攻击者观察到较高的探测时间,则推断受害者使用了这部分缓存,否则未使用。降低噪声的方式通常有: (1) 分配一个特定的物理核心用于攻击进程和 enclave 的执行; (2) 提高攻击的时空分辨率。

文献[8]实现了一个名为 CacheZoom 的侧信道攻击工具,它应用 Prime+Probe 技术攻击 L1 缓存来收集目标 enclave 的内存访问信息。CacheZoom 通过分配一个专用的物理核心,并通过减少 enclave 在两个中断之间的内存访问次数实现了一个低噪声的侧信道。

文献[9]通过分配专用物理核心,对 L1 缓存执行 Prime+Probe 攻击并采用较高频率的性能监视计数器 PMC 对其进行监测,实现了不需中断 enclave 执行的侧信道攻击技术。该方案可以避开目前已知的侧信道攻击检测方法,且不需要攻击过程与 enclave 进程同步。

2.1.2　基于页表的攻击

在计算机系统中,CPU 只能通过逻辑地址访问进程,但内存仅能识别物理地址。而页表是用来存储逻辑地址和物理地址之间映射的数据结构。

文献[10]介绍了一种新型的针对 SGX 的无噪声的侧信道攻击,称为受控信道攻击。Intel SGX 允许操作系统完全控制 SGX 程序的页表,而页表可以映射或取消映射 SGX 程序的内存页,这使得恶意操作系统能够通过监视页面错误而准确地知道受攻击的 SGX 程序试图访问哪些内存页面。作者使用该受控信道从广泛使用的文字处理工具(FreeType 和 Hunspell) 中提取出了文本文档,获得了由 libjpeg 解压的 JPEG 图像的轮廓,并可以撤消 windows 风格的 ASLR。在每种情况下,只要运行受害者的代码,就足以泄漏受保护应用程序中的数据。作者在 Haven[11] 和 InkTag[12] 两个屏蔽系统中进行了验证。

文献[13]指出在针对 Intel SGX 的攻击中,基于页表攻击所带来的威胁已超出了传统的基于页面的攻击。文献基于强对抗假设,包括攻击者能够完全控制特权软件和 OS 的调度策略、可以反复的中断 enclave 包以及知道目标应用程序的(编译) 源代码,提出了一

种新的基于页表的攻击技术,可以在指令级粒度上精确地中断一个 enclave; 并且提出了两个新的攻击向量: 页表条目监控,重复访问监控。它们可以从页表属性以及不受保护的页表内存的缓存行为中推断出 enclave 的内存访问。文献通过从流行的 Libgcrypt 加密软件套件中恢复很少甚至没有噪声的 EdDSA 会话密钥演示了该攻击的有效性。

2.1.3 基于 LBR 的攻击

分支预测是现代流水线处理器最重要的特性之一。通常指令流水线由四个主要阶段组成: 获取、解码、执行和回写。这种流水线结构使处理器执行一条指令的同时可以获取/解码下一条指令,并将上一条指令的结果存储到内存(或缓存) 中,即处理器可以并行执行多条指令,这样有利于提高处理器的效率。Intel 提供了一个专门的硬件特性 LBR(last branch record) 来记录这些分支信息。

文献[14]中作者介绍了一种新型的针对 Intel SGX 的侧信道攻击,它可以识别一个运行在 SGX 硬件上的 enclave 程序的细粒度(块级) 控制流,称为分支追踪攻击。这种攻击的原理是当处理器从 enclave 模式切换到 non-enclave 模式时,Intel SGX 没有清除分支历史信息,并通过分支预测侧信道将细粒度的跟踪信息留给了外部不可信系统。但该攻击存在两个挑战: (1) 根据定时来度量分支预测/误预测对于区分细粒度控制流的变化很不准确; (2) 需要对 enclave 进行精细复杂的控制来使得它执行攻击者感兴趣的代码块。为了克服这些挑战,作者开发了两种新的攻击技术: (1) 利用 Intel PT 和 LBR 来正确识别分支历史; (2) 调整本地 APIC 定时器来精确控制一个飞地内的执行,从而使分支追踪攻击非常精确。作者演示了利用分支追踪攻击来推断 Intel SGX SDK、mbed TLS、LIBSVM 和 Apache 中的细粒度的控制流且在此过程中不会引发页面错误,并且对于保护 Intel SGX 免受页面错误和缓存定时攻击的一些方案(如确定性多路复用[15]、T-SGX[16]、SGX-Shield[17]和 Sanctum[18]) ,分支追踪攻击同样有效。

