秘密排队：基于模型的远程计时攻击

Simone Bozzolan[1],Diletta Olliaro[2],Stefano Calzavara[1],Andrea Marin[1] *IEEE* 高级会员，Gianfranco Balbo[3],Matteo Sereno[3] *IEEE* 高级会员[1] 威尼斯大学，威尼斯，意大利[2] 瑞典皇家理工学院，斯德哥尔摩，瑞典[3] 都灵大学，都灵，意大利

arXiv:2611.30513v1 [cs.PF] 23 Nov 2026

摘要—**远程计时攻击利用响应时间的变化来推断有关远程系统的敏感信息。然而，由于难以准确建模服务器端响应时间的随机性，同时考虑请求服务时间的内在变化性以及不同工作负载对观测响应时间的影响，这类攻击的有效性往往受到限制。本文提出了一种基于排队模型和贝叶斯统计的远程计时攻击新型随机框架。我们假设攻击者知道两类请求的服务时间分布，并且当恶意观察者在外生工作负载所引入的噪声存在时，能够根据观测响应时间以高概率区分请求类别，就会发生信息泄露。**

我们在热门网站上进行的实验评估证明了该方法具有实际可行性：我们的攻击优于基于确定性服务时间的形式化技术和现有经验技术；与以往的形式化方法相比，该方法所需的调优更少，超参数也更少。总体而言，我们的结果表明，有原则的随机建模能够为在现实网络和工作负载条件下构建更有效的远程计时攻击提供坚实基础。

索引词—**网络隐私、排队论、远程计时攻击**

I. 引言

定时攻击利用时序信息来推断有关目标系统的私人信息。在互联网中实施的此类攻击通常称为远程计时攻击，已被证明会造成重大的实际影响 [1]。例如，~~Brum~~ley 和 Boneh 表明，攻击者能够测量远程网络服务器响应时间的变化，从而重构其 RSA（Rivest-Shamir-Adleman [2]）私钥 [3]。在隐私场景中，Felten 和 Schneider 使用基于缓存的计时攻击来追踪网络用户 [4]。其基本原理是：用户访问静态页面后，浏览器缓存会使该页面在之后再次访问时加载得更快。后续研究利用远程计时攻击泄露有关目标用户的敏感信息，例如活跃账户是否存在 [5]以及受害者的身份验证状态 [6]。这类研究十分重要，因为它们证明了远程计时攻击在实践中切实可行；否则，这类攻击可能显得不切实际。然而，先前的研究主要基于简单的经验观察，缺乏充分的科学依据。从总体上看，以往的攻击都依赖相同的直觉：攻击者收集响应时间，并运用某种形式的统计分析来判断这些时间是否泄露了目标系统的信息。各种方案的主要区别在于攻击目标、数据收集的细节以及统计分析的具体方式，后者从简单地目视检查直方图，到使用 t 检验等统计检验不等。

受排队论丰富研究脉络的启发 [7]–[8], ，本文提出了一种基于随机建模的远程计时攻击新方法。排队论是应用数学中一个广受欢迎的分支，被广泛用于性能工程任务。它借助随机模型来预测系统的定量行为，例如响应时间。乍看之下，为了实现数值上的可处理性，排队模型似乎依赖于严格且不切实际的假设；但工程实践经验已证明了其非凡的稳健性，使从业者能够将其应用于诸多目的 [9]。给定网络请求的类型（例如支付请求），排队模型可以预测该请求的预期响应时间及其他定量指标。在此，我们提出一种将贝叶斯统计与排队建模相结合的新方法，以解决这一逆问题：根据网络请求观测到的响应时间，确定其可能的类型。

这一方法为远程计时攻击提供了一个潜在的新基础：它概述了一个通用框架，可根据时序信息帮助区分不同类型的请求。值得注意的是，这一视角不仅具有理论意义，也带来了实际优势。我们的方法似乎天然适合应对各种复杂因素

这些因素会使性能评估变得困难，例如负载因子对目标系统响应时间的影响——有时，这些因素会限制纯实证攻击的稳健性。

此外，我们的攻击会针对预测的请求类型给出定量概率估计，从而正式保证攻击准确率的估计结果，并利用核心思想的通用性，使其能够统一应用于不同场景。即使仅从实验角度来看，我们的攻击也优于文献中提出的方法，表现胜过 BakingTimer 攻击 [6]。

我 得出结论，文中提出的形式化基础 们在此

本文已提交 IEEE，拟供其考虑发表。版权可能在不另行通知的情况下转让，届时本版本可能不再可访问。

可以成功应用于设计有原则的远程计时攻击。这类攻击在实践中更为有效，因为它们能更好地应对所要解决问题的复杂性。

贡献

本文的贡献如下：

1）我们提出了一种新的计时攻击策略，将排队建模与贝叶斯统计相结合，以改进文献中提出的现有经验方法（第三节）。通过利用所引入随机模型所支持的定量分析，我们的方法旨在解决当前方法的关键局限，为定时攻击提供一个更稳健的框架（第四节）。

2）我们提出了一种具体实现方案，基于 M/M/1处理器共享（M/M/1/PS）排队系统 [10], [11]，为我们的计时攻击提供支持。我们使用标准数学工具开展定量分析，旨在支持在实践中实施有效的远程计时攻击（第五节）。

