基于 enclave 的无服务器计算综述报告

李彬弘

摘 要 在目前的云计算发展下，无服务器计算已经成为现实，并逐渐成为部分应用程序和服务的一种新范式。在无服务器计算框架中，云用户可以在服务运行时执行任意函数，而无需关注部署、管理、可伸缩性等问题。然而在复杂的云环境中，恶意的云用户可能利用容器或 OS 内核的漏洞，危害平台的主机运行时，甚至底层系统。而另一方面，无服务器计算框架与其他云计算模型有相同的问题，即恶意或好奇的云提供商可能窃取云用户的隐私敏感数据。因此，无服务器计算的安全与隐私保护问题逐渐成为重大挑战。而随着 Intel SGX 的出现，SGX 可以提供与系统其余部分完全隔离的安全 enclave，同时允许远程用户验证 enclave 的身份，TEE 被认为是实现无服务器计算安全性和隐私保护的有前途的技术。本文综述了 3 篇基于 enclave 的无服务器计算的论文，并阐述他们的原理以及研究进展。

关键词 无服务器计算；SGX enclaves；隐私保护；安全性

A survey of enclave-based serverless computing

Binhong Li

Abstract Under the current development of cloud computing, serverless computing has become a reality and gradually becomes a new paradigm for some applications and services. In a serverless computing framework, cloud users can execute any function while the service is running without worrying about deployment, administration, scalability, and so on. However, in a complex cloud environment, malicious cloud users may exploit vulnerabilities of containers or OS kernel to harm the platform's host runtime or even the underlying system. Serverless computing frameworks, on the other hand, have the same problem as other cloud computing models, namely the potential for malicious or curious cloud providers to steal cloud users' privacy-sensitive data. Therefore, the security and privacy protection of serverless computing is becoming a major challenge. With the advent of Intel SGX, which can provide a secure enclave completely isolated from the rest of the system and allow remote users to verify the identity of the enclave, TEE is considered a promising technology to implement serverless computing security and privacy protection. In this paper, three papers on serverless computing based on enclave are reviewed, and their principles and research progress are described.

Key words serverless computing; SGX enclaves; Privacy protection; security

1 引言

随着云计算快速发展，无服务器计算已经成为部署应用和服务的新范式。无服务器计算，也称为功能即服务（Function-as-a-Service，FaaS），为云提供商和开发人员提供了极大的便利。开发人员在使用无服务器计算时，只需要指定要执行的函数，而不再需要请求特定数量的虚拟机，或者安装和维护完整的软件堆栈，这些函数所需的基础设施都将由服务提供者来完成，从而开发人员能够通过编写细粒度的、简单的和独立的函数来专注于业务逻辑。而云提供商现在可以优化底层的基础设施，使吞吐量和性能达到最大，提高了云提供商的效率。如今已经有了许多无服务器计算的服务提供商，如 Amazon AWS Lambda、Microsoft Azure Functions、Google cloud Functions。

无服务器计算作为云计算的一种，必然会涉及到敏感数据的处理，例如医疗应用程序中的患者数据或者政府、警察当局的 IT 服务、以及一些用于隐私敏感的应用程序（如 Auth0(身份验证)，Alexa 聊天机器人 (用户意图)，人脸识别 (生物信息) 等）。同样地，无服务器计算框架和其他云计算模型有相同的问题，即恶意或好奇的云提供商可能窃取云用户的隐私敏感数据。因此，在复杂的云环境中，保护用户隐私，不受恶意的租户或可疑的内部人员的侵害是非常必要的。

为了实现这种隐私保护，以往的办法要么使用同态加密，要么使用可信执行环境（TEE）。同态加密是基于数学难题的计算复杂性理论的密码学技术。信息传输方对经过同态加密的数据进行处理得

收稿日期：2022-01-06；修改日期：2022-01-06

到一个输出，信息接收方将这一输出进行解密，获得结果。同态加密虽然保障了数据的安全性，但若是对所有数据进行加密计算，必将导致昂贵的计算成本，因此使用同态加密来保护云中的数据是不现实的。而可信执行环境，例如 Intel SGX，可以提供与系统其余部分完全隔离的安全 enclave，同时允许远程用户验证 enclave 的身份。因此，TEE 被认为是实现无服务器隐私保护的一种有前途的技术。