2.1.4 基于 DRAM 的攻击

DRAM[19]一般由 channel,DIMM,rank,bank 等部分构成,每个 bank 又由 columns、rows 和 row buffer 组成,其中 row buffer 用来缓存最近访问过的一个 row。与 CPU 的缓存访问模式类似,在进行 DRAM 访问时,如果访问的 row 已经被 row buffer 缓存,则直接从 row buffer 中读取,否则将整个 row 加载到 row buffer 中再进行读取。如果 row buffer 中已缓存了其他 row,则需要先换出 row buffer 的内容再加载新的 row 进行读取。这几种访问模式的速度均不同,攻击者可以利用访问时间的差异判断当前访问的 row 是否在 row buffer 中或被换出。

文献[20]演示了一个运行在 SGX enclave 中的恶意软件,因为所有的 enclave 都位于相同的物理 EPC 中,所以运行恶意软件的 enclave 可以对其他 enclave 执行基于 DRAM 的攻击以窃取用户的敏感数据。该方案不需要依赖恶意的操作系统,即攻击者也只是一个非特权应用程序,唯一的要求是攻击进程和 enclave 进程位于同一主机中。

2.2 代码重用攻击

计算机软件的发展经验表明,任何应用程序都存在着安全漏洞。在传统环境中,此类漏洞通常允许攻击者完全地控制系统。虽然 SGX 声明可以对软件提供强有力的保护,但如果在 enclave 代码中存在着安全漏洞会有什么后果和危害?

文献[21]全面分析了针对 enclave 内部漏洞的利用技术,提出了一种新的基于面向返回编程的代码重用攻击方法 Dark-ROP。Dark-ROP 通过构建可以通知攻击者 enclave 执行状态的 oracle,从而在代码和数据都隐藏时启动 ROP 攻击。(1) 从 enclave 内存中获取隐私代码和数据; (2) 绕过本地和远程的 enclave 认证; (3) 解密和生成正确加密的数据。此外,Dark-ROP 可以通过构建一个由攻击者完全控制的 shadow enclave,并将受害 enclave 中的代码和数据提取到其中来模拟受害 enclave,如读取 enclave 的 SGX 加密密钥。

文献[22]基于弱对抗假设,即不需要拥有内核特权,提出了针对 SGX 的第一个用户空间内存的代码重用攻击。作者提出了两个新的开发原语: ORET 和 CONT,它们能够利用 SGX 异常处理的内在特性以及 enclave 代码与不可信代码的交互过程来实现对所有 CPU 寄存器的访问。同时,攻击能够破坏现有的细粒度随机化方案而又不至于使任何的飞地崩溃,如 SGX-shield,且它适用于 Linux 或 Windows Intel SGX SDK 开发的 enclave。

2.3 硬件漏洞攻击

L1TF[23]是 2018 年发现的一种 Intel 处理器安全漏洞,在使用推测执行的微处理器和英特尔 SGX 的系统中可能在未经授权的情况下就将本地用户访问的 enclave 驻留在 L1 数据缓存中的信息泄露给攻击者。L1TF 的发现者利用该漏洞开发了名为 Foreshadow[24]的攻击,它可以在没有 ROOT 特权,不知道受攻击 enclave 代码的情况下发起攻击; 甚至如果攻击者获得了 ROOT 特权,不需要受害 enclave 的执行就可获取 enclave 中的敏感信息。L1TF 漏洞破坏了 SGX 的安全保证,但它最严重的后果是 L1TF 可以转储 L1 数据缓存的全部内容,而不管数据的所有者是谁。