3）我们实现了这种新的计时攻击，并将其效果与当前最先进方法进行比较，探讨采用我们方案所带来的主要优势（第六节）。

为支持研究的可复现性并全面遵守开放科学政策，我们公开实现代码和收集的数据以供审查 [12]。

相较会议版本的扩展

本文对我们在初步会议版本 [13] 中提出的模型进行了大幅重构。在会议论文中，该模型依赖于对服务时间概率分布的离散化。具体而言，该分布被划分为 q 个连续且互不重叠的区间（分箱），从而得到服务时间分布的简化离散近似。由于在连续情形下观测到任何精确服务时间值的概率为零，这种方法需要引入经验阈值，将连续的服务时间观测映射为离散事件。这些阈值需要根据测得的响应时间进行调优；由于其对测量变异性的敏感性，这不仅增加了开销，也降低了稳健性。相比之下，本文提出的模型直接在连续时间内运行，从而消除了离散化的需求以及相关的经验参数（第五节）。这是相较先前模型的一项重大方法扩展：它改变了攻击的统计表述，移除了原方法中需要经验调优的组件，并能够更加稳健地处理服务时间变异性。

转向连续时间表述也需要应对若干不容忽视的方法学挑战。其中一项主要挑战是确定一种服务时间分布，既能准确捕捉所研究场景的特征，又能在所提出的推断框架中保持解析上的可处理性。此外，连续时间表述促成了新的数学推导。特别是，据我们所知，文献中此前尚未明确推导过式（8）中的表达式。这些进展显著拓展了原有工作的理论基础。此外，正如我们的实验评估（第六节）所示，这些方法学上的进步也带来了不逊于甚至优于先前方法的性能。

II. 背景

我们将在此回顾理解本文所需的部分技术基础，并介绍关键术语和符号。

A. 远程计时攻击

在远程计时攻击中，攻击者根据测得的响应时间推断有关服务器端计算的信息，例如确定系统其他用户的敏感状态信息 [1]。更具体地说，我们假设每次计算都属于 $c \geq 2$ 可能类别中的一个类别，而攻击者通过测量网络请求的响应时间来确定相应远程计算所属的类别。为简单起见，我们将请求的类别称为相关服务器端计算的类别。

为了使我们的研究更加具体，我们聚焦于网络安全领域中的跨站计时攻击，尽管我们的方法更为通用，适用于任何远程计时攻击。跨站计时攻击通过让恶意网站（例如 attacker.com ）的所有者推断其他网站（例如 target.com）用户的敏感信息，绕过网页浏览器的同源策略（SOP）；任何具备传统网络攻击者有限能力的攻击者都可以实施此类攻击 [14]。

在实践中，当受害者使用浏览器访问 attacker.com 时，攻击便开始了。攻击可能针对普通访客，也可能利用网络钓鱼等社会工程学手段诱骗特定受害者。恶意网站使用 JavaScript 从受害者的浏览器向 target.com 发送请求，并测量响应时间，以推断受害者在目标服务器上的状态信息，例如使用 Performance API。比如，目标 Web 应用程序可能设有一个私有区域，生成有效的个人资料页面所需的时间比生成拒绝访问个人资料的错误页面更长。这样一来，攻击者便可以从受害者的浏览器向个人资料页面发送请求，并测量响应时间，以推断受害者在目标服务器上的身份验证状态。这些攻击对用户隐私构成严重威胁。例如，如果攻击者将目标锁定为某家专科医疗中心（如提供 HIV治疗或心理健康服务的医疗中心），那么确认用户是否已登录，就可能泄露高度敏感的健康信息，因为这会暴露用户正在接受的医疗服务及其持续参与情况。如果这些信息遭到滥用，便可能导致歧视、敲诈勒索或严重的精神痛苦。

B. 术语

本节介绍排队论中的关键概念。排队论用于对处理传入请求的系统进行建模。

在排队系统中，请求随机到达，争用有限资源，接受服务，然后离开系统。调度策略决定如何在请求之间分配资源。本文所考虑的模型采用处理器共享（PS）策略 [10], [11], ，将计算资源均匀分配给待处理请求。该策略较为公平地近似了实际系统中常用的轮询调度。

- 服务时间（$\mathcal{Z}$）是指当某个请求是系统中唯一请求时，处理该请求所需的时间。由于系统内部行为（例如垃圾回收、缓存等）会引入变化，服务时间被视为随机变量（r.v.）。每秒处理的作业平均数量（即服务率）记为 μ，因此平均服务时间等于 μ^{-1}。
- 响应时间（$\mathcal{R}$）是指从请求到达系统到生成响应所经过的总时间。
- 等待时间（$\mathcal{W}$）是请求在获得服务之前为争用系统资源所花费的时间。在 PS 系统中，它表示由于与其他请求同时共享资源而产生的延迟，并且等于系统中的响应时间减去服务时间，即与其他请求争用服务器处理能力时的延迟，即$\mathcal{W} = \mathcal{R} - \mathcal{Z}$。
- 到达率（λ）是单位时间内进入系统的预期请求数。
- 负载因子（ρ）用于衡量系统利用率，其定义为到达率与服务率之比（$\rho = \lambda/\mu$对于单服务器系统）。
- 平稳系统是指负载因子在足够长的时间内保持不变的系统。对于排队系统而言，这通常要求系统具有稳定性（即负载因子 ρ <1）。尽管资源使用情况可能有所波动，但系统的统计特性会随时间保持稳定。这是排队论中的典型假设，因为它有助于对等待时间和响应时间进行有意义的分析。在本文余下部分中，我们假设所讨论的系统均为平稳系统。