本文综述了三篇基于 enclave 提高无服务器计算的安全性的论文，论文 1—Confidential Serverless Made Efficient with Plug-In Enclaves—观察到将无服务器计算直接移植到 SGX 会导致显著的性能下降，从 5.6× 到 422.6× 不等，而性能下降的根本原因在于当前的 SGX 设计禁用了 enclave 实例之间的内存共享。在这个基础上，论文 1 提出了一种新的、灵活的 enclave 模型，称为 PIE，提高了无服务器计算的效率和实用性。论文 2–Trust More, Serverless(T-FaaS)–基于 Intel SGX 以解决安全性与隐私泄漏的问题，设计了可信执行安全 FaaS 平台，该平台既保留了 FaaS 的优点，同时又尊重 Intel SGX 提供的可信执行的特点。论文 3—Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave—主要关注无服务器计算中的安全性问题，提出了一种新的无服务器计算框架 Se-Lambda，它通过使用 SGX enclave 来保护 API 网关，并利用 SGX enclave 和 WebAssembly 沙箱环境结合的双向沙箱来保护服务运行时。在服务运行时，用户的不可信代码受到 WebAssembly 沙箱环境的限制，而 SGX enclave 防止恶意云提供商窃取用户的隐私敏感数据，通过较少的性能损失，换来了显著的安全性能的提升。

2 原理和优势

2.1 Intel Software Guard Extension (SGX)

Intel Software Guard Extensions(SGX) 是 Intel 架构的扩展，用于增强软件安全性。使用 SGX，开发人员可以将软件的可信部分放置在 enclave 中，以保护选定的代码和数据不被泄露或修改，因为 SGX CPU 将 enclave 与一系列威胁隔离开来，包括

(1) 外部设备的直接内存访问 (DMA)

(2) 特权系统软件，如固件、管理程序和操作系统

(3) 与 enclave 在同一地址空间的应用程序

(4) 共享同一托管应用程序的 enclave

在 SGX 的上下文中，enclave 是独立的执行单元，具有加密的代码和数据。Enclave 内存称为 Enclave Page Cache (EPC)。数据被放置在 Enclave Page Cache (EPC) 中，EPC 中的内存由内存加密引擎 (MEE) 加密，以防止已知的内存攻击。EPC 中的内存内容只有在进入 CPU 包时才会被解密，在离开 CPU 包时才会被加密。EPCs 是从 DRAM 的一个物理连续区域分配的，该区域称为处理器预留内存 (PRM)。SGX 对 EPC 有严格的访问控制模型: 一个 EPC 只属于一个 enclave 实例。每个 enclave 实例都有一个唯一的 enclave 标识符 (EID)，存储在其 SGX enclave 控制结构 (SECS) 中。当一个 EPC 页面添加到一个飞地时，SGX CPU 将这个 EPC 页面与一个名为 EPC Map (EPCM) 的元数据关联起来。EPCM 条目指示 EPC 页面的页面类型、权限、虚拟地址 (VA) 以及所有者 EID，如图 1 所示。当一个正在执行的 enclave 试图访问一个特定的 EPC 页面时，CPU 将检查它的 SECS。EID 将匹配相应的 EPCM 与此 EPC 页的 EID 匹配。任何软件都无法访问 SECS 和 EPCM 等 Enclave 元数据。

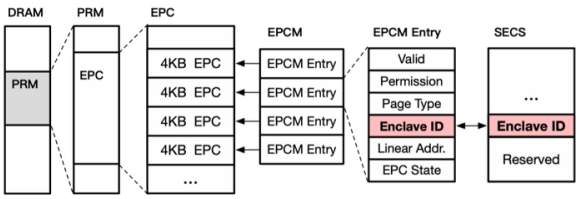


图 1 SGX 内存访问控制模型

2.2 Lambdas

FaaS 可以被视为早期云计算方法 (如 IaaS 和 PaaS) 的继承者。但 FaaS 范式仅处理单个独立的函数。在本文中，我们将其称为 Lambdas，其灵感来自于 Amazon 的第一个 FaaS 平台。

Lambdas 是天生的小型无状态函数，通常用 JavaScript 或 Python[14] 这样的解释语言编写。它们由 FaaS 用户或客户提供，并在独立的环境中执行，例如云提供商的机器 [27] 上的 Docker 容器。