Intel 在 Foreshadow 之后发现了两种密切相关的变体,统称它们为 Foreshadow-NG[25],这是第一个完全破除虚拟内存沙箱的瞬态执行攻击,传统的页表隔离已不足以防止未经制授权的内存访问。文献[25]中讨论并分析了三种 Foreshadow 攻击变体: Foreshadow-OS、Foreshadow-VMM 和 Foreshadow-SGX。其中 Foreshadow-OS 的机理是执行用户空间代码的无特权攻击者控制第一个页表遍历的虚拟地址输入,当需要将内存中的页面交换到磁盘时,攻击者只需要等待操作系统清除某些 PTE 条目中的 PTE 表示位就可以导致终端错误。此时,可以使用瞬态无序指令读取位于 PTE 条目所指向的物理地址的任何缓存内容。Foreshadow-VMM 的机理是恶意客户虚拟机控制第一个地址映射,因此可以通过清除客户页表中的当前位直接触发终端错误。由于终端故障行为跳过了主机地址转换步骤,并立即将客户物理地址传递给了 L1 缓存,因此攻击者可以临时读取系统上任何缓存的物理内存,包括属于其他虚拟机或管理程序本身的内存。Foreshadow-SGX 的攻击机理如文献[24]中所演示的,控制最终地址转换输出的攻击者在对缓存的 enclave 机密进行瞬时计算时,可以滥用终端错误来绕过 SGX 中断页面语义机制。攻击者可以通过清除页表当前位(例如,通过 mprotect 系统调用)或在攻击者控制的飞地中设置恶意内存映射来触发终端错误。

2.4　拒绝服务攻击

Intel SGX 的完整性是使用完整性树来进行验证的,当处理器检测到任何完整性违规时,会进行处理器锁定,以防止进一步的损坏,此时,若想恢复系统只能对系统进行冷启动。通常想要破坏飞地的完整性只能采用硬件攻击的手段,但如果攻击者有基于软件的方法来破坏飞地的完整性,那么处理器锁将会导致严重的拒绝服务攻击。

文献[26]提出了第一个破坏飞地完整性的基于软件的攻击—SGX-Bomb。它可以对 enclave 内存发起 Rowhammer 攻击[27]以触发处理器锁定。它首先在同一个 DRAM 库中发现冲突的行地址,然后重复访问它们,如果由于 Rowhammer 攻击而在 enclave 内发生任意位翻转,则对 enclave 内存的任何读取操作都会导致飞地的完整性检查失败,从而导致处理器被锁定,此时,只能通过重新启动系统处理。这对公共云提供商而言是极具威胁的。因为这些提供商从客户端接收到未知的 enclave 程序并运行,但这些程序却可能会关闭与其他客户共享的服务器。而且 SGX-Bomb 基于非常简单的假设,不需要有 ROOT 特权,不需要有物理接触,只需计算机处理器支持 SGX,计算机的 DRAM 模块具有 Rowhammer 漏洞以及用户级别的 enclave 执行环境。作者利用 DDR4 DRAM 在真实环境中对 SGX-Bomb 攻击的有效性进行了评估,使用默认的 DRAM 刷新率(64 ms)花费了 283 秒就使整个系统停止了响应。

3　SGX 防御

针对上述介绍的 SGX 攻击方式,目前学术界研究的 SGX 防御技术主要包括: 基于 enclave 执行频繁被打断特征的异常检测技术、基于指令集和地址空间布局随机化的随机化技术、隔离 SGX 的整个攻击面的增强隔离技术,以及修改源码实现隐藏 enclave 程序的控制流和数据流的源码重构技术。具体如表 1 所示。

表 1　SGX 防御技术

SGX 防御技术	侧信道攻击	代码重用攻击	硬件漏洞攻击	拒绝服务攻击
异常检测	√	×	×	×
随机化技术	√	√	×	×
增强隔离	√	√	×	×
源码重构	√	√	×	×

其中异常检测技术可防御侧信道攻击,随机化技术可防御代码重用攻击,增强隔离技术可防御侧信道和应用层的攻击,源码重构技术可防御侧信道和应用层的攻击。

3.1　异常检测

很多针对 SGX 的侧信道攻击需要不断地打断受害 enclave 的执行来提高时间分辨率,这些攻击导致的异常中断很容易被 T-SGX[16]和 Déjá Vu 系统[28]检测到。