III. 动机与贡献

远程计时攻击利用了这样一个事实：网络请求的响应时间可能会泄露有关服务器端计算的信息。现有的大多数攻击都遵循相同的总体策略：攻击者收集响应时间，并应用统计分析来确定观测到的差异是否揭示了有关目标系统的信息 [1], [6]。

本文提出的方法不同于这种纯粹的经验性方法。相反，该方法融入了随机建模，以一种新颖的方式结合排队论与贝叶斯统计，从而增强并补充从响应时间中获取的信息。通过这种方式，它解决了以往工作仅依赖对原始响应时间样本进行统计比较所存在的局限性。具体而言，所提出的框架考虑了额外的系统参数，例如服务器负载因子；该因子会显著影响响应时间，进而影响所得推断的正确性。因此，所提出的方法克服了现有方法固有的局限性，生成了一个预测能力很强的模型，提高了远程计时攻击的准确性和实用性。

A. 响应时间与服务时间

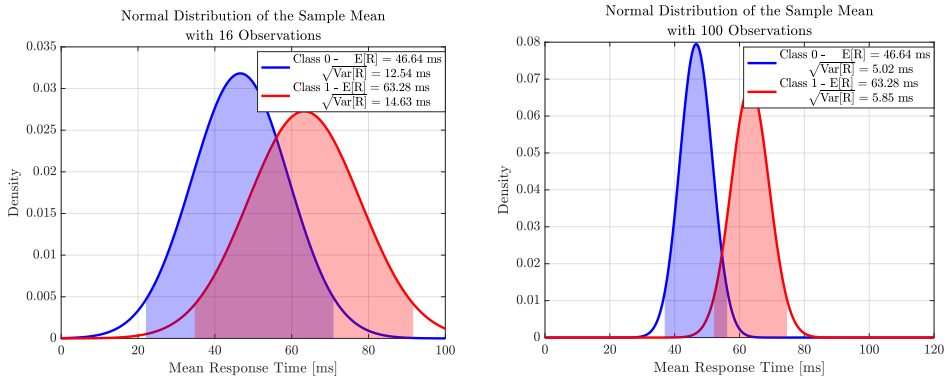
我们研究中的一个关键观察是，定时攻击在实践中难以实施，因为它们利用的是服务时间差异，而攻击者只能测量响应时间。当目标 Web 应用程序处于低负载因子（$\rho \approx 0$）下时，这两个概念大致一致；但当负载因子增加并接近 1 时，响应时间可能会与服务时间产生显著差异。以往研究仅从经验角度对这一问题进行了有限考察。例如，BakingTimer 攻击的作者 [6]展示了一项小规模实验，表明一天中的时间对其攻击效果影响有限；但对于这一非正式且有限的评估，我们仍应持一定保留态度。

为澄清我们的批评并非仅仅是推测，图 1 报告了在本地安装的流行 WordPress Web 应用程序中，对两类别请求（已认证与未认证）计算得到的响应时间均值。直观地说，响应时间的差异可能被滥用于实施跨站计时攻击，通过推断受害者在 WordPress 上的身份验证状态来侵犯其隐私。图中显示，负载因子 ρ 可能会对攻击的精确度产生显著影响。

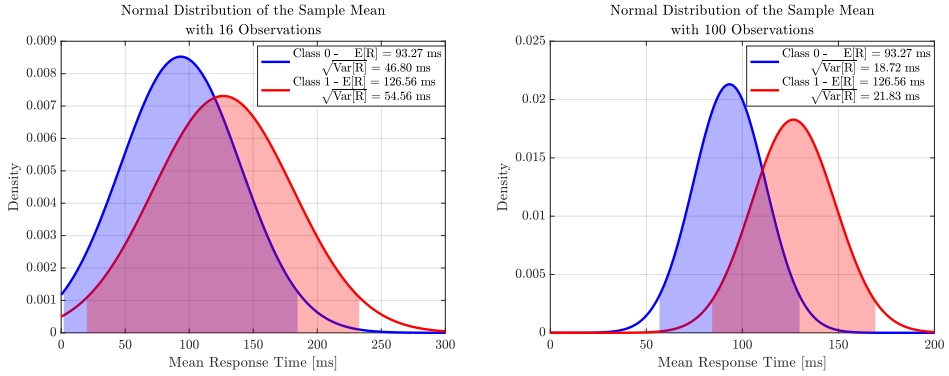
图 1a 和图 1b 展示了中等负载（ρ =0.4）下的情况，此时将观测值数量从 16 增加到 100，可以提高区分不同分布的能力。另一方面，图 1c 和图 1d 显示，在负载较高（$\rho = 0.7$）时，响应时间分布之间的重叠显著增大，即使观测值增加到 100，也难以区分类别。这凸显了传统定时攻击的关键局限：服务器负载增加时，仅依靠响应时间测量可能变得不可靠。