大型 Lambda 应用程序由多个 Lambdas 组成,通过互连这些可以使用公共数据库或其他形式的持久存储的 Lambdas，形成一个整体应用程序。Lambdas 可以直接调用或由事件触发。对于 Lambdas 来说，最重要的是它们能够快速启动，并且能够根据当前的需求自动跨多台机器扩展。

2.3 WebAssembly

WebAssembly（WASM）是一种实验性的、低层次的编程语言，通常应用于 web，它定义了一种适合于 web 编译的小型、可移植和负载时间效率高的格式。到目前为止，Chrome、Microsoft Edge、Safari 和 Firefox 等浏览器已经开始支持初始版本的 WebAssembly。

WebAssembly 利用软件故障隔离 (SFI) 来构建与主机运行时隔离的沙箱环境。沙箱环境用于防止漏洞或恶意模块泄漏用户数据。每个 WebAssembly 模块都受其嵌入的安全策略的约束。例如，运行在 web 浏览器中的模块遵循与浏览器相同的源策略，而运行在非浏览器环境中的模块遵循与系统相同的 POSIX 安全策略。WebAssembly 模块必须在加载时声明所有可访问的方法及其相关类型，包括它使用的动态链接函数。因此，我们可以隐式地利用控制流完整性 (CFI) 来检查应用程序是否在程序执行期间被劫持。

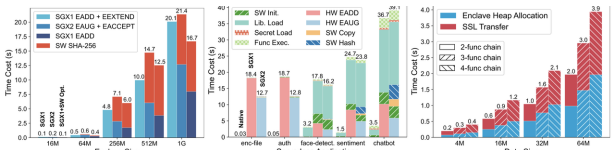
WebAssembly 中的函数调用方式与 C/ c++ 中的函数调用有以下几个方面的不同:(1) 函数调用必须指定对应于函数索引或表索引空间中有效条目的目标索引。(2) 间接函数调用需要在调用时指定目标函数的类型签名，那么在调用函数之前必须精确匹配类型签名。通过上述机制，WebAssembly 可以有效地避免控制流劫持攻击。

3 研究进展

基于 enclave 的无服务器计算隐私保护近年来的研究较多,有通过研究 SGX enclave 和 WebAssembly 沙箱环境结合的双向沙箱以进一步加强安全性的, 如 Se-Lambda, 有基于 enclave 完善 FaaS 平台设计应用的, 如 T-FaaS, 也有注重于通过改善 enclave 模型以提高效率的, 如论文 1 Confidential Serverless Made Efficient with Plug-In Enclaves。

3.1 Plug-In Enclaves（PIE)

PIE 对使用 SGX enclave 来保护无服务器功能的性能方面进行评估，发现了显著的性能下降，并经过定量调查，观察到大部分开销来自于 enclave 的初始化（即: 硬件 enclave 的创建和认证度量的生成）与函数间的数据传输。得出效率低下的根本原因在于当前的 SGX 设计禁用了 enclave 之间的内存共享。这种无共享的设计提供了强大的安全保障，但导致了严重的启动延迟。



(a) Enclave instance startup time breakdown. (b) Startup breakdown of enclave functions. (c) Data transfer cost between enclaves.

图 2　SGX 区域测量无服务器计算

基于此，论文提出了一种新的 enclave 模型，即 Plug-In Enclaves （PIE）。PIE 提出了一种新的硬件内存原语：共享 enclave 区域，它能够不可变地映射到不同的隔离 enclave 以实现安全共享。为了在提升效率的同时，依然注重隐私保护，PIE 利用这个硬件原语，创建了两种类型的逻辑 enclave:

（1）插件 enclave：插件 enclave 完全由共享 enclave 区域组成，并且可以容纳不敏感的公共环境，如语言运行时 (如 Python)、框架 (如 Tensorflow)、第三方库 (如 OpenSSL) 和初始状态 (如机器学习模型)

（2）主机 enclave：运行一个安全的沙箱，处理用户的数据 (通常从一个密码通道)，并小心翼翼地保护其加工过程和最终的结果。

在 PIE 设计中, 主机 enclave 之间是严格隔离的, 但是可以将插件 enclave 映射到它们自己的 enclave 地址空间中，以便重用插件 enclave 加载的内容以及它们容易生成的度量值。主机 enclave 可以进一步重新映射插件 enclave 以适应不同的应用逻辑, 而无需迁移其秘密数据。通过这种方式，既保护了隐私数据，同时也提高了性能。为了实现这种设计, PIE 对 SGX 的架构进行了扩展，包括一个新的页面类型 PT-SREG 用于表示共享 enclave 内存，以及两个新的指令 EMAP 和 EUNMAP 来映射/取消一个插件 enclave 到另一个主机 enclave 的映射。同时为了确保插件 enclave 的内容和度量之间的一致性, PIE 重用 SGX2 动态调整大小来实现硬件强制的写时拷贝机制。为了评估 PIE 的性能，论文将以下三种情况进行比较:

(1) 基于 SGX 的冷启动: 一个软件优化的环境 (使用基于软件的测量和基于模板的技术)，每个 enclave 都是根据新的请求创建的。

(2) 基于 SGX 的暖启动: 一个“智能”环境，它预暖大量的 enclaves，准备在一个容量内服务并发请求; 出于隐私原因，软件重置必须在两次调用之间执行。

(3) 基于 PIE 的冷启动: 一个 PIE 优化的环境, 在这个环境中，预先创建了许多插件 enclave，但

是用于无服务器功能的主机 enclave 是按需创建的，类似于（1）。

实验根据功能启动 (§VI-A)、自动伸缩 (§VI-B) 和功能链接 (§VI-C) 来评估，实验结果如下图所示。得益于 PIE，enclaves 可减少启动延迟 94.74-99.57

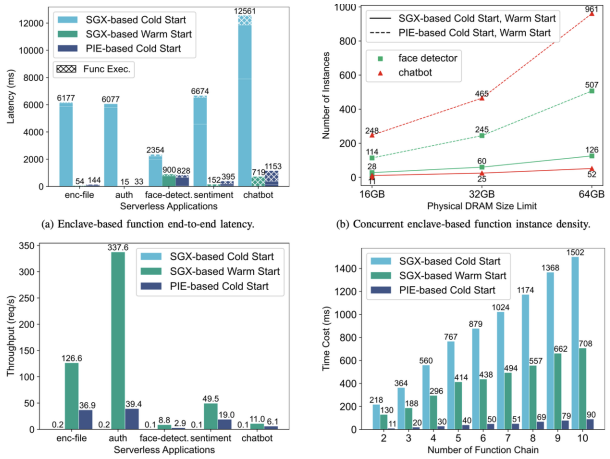


图 3 基于 PIE 的冷启动、基于 SGX 的暖启动性能、基于 SGX 的冷启动性能比较

3.2 Se-Lambda

Se-Lambda 通过引入双向沙箱和利用硬件隔离核心模块，防止 API 网关上的不可信软件对核心模块进行破坏，并在云用户的功能模块和云提供商之间提供双向保护，防止服务运行时泄漏云用户的数据，防止恶意代码损害云提供商的主机运行时。当在受保护的服务运行时上部署无服务器应用程序时，云用户和云提供商都不需要担心信任问题。参考 Amazon 的 AWS Lambda 体系结构，图 4 显示了 SeLambda 体系结构的详细概述，它包含一个 API 网关与一个服务运行时。

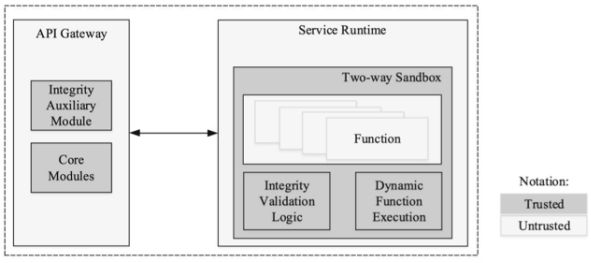


图 4 Se-Lambda 系统架构

3.2.1 API 网关

API 网关是 Se-Lambda 的关键组件，它包含了各种核心模块，如认证和授权。图 2 显示了 API 网关处理用户请求的工作流，其中简要包括以下步骤。首先，API 网关通过解析请求信息和执行安全检查来处理用户请求，以防止恶意攻击。然后，当验证请求时，API 网关调用服务中介中的用户身份验证和访问控制模块。最后，在认证成功后，API 网关调用相应的服务运行时执行功能模块，并将结果输出给用户。Se-Lambda 不信任 API 网关上的其他软件，包括操作系统和管理程序。将 API 网关的所有模块放置到 SGX enclave 将导致一个巨大的 TCB。相反，只将对隐私敏感的核心模块放置在 SGX enclave 中，以隔离它们。由于 Se-Lambda 只保护 API 网关的一部分，所以可以提供细粒度的隔离和小 TCB。