Intel 处理器从 Haswell 系列开始引入了事务同步扩展(TSX)组件,它的一个关键特性是 TSX 中断会禁止向底层操作系统发出错误通知,这意味着操作系统无法知道在事务中是否发生了页面错误。T-SGX 基于一个修改过的 LLVM 编译器,可以自动地将 enclave 程序转换为安全版本,所有的代码和数据页都用 TSX 包装。同时,T-SGX 将回退处理程序和其他事务控制代码的特定页面(称为跳板)与原始程序的代码和数据页面隔离,以确保包括页面错误和计时器中断在内的异常只能在跳板上触发。操作系统虽然可以确定在跳板处是否发生了异常,但这不会显示任何有意义的信息,从而实现了异常的检测和隔离。T-SGX 对于典型的受控侧信道攻击,如 libjpeg、Hunspell 和 FreeType 等,具有较好的防御效果。但 T-SGX 不足以应对使用 LLC 的异步缓存定时攻击和页表与缓存的并发攻击。

Déjá Vu 系统实现了一种新的软件参考时钟,它

根据这个时钟对自己的执行步骤计时,以检测在其中一个步骤中是否发生了异常或中断。同时为了避免由于计算机上发生的正常中断和页面错误而导致的时钟异常,Déjá Vu 系统利用 Intel TSX 技术对该软件时间进行保护。通过设置合理的 AEXs 阈值,攻击者若中断或减缓 enclave 的执行,该受保护的时钟会检测出时间异常,而且目前绝大多数的侧信道攻击为了提高攻击的时空分辨率都会引起时间异常,因此该方案是检测和防御侧信道攻击的一个较有效的方法。

3.2 随机化技术

地址空间布局随机化(ASLR)[29-30]无论是在传统执行环境还是 SGX 可信环境中都是防御代码重用攻击的有效手段,但将 ASLR 应用于 SGX 程序却会带来新的挑战:(1) SGX 强大的攻击模型将 enclave 内存布局暴露给不可信的系统软件,使得 SGX 程序完全不受 ASLR 的保护;(2) SGX 只为飞地提供有限的内存,SGX 的 ASLR 不能充分利用虚拟地址空间,极大地限制了 ASLR 的随机性和安全性;(3) ASLR 需要动态重定位来实现代码和数据的相关地址,但这与 SGX 的认证过程相冲突,因为 SGX 在 enclave 执行开始之前就需要完成完整性度量,但 ASLR 的重定位需要在 enclave 执行之后进行;(4) SGX 对 enclave 中的某些安全关键数据使用了固定地址,且出于安全考虑,SGX 使一个飞地中的部分数据结构不可变,这使得攻击程序可以利用这些数据结构来绕过 ASLR。

SGX-Shield[17]将随机化技术应用在了 SGX 环境中,并克服了上述 SGX 与随机化“水土不服”的一些技术缺陷。SGX-Shield 引入了多级加载器,可以向攻击者隐藏 ASLR 的相关操作,同时 SGX-Shield 采用了一种细粒度的随机化方法并集成了粗粒度的软件故障隔离,可以克服 EPC 有限内存问题并保护固定位置且敏感的数据结构。为了解决 ASLR 需要动态重定位的问题,SGX-Shield 实现了一个软件数据执行保护来执行飞地代码页中的 W ⊕ X。SGX-Shield 可以有效地防御文献[21]的 Dark-ROP 攻击,但对文献[22]中的攻击无能为力。

文献[31]提出了一种称为语义无关的数据随机化防御方法,可以用于对抗基于缓存的针对 SGX 的侧信道攻击。该方法设计并实现了一个名为 DR.SGX 的基于编译器的工具,DR.SGX 采用 CPU 的加密硬件加速单元将置换计算为小域加密,可以以缓存行粒度安全地随机置换 enclave 数据的内存位置。同时为了防止重复内存访问的相关性,在 enclave 进程执行期间还会不断地重新随机化所有 enclave 数据。

3.3 增强隔离

侧信道攻击可以破坏 SGX 提供的数据机密性保证,可以利用异常检测(3.1 节)的方法来检测和隔离部分侧信道攻击,但却没有消除侧信道的攻击面。而增强隔离是在 SGX 为用户程序提供机密性和完整性保证的基础上,隔离 SGX 的整个攻击面来防御侧信道攻击。