B. 提出的方法

图 2 概述了本文提出的攻击方法。攻击者并不直接使用测得的响应时间得出结论，而是利用这些数据构建目标 Web 应用程序的排队模型。传统上，排队系统将到达率 λ 和服务率μ 作为已知输入参数，用于计算关键性能指标，例如系统的平均响应时间和等待时间。相比之下，我们的方法将这一过程反转：通过收集真实系统的响应时间，我们运用排队论估计未知参数 λ 和μ ，并用它们构建真实系统的抽象模型。根据估计参数得到的排队模型，能够确定不同类别请求的服务时间概率分布。随后，可以将这些信息



(a) $\mu = 40$ 次/秒, $\rho = 0.4$ (b) $\mu = 40$ j/s, $\rho = 0.4$



(c) $\mu = 40$ j/s, $\rho = 0.7$ (d) $\mu = 40$ j/s, $\rho = 0.7$

图1：不同负载因子下的响应时间分布 ρ （固定服务率为 μ）

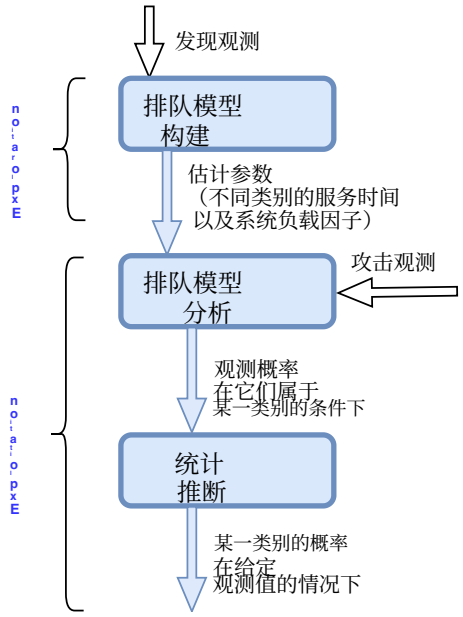


图2：本文提出的方法示意图。

作为输入提供给贝叶斯统计模型，以回答逆向问题：给定一组观测响应时间，请求属于某个特定类别的概率是多少？

该方法具有许多显著优势。首先，该模型通过根据网络请求的服务时间和系统负载预测其响应时间分布，原生地纳入了系统负载因子的影响。因此，给定在特定负载因子下观测到的某个网络请求的响应时间，该模型能够在相同负载条件下，预测服务时间已知的目标网络请求的响应时间分布。随后，模型可以估计观测到的响应时间属于所预测分布的概率，从而无论负载因子如何影响响应时间，都能实现准确的概率估计。这也使得该攻击具有通用性：只要能够根据服务时间，在关于攻击者的最少假设下，将服务器端计算划分为 $c \geq 2$ 类别，就可以实施该攻击。攻击者只需使用自己的浏览器发送属于 c类别的 HTTP 请求（例如，已认证和未认证的请求），而无需事先知道这些请求是否确实可区分。然后，他们利用这些观测到的响应时间来估计服务时间和负载因子，并构建一个能够揭示可利用差异的随机模型。模型构建完成后，即可仅依据测得的响应时间对请求进行分类。

IV. 攻击描述

我们的攻击分为两个阶段，分别称为探索和利用。在探索阶段，

攻击者会在不同时间收集有关目标 Web 应用程序的时序信息，并针对不同类别的请求进行收集。利用阶段会利用探索阶段收集的信息，为目标 Web 应用程序建立随机模型，并实际发起攻击。具体而言，攻击者测量未知类别请求的响应时间，并使用该模型推断这些请求所属的类别。为简明起见，我们假设攻击者希望区分两个可能的类别来描述该攻击；将其扩展到 $c > 2$ 个类别也很直接。

A. 预备知识

我们使用伪代码介绍攻击的两个阶段。探索阶段和利用阶段都需要测量响应时间，并将测量结果输入随机模型。然而，测量 HTTP 请求与其响应之间的时间时，我们也测量了网络往返时间（RTT）。由于我们的模型未考虑 RTT 及其变化，因此需要从测量结果中消除其影响。我们通过实测RTT 并从数据中减去该值来实现这一点。为了可靠地估算响应时间，我们采用了两个子过程。

第一个子过程 TIMEREQUEST(u) 向 URL u 发送 HTTP 请求，并估算其响应时间。该子过程用于利用阶段，实现如下。我们首先向 u 发送 HTTP 请求 r ，并测量 t_r 直到收到响应所经过的时间。 t_r 包含 r 的响应时间和 RTT。除了请求 r，我们还构造另一个请求 r'，其响应时间 t'_r 可作为 RTT的可能估值。适合此用途的请求示例是那些必定会触发重定向的请求。原因在于，这类重定向通常只需极少的服务器端计算。构造此类重定向请求的一种有效方法，是将 URL u 的协议从 HTTPS 改为 HTTP。如果目标网站未启用 HTTP严格传输安全（ HSTS），该 HTTP 请求很可能会触发重定向，转至该网站的 HTTPS 版本。随后，我们可以将 r 的响应时间估算为 $t_r - t'_r$。由于任何实证分析都不可避免地存在测量误差，从测量值（$t_r - t'_r$）中减去 RTT 可能会得到零或负的响应时间。为解决这一问题，我们将所有测量中观察到的 t'_r 最小值估算为 RTT ，并从每次测量中减去该值。如果经过这一调整后响应时间仍为零或负值，则将结果设为 0。1，以避免不切实际的结果。