图 5 显示了 API 网关处理用户请求的工作流。受保护服务运行时创建双向沙箱后，沙箱向 API 网关验证其是否受信任，并通过启动远程认证建立经过身份验证的通信通道。当 API 网关调用受保护的服务运行时执行一个函数模块时，它通过上述经过身份验证的通道将函数模块的哈希值 (来自用户请求) 发送到双向沙箱。然后，在执行该功能模块之前，双向沙箱利用哈希检查来验证该功能模块的完整性。

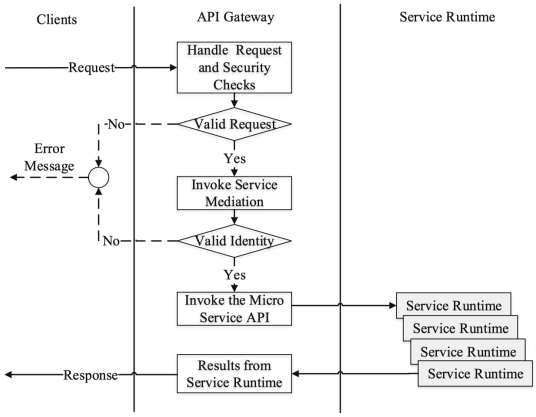


图 5 API 网关处理用户请求

3.2.2 双向沙箱

服务运行时是无服务器计算框架中执行用户功能模块的引擎。然而，无论是云用户还是云提供商都不值得信任，因为恶意用户的代码可能会危害主机运行时，恶意云提供商可能会窃取用户的数据。一些解决方案提供可信环境来解决这些问题，这并不适合无服务器编程模型。在本节中，我们提供了一个受保护的服务运行时，其中包括一个双向沙箱。

图 6 显示了双向沙盒的体系结构，其中包括灰色部分为可信，白色部分为不可信。可信部分包含 CPU、SGX 驱动程序和一个双向沙箱，它是一个包含 WebAssembly 沙箱环境的 SGX enclave。沙箱提

供的双向保护是 SGX enclave 保护用户数据的机密性, 而 WebAssembly 沙箱环境保护云提供商的主机运行时的安全性。

但是, 代码内部 SGX enclave 不能执行特权指令, 需要切换到外部世界来执行, 这可能面临恶意攻击, 如 Iago 攻击中恶意内核可以操纵系统调用的返回值, 然后诱导过程采取行动对其利益的保护。此外, 虽然 WebAssembly 沙箱环境可以防止软件漏洞损害主机运行时，但它不能阻止超出权限的执行。

双向沙箱的安全功能之一是管理功能模块与外界的接口。因此，双向沙箱在 SGX enclave 中包含一个权限监控模块，用于管理功能模块的访问控制，该模块对功能模块的系统调用进行编码，并根据权限集对其进行过滤。为了防止 Iago 攻击，我们采用了让权限监控模块对系统调用的返回值进行检查的方法。

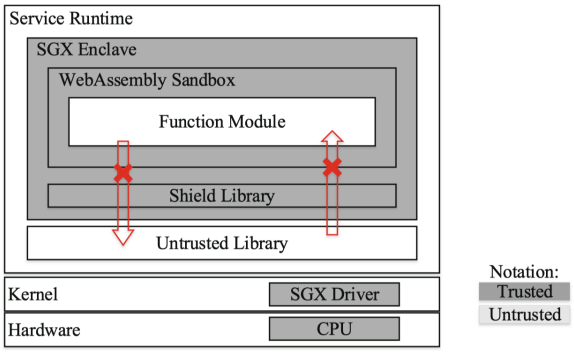


图 6　双向沙盒体系结构

3.2.3　性能评估

Se-Lambda 旨在为云用户提供一个轻量级、安全的无服务器框架。因此 Se-Lambda 的评价包括三个部分: (1) 安全性评估; (2) 将基于 SGX enclave 的 API 网关与本地实验性能对比；(3) 对 Se-Lambda 的服务运行时进行了一个实证实验，以评估双向沙箱对性能的影响

（1）安全性评估

Se-Lambda 针对服务运行时和 API 网关内部的双向沙箱模拟了一些攻击事件，以验证其安全性保证，实验结果如下图所示。Se-Lambda 可以防御列出的所有攻击，同时在保密性方面，Se-Lambda 可以保护用户的敏感数据，最后，Se-Lambda 可以防止不受信任的代码损害主机运行时。