Varys[32]通过限制 L1 和 L2 缓存等核心资源的共享为 SGX 飞地提供了一个侧信道保护的执行环境,该环境确保时间片或并发缓存计时以及页表攻击都不能成功。为了建立这样的增强隔离环境,Varys 实现了两种机制:(1) 可信内核保留,将物理内核严格保留给 enclave 线程,这样攻击者就不能在 enclave 线程运行时访问它们共享的内核资源,也不能在运行后从内核的 L1 和 L2 缓存中恢复任何的秘密,这可以有效地防止对内核共享资源的任何并发攻击,如分支预测器和浮点单元,还可以防止页表属性上的无退出 SCAs(因为它需要访问内核的 TLB);(2) 异步 enclave 退出监控,限制 AEX 退出的频率(该频率通常比无攻击执行中的退出频率要高得多但比所有已知攻击又要小,这样可以最小化误报的概率),一旦超出频率范围就终止 enclave 的执行,这可以防止更广泛的侧信道攻击,包括基于 LLC 的攻击。

Sanctum[18]的隔离机制专门用于防御针对 SGX 的软件攻击,它可防御已知的缓存定时攻击和被动地址转换攻击,主要解决了在相互不信任的应用程序之间共享一台计算机所产生的安全问题,但 Sanctum 方案需要对硬件进行微小的改动且无法防御 DoS 攻击和利用硬件错误的软件攻击(如 Rowhammer)。Sanctum 是一种联合设计,它将最小侵入性的硬件修改与受信任的软件安全监视器结合在一起,该软件安全监视器能够进行严格的分析,并且不使用密钥执行加密操作。

3.4 源码重构

以上所介绍的异常检测(3.1 节)、随机化技术(3.2 节)和增强隔离(3.3 节)虽然可以在某些方面防御针对 SGX 的攻击,但这些措施要么有较高的执行成本,要么对某些攻击变体无效,最大的不利之处就是增大了 SGX 的可信计算基(TCB)。本来 Intel 开发 SGX 的初衷就是要以硬件安全为强制性保障,为用户提供可信的执行环境。SGX 的可信计算基是仅包括硬件(CPU 和 enclave)的,但上述的防御措施却又破坏了这一点,使得 SGX 的可信计算基又包含了软件等。

而源码重构则主要是通过修改源码,小心地隐藏 enclave 程序的控制流和数据流,从而防御侧信道攻击。重构源码并未增大 SGX 的可信计算基,仍仅需信任硬件安全即可。如文献[33]使用 oblivious store 隐藏 if-else 控制流,使用 ORAM[34-35]隐藏数据流。文献

[36]使用硬件事务内存(HTM) 的事务原子性来确保与机密相关的控制流和数据访问都保留在 CPU 缓存中,若事务失败则会清除与事务相关的缓存,从而实现了控制流和数据流的隐藏。也可以在 enclave 程序执行过程中严格地审查地址映射防止敏感信息地址外泄、优化调度算法以防止不安全的共享、主动删除遗留在缓存中的敏感信息等。但源码重构相对复杂,需要对应用进行严密的分析与设计,很难在一个通用的计算环境中实现,且源码重构无法防御基于硬件漏洞的攻击。

4　发展方向

上述提及的针对 SGX 的攻击主要是单一的攻击方式,随着 SGX 攻击面的不断发掘,综合运用多种攻击方式的组合实现混合多层次攻击将会成为攻击 SGX 的新手段。混合多层次攻击最大的优势在于可以放大不同层次不同攻击方式的差异来提高攻击的准确度。如已有的 Cache 和 DRAM 混合攻击[37]可以将精度由 row(8 kB) 粒度提高到 Cache line(64 B) 。但已有的混合攻击重点关注于内存管理与地址转换方面,未来新的混合多层次攻击可以尝试更多类型攻击方式的组合或结合其他方面的信息,重点是 CPU 新特性,如 Intel TSX-NI(transactional synchronization extensions - new instructions) ,Intel MPX (memory protection extensions) ,Intel CAT(cache allocation technology) 等等。

未来针对 SGX 攻击的防御措施可以从 enclave 的加载执行时间方面进行综合分析。现有的异常检测主要是检测 enclave 的执行中断异常频次,但如何依靠终端的执行环境对 enclave 加载执行时间进行综合分析是检测 SGX 攻击的新思路,如利用人工智能技术训练 enclave 加载执行时间模型来进行异常检测。

5　结束语

Intel SGX 是一项有着广泛应用前景的安全技术。文中介绍了针对 SGX 的威胁模型,总结了针对 SGX 的攻击类型,探讨了目前针对 SGX 攻击的一些防御措施以及 SGX 攻击与防御可能的发展方向。

参考文献:

[1] INTEL. Intel software guard extensions programming reference [EB/OL].(2014) .https: //software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.