另一个子过程 TIMEREQUESTWITHCLASS(u, i) 会向 URL u 发送一个属于类别 i 的 HTTP 请求，并估算其响应时间。该子过程用于探索阶段，因此假设攻击者能够构造一个属于类别 i的请求。攻击者如何做到这一点取决于请求的具体类别。例如，已认证请求要求攻击者能够登录目标 Web 应用程序，而未认证请求则始终可以伪造。响应时间的估算采用了与 TIMEREQUEST(u) 子过程相同的技术，只是请求 r 的格式会因其类别而异。

最后，我们的伪代码依赖于随机模型 M的构建，并通过查询该模型来确定目标请求所属类别的概率估计。模型构建和查询由子过程 BUILDMODEL 和 ESTIMATEPROBA 实现，相关内容将在第五节讨论。在本节中，我们将模型 M 视为黑盒，以简化说明。

算法1 探索阶段

```
1: 函数 EXPLORE(u, endtime, tick, samples)
2:   R_0 ← {}      ▷ 0类的响应时间（空字典）
3:   R_1 ← {}      ▷ 1类的响应时间（空字典）
4:   b_0 ← +∞              ▷ 0类的服务时间估计
5:   b_1 ← +∞              ▷ 1类的服务时间估计
6:   t_0 ← 0
7:   t_1 ← 0
8:   t ← 获取当前时间()   ▷ 探索开始时间
9:   当 t ≤ endtime 时执行
10:    R_0[t] ← []       ▷ R_0[t]初始化为空列表
11:    R_1[t] ← []       ▷ R_1[t]初始化为空列表
12:    reqs ← 0
13:    当 reqs < samples 时执行
14:      t_r ← TIMEREQUESTWITHCLASS(u,0)
15:      R_0[t] ← t_r :: R_0[t]   ▷ 将 t_r 追加到列表 R_0[t]
16:      t_r ← TIMEREQUESTWITHCLASS(u, 1)
17:      R_1[t] ← t_r :: R_1[t]   ▷ 将 t_r 追加到列表 R_1[t]
18:      reqs ← reqs + 1
19:    如果 MEAN(R_0[t]) < b_0 则
20:      b_0 ←均值(R_0[t])
21:      t_0 ← t
22:    如果 均值(R_1[t]) < b_1 则
23:      b_1 ←均值(R_1[t])
24:      t_1 ← t
25:    t ← t + tick
26:    等待至(t)                ▷ 等待至下一个时钟周期
27:  返回 (R_0[t_0], R_1[t_1], R_0[t − tick])
```

B. 探索阶段

在探索阶段，攻击者收集有关目标 Web 应用程序的时序信息，这些信息用于构建利用阶段所依据的随机模型。探索阶段从攻击者的浏览器中进行。我们先介绍探索再介绍利用，因为模型构建是利用的前提；不过，攻击者可以延长探索阶段，使其与利用阶段同时运行。这样，攻击者就能持续收集最新的时序信息，从而为利用阶段创建更准确的模型。

算法1所示的探索过程以目标 URL u、探索阶段的结束时间$endtime$、用于定义数据收集频率的时间间隔 $tick$ ，以及每个时钟周期、每个类别要收集的响应时间估计值数量$samples$作为输入。该过程使用字典 R_0， R_1 将每个时间刻度 t 与两个请求类别的响应时间列表关联起来。它计算

算法 2 利用阶段

1: 函数 利用($u, Y^0, Y^1, Y^R, \alpha, samples$)
2: $class \leftarrow \perp$ ▷请求类别最初未知
3: $Y^A \leftarrow []$ ▷ 用于发起攻击的响应时间
4: $reqs \leftarrow 0$
5: $M \leftarrow$ BUILDMODEL(Y^0, Y^1, Y^R)
6: **当** $reqs < samples$ **时**
7: $t_r \leftarrow$时间请求(u)
8: $Y^A \leftarrow t_r :: Y^A$ ▷ 将 t_r 追加到列表 Y^A
9: $reqs \leftarrow reqs + 1$
10: (p_0, p_1) $\leftarrow$ESTIMATEPROBA(M, Y^A)
11: **如果** $(p_0 \geq \alpha \vee p_1 \geq \alpha)$ **则**
12: $class \leftarrow \arg\max^{i \in \{0,1\}}(p_i)$
13: **返回** $class$

b_0, b_1，分别对应0类和1类：通过查找时间刻度 t_0, t_1，找出响应时间均值最小的时刻，即目标系统负载因子可观测到的最低值。该过程返回三个计时观测列表：负载因子观测值最低时0类的响应时间列表（$R_0[t_0]$）、负载因子观测值最低时1类的响应时间列表（$R_1[t_1]$），以及包含最近测得的0类响应时间的列表（$R_0[t - tick]$）。随后，这三个列表将作为下一节所述利用过程的输入 Y^0, Y^1 和Y^R。

如果探索阶段在时间 $starttime$ 开始，则发送到目标 URL 的请求总数为 $2 \cdot (\lfloor (endtime - starttime)/tick \rfloor + 1) \cdot samples$。我们定期进行数据收集，以便准确重建不同时刻负载因子的估计值，从而支持构建在整个利用阶段始终保持最新的模型。