（2）API 网关性能

Se-Lambda 评估了 API 网关在 SGX 启用和禁用情况下的 CPU 利用率。基准是运行不带 SGX 的

Component	Attack event	Security threat	Defense ability
Two-way sandbox	Executing untrusted code on sandbox	Confidentiality	Yes
	Stealing the sensitive data of users	Confidentiality	Yes
	Executing arbitrary system calls	Confidentiality	Yes
	Tampering with users' code	Integrity	Yes
	Tampering with the integrity validation	Integrity	Yes
API gateway	Stealing isolated modules' sensitive data	Confidentiality	Yes
	Tampering with the isolated module	Integrity	Yes

图 7　Se-Lambda 安全性评估

API 网关，也叫 gateway，带 SGX 的 API 网关称为 gateway -SGX。如图所示，Gateway-SGX 每秒可以实现大约 11,600 个请求，而当吞吐量超过这个值时，延迟会显著增加。CPU 利用率随着吞吐量的增加而增加。Gateway 和 Gateway- sgx 的 CPU 利用率峰值约为 125，这是因为 API 网关是一个多线程程序，不同的线程运行在多个 CPU 核上。这说明 SGX 施加的开销是可以接受的。

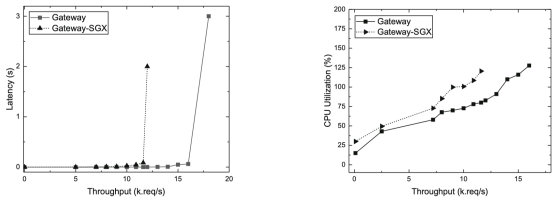
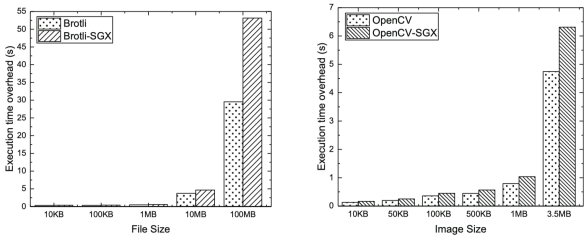


图 8　API 网关在 SGX 启用和禁用时性能对比

（3）Se-Lambda 的服务运行时实证实验

实验选择谷歌 brotli[6] 和 OpenCV[20] 作为评价基准。此外，实验量化了双向沙箱的内存占用和时间开销。如图所示，执行时间开销随着测试文件的大小而增加。实验发现，小工作负载的大部分时间都花在创建飞地上，而大工作负载的飞地创建开销则不那么重要。在实验中，我们选取五个不同的阶段来评估双向沙盒的内存占用，发现双向沙盒的内存占用在创建飞地之后逐渐增加。但在双向沙盒的生命周期中，最大的内存开销花费在函数的执行上，这是合理的，因为内存是在函数模块执行时动态分配的。



(a) Brotli　　(b) OpenCV

图 9　Brotli, OpenCV 的执行时间

3.3 T-FaaS

FaaS 云平台和 LaaS、PaaS 一样，安全性都是关键因素。因此 T-FaaS 提出在云内部采用可信执行环境（TEE），从而实现安全性问题。但如果只是单纯地将 FaaS 平台移植到 SGX 并不是一个有效的解决办法，因为这将会导致内存使用低效的问题。而 Intel SGX 的可信内存有限，无法适应大量的 Lambdas，这也就导致了性能问题。

因此 T-FaaS 提出了一个基于 JavaScript 的安全 FaaS 平台的通用方法，即在同一个 enclave 内的单个 JavaScript 引擎上运行多个纯基于 JavaScript 的 Lambdas，从而在多个来自不同的 Lambda 提供商的 Lambdas 之间共享相对较大的解释器。以谷歌的 V8 JavaScript 引擎为例，它是基于相互不信任的 JavaScript 组件思想而编写的，通过 V8 隔离来隔离彼此。