[2] DANEV B,JAYARAM M R,KARAMe G O,et al.Enabling secure VM-vTPM migration in private clouds [C]//27th annual computer security applications conference. Orlando: ACM,2011: 187-196.

[3] COSTAN V, DEVADAS S. Intel SGX explained. tech. rep., cryptology eprint archive: report 2016/086 [R/OL]. (2016-01-30) .https: // eprint.iacr.org/2016/086.pdf.

[4] MCKEEN F,ALEX I,BERENZON A,et al.Innovative instructions and software model for isolated execution [C]// 2nd international workshop on hardware and architectural support for security and privacy.Tel Aviv: ACM,2013.

[5] 王进文,江　勇,李　琦,等.SGX 技术应用研究综述[J]. 网络新媒体技术,2017,6(5) : 3-9.

[6] REINBRECHT C,SUSIN A,BOSSUET L,et al.Timing attack on NoC-based systems: prime+probe attack and NoC-based protection [J]. Microprocessors and Microsystems, 2017,52: 556-565.

[7] LIU F,GE Q,YAROM Y,et al.CATalyst: defeating last-level cache side channel attacks in cloud computing [C]// 22nd IEEE international symposium on high performance computer architecture (HPCA) .Barcelona: IEEE,2016: 406-418.

[8] MOGHIMI A, IRAZOQUI G, EISENBARTH T. Cache-Zoom: how SGX amplifies the power of cache attacks [C]// 19th international conference on cryptographic hardware and embedded systems.Taipei: Springer,2017: 69-90.

[9] BRASSER F,MÜLLER U,DMITRIENKO A,et al.Software grand exposure: SGX cache attacks are practical [C]//11th USENIX workshop on offensive technologies. Vancouver: USENIX,2017.

[10] XU Y,CUI W,PEINADO M.Controlled-channel attacks: deterministic side channels for untrusted operating systems [C]//36th IEEE symposium on security and privacy.San Jose: IEEE,2015: 640-656.

[11] BAUMANN A,PEINADO M,HUNT A G.Shielding applications from an untrusted cloud with Haven [C]//11th USENIX symposium on operating systems design and implementation.Broomfield: OSDI,2014: 267-283.

[12] HOFMANN O S,KIM S,DUNN A M,et al.InkTag: secure applications on an untrusted operating system [C]//18th international conference on architectural support for programming languages and operating systems.Houston: ACM,2013: 265-278.

[13] VAN BULCK J,WEICHBRODT N,KAPITZA R,et al.Telling your secrets without page faults: stealthy page table-based attacks on enclaved execution [C]//26th USENIX security symposium (USENIX security) . Vancouver: USENIX,2017: 1041-1056.

[14] LEE S,SHIH M W,GERA P,et al.Inferring fine-grained control flow inside SGX enclaves with branch shadowing [C]//26th USENIX security symposium (USENIX security) .Vancouver: USENIX,2017: 557-574.

[15] SHINDE S,CHUA Z L,NARAYANAN V,et al.Preventing page faults from telling your secrets [C]//11th ACM on Asia conference on computer and communications security.Xi'