C. 利用阶段

在利用阶段，攻击者利用探索阶段收集的信息构建随机模型，用于推断目标请求的类别。在跨站计时攻击中，利用过程发生在受害者的浏览器中。

算法 2 中所示的利用过程以目标 URL u、在探索阶段收集的计时观测列表 Y^0、Y^1 和 Y^R、置信度参数 $\alpha \in (0, 1]$，用于定义推断两个类别之一所需的概率阈值），以及攻击可执行的最大请求数 $samples$ 作为输入。攻击首先使用 Y^0、Y^1 和 Y^R 构建攻击期间使用的随机模型 M。直观而言，Y^0, Y^1 用于估算两个类别的服务时间，而 Y^R 用于估算最新负载因子。随后，它通过一个循环收集计时观测，直到发送完最大请求数为止。每次迭代时，都会估算 u 处的响应时间，并将其追加到计时观测列表 Y^A 中。之后，随机模型 M 会使用此列表计算两个类别的概率（p_0, p_1）。如果其中任一概率达到置信度阈值 α，则返回对应的类别。否则，如果未达到所需置信度，则返回未知类别 $\perp$。

V. 随机模型

最后，我们介绍攻击中使用的形式化模型。首先，我们介绍用于表示目标 Web 应用程序的排队模型。由于该模型需要使用适当的参数进行初始化，才能预测响应时间，因此接下来讨论如何根据对目标 Web 应用程序的黑盒观测估算这些参数。最后，我们介绍一种统计模型，该模型利用排队模型回答逆向问题，即给定对某个请求服务时间的估计，该请求最可能属于哪个类别？这两个模型分别由 BUILDMODEL 和 ESTIMATEPROBA 子过程使用（第 IV 节）。

A. 排队模型

我们使用 M/M/1 排队模型 [10], [11] 对所考虑的系统进行建模。这意味着我们假设该系统由一台单服务器组成。尽管 Web 应用程序通常采用分布式架构并运行在多台服务器上，但简单的排队模型通过假设单服务器抽象涵盖了真实系统的全部处理能力，仍能成功捕捉这些系统的定量行为 [15], [16]。事实上，对于许多现实世界的网站而言，这种简化依然非常有效，我们的方法和评估结果也证实了这一点。

到达作业遵循泊松过程，到达率为 λ，并按照处理器共享（PS）策略接受服务。其服务时间是服从负指数分布的随机变量的取值，均值为 $E[\mathcal{Z}](= \mu^{-1})$，因此 μ是作业的服务率。系统服从泊松到达过程以及服务时间服从负指数分布，都是便于分析的标准假设，且已被证明具有稳健性 [17], [16]。

M/M/1/PS排队系统已在文献中得到广泛研究，因此，在假设系统处于稳态的情况下，许多关键性能指标都有众所周知的解析表达式。这一假设不仅是排队论文献中的标准做法，而且在这一特定场景中也合乎情理，因为负载变化会随时间波动，但相对于服务时间而言，变化速度相当缓慢 [18]。

a) 可观测性挑战：确定目标系统的代表性模型后，我们必须应对一个根本性挑战：要使用模型本身，就必须能够对其进行参数化。这意味着我们必须确定模型的到达率 λ、服务率 μ，进而确定系统负载 ρ。然而，在我们试图应对的这类现实攻击场景中，我们受到一些严格的黑盒约束：(i) 我们无法直接观察系统的到达率或其处理能力；(ii) 我们无法测量请求的服务时间，只能测量其端到端响应时间。

为克服这一挑战，我们向系统中引入一个探测作业。具体而言，我们假设在表示我们的

M/M/1/PS 系统输入的统计特性相同的作业流中，注入一个服务特性已知且不同于其他作业的作业，并将其响应时间分布作为分析目标。

为确保探测作业不会干扰或改变我们希望测量的系统状态，我们以极低的频率注入这些作业。这保证了探测作业对系统统计数据的总体影响可以忽略不计。在这一非侵入性假设下，系统的后台行为实际上不会受到影响，因此我们可以安全地应用标准平稳排队方程来刻画探测作业的性能[1]。

具体而言，在假设系统处于平稳状态且 $\rho = \dfrac{\lambda}{\mu} < 1$的情况下，服务时间等于确定值 η 的作业的期望响应时间为：

$$E[\mathcal{R}|\eta] = \frac{\eta}{1-\rho}\,. \tag{1}$$

如第 III-B 节所述，在此情境下，我们反向使用 M/M/1/PS 模型，对上述方程求逆，得到：

$$\rho = 1 - \frac{\eta}{E[\mathcal{R}|\eta]}\,, \tag{2}$$

该方程表明，一旦我们能够识别出大小为 η 的作业，并且能够获得 $E[\mathcal{R}|\eta]$的值，就可以计算系统负载因子 ρ.

b) 从确定性服务时间到随机服务时间：在实际场景中，假设服务时间 η是确定性的并不现实，因为现实世界中的执行时间会受到多种系统因素的影响（例如资源争用、上下文切换开销等）。因此，为确保我们的方法具有稳健性，我们将探测作业的服务时间建模为连续随机变量 $\mathcal{V}$，其服从伽马分布；其中，V 表示其累积分布函数（CDF），而 v 表示其密度函数。

伽马分布严格定义在非负值范围内（这符合服务时间的要求），并且其灵活性（通过形状参数和率参数体现）使其能够准确捕捉探测作业请求的服务时间中观察到的变异性，同时保持解析上的可处理性。