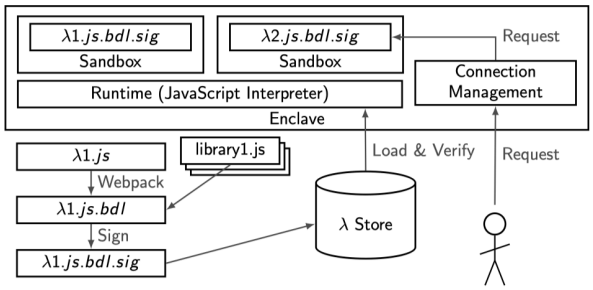


图 10 T-FaaS 平台架构

T-FaaS 构建的安全 FaaS 平台的通用架构如图所示，其原则上独立于所使用的 JavaScript 解释器。尽管如此，解释器仍然是平台的核心。解释器在 enclave 内运行一次，并在 enclave 内执行的所有 Lambdas 之间共享。所有的 Lambdas 都在它们自己的上下文中执行，以便能够并行执行 Lambdas 并在它们之间提供合理的隔离。相同的 Lambda 脚本在不同的上下文中多次执行，以便在高负载情况下通过并行执行来提高性能。通常，Lambdas 以 Lambda JavaScript 代码的签名包的形式存储在 enclave 之外。平台根据需要将此包从不受信任的存储加载到新创建的上下文中，并为用户调用做好准备。在加载时，Lambda bundle 的签名由平台进行验证，以确保只有经有效客户正确签名的 Lambdas 在平台上执行。

在加载并验证 Lambda 之后，将创建一个新上下文，以便使用 JavaScript 解释器执行它。这确保 Lambda 是独立执行的，与其他 Lambdas 隔离，并且可以与其他 Lambdas 并行运行。在单个 Lambda 的高负载情况下，为同一个 Lambda 实例化多个上下文也是有益的，在这种情况下，Lambda 只从外部加载一次，但会从它创建多个独立的上下文。当用户对这个 Lambda 的请求到达，而 Lambda 还未出现在飞地时。Lambda 只按需加载。新连接还需要创建一个连接上下文，用于在其他连接中存储特定于连接的数据，比如 TLS 会话密钥。这样的连接上下文没有绑定到特定的 Lambda, Lambda 上下文独立于用户连接。

通过利用最新的 JavaScript 引擎的隔离功能，T-FaaS 实现了功能级隔离；功能快速冷启动；有效使用可用资源等 FaaS 平台的优点，同时利用 enclaves 的可信执行特点增强了安全性。T-FaaS 通过基于 Duktape 和谷歌 V8 JavaScript 引擎的两个安全 Lambda 平台 SecureDuk 和 SecureV8 的评估结果进行验证。评估内容包括两个平台的 TCB、它们的吞吐量和响应时间，以及 enclaves 的工作集内存占用。评估表明，轻量级 SecureDuk 平台，提供了一个相对较低的 TCB，而 SecureV8 平台具有更高的性能。

4 总结与展望

基于 enclave 对无服务器计算进行改进，PIE 提出了新的 enclave 模型，扩展了插件 enclave，从而可以重用非机密的重量级状态，并通过重新映射 enclave 函数的方法来消除数据传输瓶颈，保障了安全性的同时提高了无服务器计算性能。而 Se-Lambda 更重视安全性的问题，提出了利用 SGX enclave 和 WebAssembly 沙箱环境结合的双向沙箱来保护服务运行时，通过这种办法云用户可以放心处理他们的隐私数据，而云服务平台也不必担心恶意用户代码会破坏主机运行时。T-FaaS 则是将 JavaScript 引擎移植到 SGX，构建一个安全的无服务器平台。

无服务器计算是云计算的未来趋势之一。尽管上面几篇文章能够解决一部分问题，诸如安全性、性能、隐私保护问题等。但无服务器计算仍存在许多亟需解决的问题，如监控、日志记录、开发和调试工具还没有或不成熟；供应商锁定等问题。而将多种问题的解决方案整合在一起形成一个连贯的解决方案更是一个值得关注的问题。

参考文献

[1] BRENNER S, KAPITZA R. Trust more, serverless[C/OL]//SYSTOR '19: Proceedings of the 12th ACM International Conference on Systems and Storage. New York, NY, USA: Association for Computing Machinery, 2019: 33–43. https://doi.org/10.1145/3319647.3325825.

[2] QIANG W, DONG Z, JIN H. Se-lambda: Securing privacy-sensitive serverless applications using sgx enclave[C]//International Conference on Security and Privacy in Communication Systems. [S.l.]: Springer, 2018: 451-470.

[3] LI M, XIA Y, CHEN H. Confidential serverless made efficient with plug-in enclaves[C]//2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). [S.l.]: IEEE, 2021: 306-318.