an: ACM, 2016: 317-328.
[16] SHIH M W, LEE S, KIM T, et al. T-SGX: eradicating controlled-channel attacks against enclave programs [C]//2017 network and distributed system security symposium. San Diego: NDSS, 2017.
[17] SEO J, LEE B, KIM S, et al. SGX-shield: enabling address space layout randomization for sgx programs [C]//2017 network and distributed system security symposium. San Diego: NDSS, 2017.
[18] COSTAN V, LEBEDEV I, DEVADAS S. Sanctum: minimal hardware extensions for strong software isolation [C]//25th USENIX security symposium. Austin: USENIX, 2016: 857-874.
[19] 王 鹃, 樊成阳, 程越强, 等. SGX 技术的分析和研究 [J]. 软件学报, 2018, 29(9): 2778-2798.
[20] SCHWARZ M, WEISER S, GRUSS D, et al. Malware guard extension: using SGX to conceal cache attacks [C]//14th international conference on detection of intrusions and malware, and vulnerability assessment. Bonn: Springer, 2017: 3-24.
[21] LEE J, JANG J, JANG Y, et al. Hacking in darkness: return-oriented programming against secure enclaves [C]//26th USENIX security symposium. Vancouver: USENIX, 2017: 523-539.
[22] ANDREA B, MAURO C, LUCAS D, et al. The guard's dilemma: efficient code-reuse attacks against intel SGX [C]//27th USENIX security symposium. Baltimore: USENIX, 2018: 1213-1227.
[23] INTEL. L1 terminal fault [EB/OL]. (2018-08-14). https://software.intel.com/security-software-guidance/software-guidance/l1-terminal-fault.
[24] BULCK J V, MINKIN M, WEISSE O, et al. Foreshadow: extracting the keys to the Intel SGX kingdom with transient out-of-order execution [C]//27th USENIX security symposium. Baltimore: USENIX, 2018: 991-1008.
[25] WEISSE O, BULCK J V, MINKIN M, et al. Foreshadow-NG: breaking the virtual memory abstraction with transient out-of-order execution [EB/OL]. (2018-08-14). https://foreshadowattack.eu.
[26] JANG Y J, LEE J, LEE S, et al. SGX-bomb: locking down the processor via rowhammer attack [C]//SysTEX'17: 2nd workshop on system software for trusted execution. Shanghai: ACM, 2017.
[27] KIM Y, DALY R, KIM J, et al. Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors [C]//41st annual international symposium on computer architecture. Minneapolis: IEEE, 2014: 361-372.
[28] CHEN S, ZHANG X K, REITER M K, et al. Detecting privileged side-channel attacks in shielded execution with Déjá Vu [C]//2017 ACM on Asia conference on computer and communications security. Abu Dhabi: ACM, 2017: 7-18.
[29] GIUFFRIDA C, KUIJSTEN A, TANENBAUM A S, et al. Enhanced operating system security through efficient and fine-grained address space randomization [C]//21st USENIX security symposium. Bellevue: USENIX, 2012: 475-490.
[30] LU K J, Nürnberger S, BACKES M, et al. How to make aslr win the clone wars: runtime re-randomization [C]//2016 network and distributed system security symposium. San Diego: NDSS, 2016.
[31] BRASSER F, CAPKUN S, DMITRIENKO A, et al. DR. SGX: automated and adjustable side-channel protection for SGX using data location randomization [C]//35th annual computer security applications conference. San Juan Puerto Rico: ACSAC, 2019: 788-800.
[32] OLESENKO O, TRACH B, KRAHN R, et al. Varys: protecting SGX enclaves from practical side-channel attacks [C]//2018 USENIX annual technical conference. Boston: USENIX, 2018: 227-240.
[33] RANE A, LIN C, TIWARI M. Raccoon: closing digital side-channels through obfuscated execution [C]//24th USENIX conference on security symposium. Washington: USENIX, 2015: 431-446.
[34] DEVADAS S, DIJK M V, FLETCHER C W. Onion ORAM: a constant bandwidth blowup oblivious RAM [C]//14th theory of cryptography conference. Beijing: Springer, 2016: 145-174.
[35] STEFANOV E, VAN D M, SHI E, et al. Path ORAM: an extremely simple oblivious RAM protocol [C]//2013 ACM SIGSAC conference on computer and communications security. Berlin: ACM, 2013: 299-310.
[36] GRUSS D, LETTNER J, SCHUSTER F, et al. Strong and efficient cache side-channel protection using hardware transactional memory [C]//26th USENIX security symposium. Vancouver: USENIX, 2017: 217-233.
[37] WANG W H, CHEN G X, PAN X R, et al. Leaky cauldron on the dark land: understanding memory side-channel hazards in SGX [C]//2017 ACM SIGSAC conference on computer and communications security. Dallas: ACM, 2017: 2421-2434.