随着我们将视角从确定性服务时间转向随机变量 $\mathcal{V}$，我们无法再使用确定性方程 (2) 直接估计 ρ。相反，我们必须按照以下方式，对探测作业服务时间的整个分布进行去条件化，以计算式 (1) 中的期望响应时间：

$$E[\mathcal{R}] = \int_0^\infty E[\mathcal{R}|\eta]\, v(\eta)\; d\eta = \frac{1}{1-\rho}\, E[\mathcal{V}]\,.$$

由此，我们可以轻松推导出所关注的 ρ 表达式：

$$\rho = 1 - \frac{E[\mathcal{V}]}{E[\mathcal{R}]}\,. \tag{3}$$

[1]这些近似假设已通过一项仿真实验成功验证：在由 M/M/1/PS 模型处理的大量探测作业所产生的超长基本请求序列中，直接测量探测作业的性能指标

虽然公式 (3) 使用可直接测量的量（如第 V-C 节所述）为负载因子ρ提供了可靠估计，但这不足以完整地确定模型参数。由于 ρ 仅决定到达率参数与服务率的比值，因此还需要一个额外的独立方程来分别确定 λ 或 μ。此外，如第 V-D 节所示，我们的推断框架依赖于根据探测作业响应时间的概率密度函数推导出的完整累积分布函数，而不仅仅是其均值。

为满足上述两项要求，我们在变换域中分析伽马分布探针的响应时间。在下一小节中，我们将推导其拉普拉斯-斯蒂尔切斯变换（LST）$r^*(s)$，该变换有两项关键作用：(i) 对 LST 求导可得到响应时间方差的闭式表达式，从而提供求出 λ（进而求出 μ）所需的附加方程；(ii) 对 LST 进行数值反演，以获得贝叶斯统计分类器所使用的精确条件响应时间概率密度函数。

B. 响应时间分布及其矩 *nts*

考虑一个标准的 M/M/1/PS 排队模型，并令 $\mathcal{W}_\eta$ 表示服务时间为 η 的作业的（平稳）等待时间随机变量。根据 [19, 推论 1] 可知，当 $\rho < 1$ 时，该随机变量的累积分布函数可表示如下：

$$W(\eta, t) = Pr\{\mathcal{W}_\eta \le t\}\,.$$

然后，记相应的概率密度函数为 $w(\eta, t)$，我们可以定义拉普拉斯-斯蒂尔切斯变换：

$$w^*(\eta, s) \triangleq \int_0^\infty \mathrm{e}^{-st} w(\eta, t)\, dt$$

即以需要 η 个单位服务时间的作业为条件的平衡等待时间分布

$$w^*(\eta, s) = \frac{(1-\rho)(1-\rho y(s)^2)\,\mathrm{e}^{-a(s)\eta}}{b(s) - g(s)\,\mathrm{e}^{-h(s)\eta}}\,. \tag{4}$$

其中，$y(s)$ 是方程的两个正根中较小的一个

$$\rho y(s)^2 - (\lambda + \mu + s) y(s) + \mu = 0$$

并引入以下表达式以简化记号：

$$a(s) = \lambda(1 - y(s))\,, \quad b(s) = (1 - \rho y(s))^2\,,$$
$$g(s) = \rho(1 - y(s))^2\,, \quad h(s) = \frac{\mu(1 - \rho y(s)^2)}{y(s)}\,.$$

$y(s)$ 的表达式如下

$$y(s) = \frac{\lambda + \mu + s - [(\lambda + \mu + s)^2 - 4\mu\lambda]^{1/2}}{2\lambda}\,.$$

令 $\mathcal{R}_\eta$ 表示大小为 η 的作业的响应时间随机变量，并令 $R(\eta,\ t)$ 表示相应的累积分布函数，其定义如下：

$$R(\eta, t) = \begin{cases} 0 & \text{if } t < \eta \\ W(\eta, t-\eta) & t \ge \eta \end{cases}$$

也就是说， $\mathcal{R}_\eta$ 和 $\mathcal{W}_\eta$ 的累积分布函数相同，只是在 t 轴上平移了宽度为 η 的距离。

$$W(\eta,t) = R(\eta, t+\eta)\,, \quad t \geq \eta$$

应用拉普拉斯变换的时移性质，可得在作业需要确定性服务时间 η的条件下，响应时间的拉普拉斯变换:

$$r^*(\eta,s) = \mathrm{e}^{-s\eta} w^*(\eta,s)\,. \tag{5}$$

为了对 $r^*(\eta,s)$进行去条件化，并得到考虑服务时间分布的 $r^*(s)$，我们沿用 [20]中的推理。对于更简单的情形，即输入流中任意作业的服务时间服从负指数分布，可得到$r^*(s)$的显式表达式。记 $z(\eta)$为探测作业的服务时间密度函数，该显式表达式可通过对上述表达式进行去条件化推导得出。

根据全概率定律，可得:

$$r^*(s) = \int_0^\infty r^*(\eta,s)\, v(\eta)\, d\eta\,. \tag{6}$$

反之，假设大小为 η 的作业是一个具有已知分布的探测作业，且该分布通常不同于输入泊松流中作业的分布，则可得到一个对我们的应用有用的结果。

定理1. 给定一个具有 $\rho < 1$的*M/M/1/PS*排队系统，假设探测作业的服务时间服从形状参数为 k 、率参数为 $k\gamma$的伽马随机变量。那么，探测作业响应时间的拉普拉斯-斯蒂尔切斯变换为:

$$\begin{aligned} r^*(s) = {} & (k\gamma)^k (1-\rho)(1-\rho y(s)^2) \\ & \cdot \frac{1}{b(s)h(s)^k} \Phi\left(\frac{g(s)}{b(s)}, k, \frac{s+a(s)+k\gamma}{h(s)}\right)\,, \end{aligned} \tag{7}$$

其中 Φ 是勒奇超越函数。

证明概要。通过对探测作业的伽马分布服务时间进行条件化，并对其条件响应时间变换进行积分，即可得到该结果。随后，将所得积分转换为级数表示并进行求值，从而直接得到最终表达式。完整的代数推导和支持性引理见补充材料。

□

根据探测作业响应时间的拉普拉斯变换，可以推导出其解析方差。为简洁起见，此处省略其代数推导，完整计算过程见补充材料。结果如下:

$$\begin{aligned} \mathrm{Var}(\mathcal{R}) = {} & \left.\frac{d^2 r^*(s)}{ds^2}\right|_{s=0} - \left(\left.\frac{dr^*(s)}{ds}\right|_{s=0}\right)^2 \\ = {} & \frac{1}{\gamma^2 k \left(\lambda - \frac{\lambda}{\rho}\right)^4} \left(\frac{\lambda}{\rho}\right) \left(\left(\lambda - \frac{\lambda}{\rho}\right)^2 \left(\frac{\lambda}{\rho}\right) \right. \\ & \left. + 2\gamma k \lambda \left(\frac{\lambda}{\rho} - \lambda\right) + 2\gamma^2 k \lambda \left(\frac{\gamma k}{\gamma k - \lambda + \frac{\lambda}{\rho}} - 2\gamma^2 k \lambda\right)^k \right). \end{aligned} \tag{8}$$

这一结果的重要性在于，正如前文所述，探测作业响应时间的方差仅取决于 ρ、 λ 以及探测作业服务时间分布的（已知）参数，因此可以通过对这一结果求逆得到 λ 。

值得注意的是，尽管推导公式 7 和 (8) 的过程完全是机械性的，而非概念性的，但据我们所知，所得的显式表达式此前尚未见诸文献。

C. 参数估计

本节介绍我们用于配置排队模型的方法。具体而言，我们需要估计系统参数（λ 和 μ）以及两个请求类别（0类和1类）的服务时间分布参数，以便能够区分这两类请求。为此，我们收集以下三组特定测量数据:

- $Y^0 = (Y_1^0,\ Y_2^0,\ \ldots,\ Y_{N^0}^0)$: 在观测到的最低负载因子下测得的 0 类请求响应时间。
- $Y^1 = (Y_1^1,\ Y_2^1,\ \ldots,\ Y_{N^1}^1)$: 在观测到的最低负载因子下测得的 1 类请求响应时间。
- $Y^R = (Y_1^R,\ Y_2^R,\ \ldots,\ Y_{N^R}^R)$: 最新负载因子下 0 类请求的响应时间。

如前所述，由于系统对攻击者而言是一个黑盒，因此无法精确确定系统在任意时刻的负载因子或其他定量特征。因此，参数估计依赖一系列合理假设，并利用所收集观测值的特性。

如第IV-B节所述，向量 Y^0 和 Y^1是根据平均值最低的一部分观测值推断得出的。这些观测值被选来代表零等待时间条件，从而为最低负载条件下各类别的服务时间提供最准确的估计。同样，向量 Y^R 取自最新一组观测值，反映系统最新的运行状态。借助这种方法，可以按照下文概述的方法，估计两个类别的服务时间以及系统的负载因子。

a) 服务时间分布的估计: 为了对每个类别的服务时间建模，我们对低负载测量向量拟合伽马分布。由于两个类别的估计方法完全相同，我们省略类别

上标（0 和 1），并给出通用低负载观测向量 Y 的推导。

令 $\mathcal{V}$ 表示请求的服务时间，并将其建模为形状参数为 k 、均值为$1/\gamma$ （因此率参数为 $k\gamma$）的伽马随机变量：

$$V(t,k,\gamma)=\Pr\{\mathcal{V}\le t\}=1-\sum_{n=0}^{k-1}\frac{(k\gamma t)^n}{n!}e^{-k\gamma t}. \quad (9)$$

具体而言，假设在低负载条件下排队延迟可以忽略不计，则 Y 中的观测响应时间可直接反映服务时间。因此，我们对 Y中收集的样本直接拟合伽马分布，采用最大似然估计（MLE）来估计参数 k 和 γ 。

b) 系统参数估计：在刻画了服务时间分布后，我们现在可以利用 Y^R中的测量值来估计系统负载因子$\hat{}\rho$。

根据式（3），注意到 $E[\mathcal{V}]=1/\gamma$，并令观测值的样本均值等于解析均值 $E[\mathcal{R}]=Y^R$，即可得到$\hat{}\rho$，如下所示：

$$\widehat{\rho}=1-\frac{1}{\gamma\overline{Y^R}}.$$

采用类似的推理，为了估计到达率，我们令根据观测值 Y^R计算出的经验方差（即 Var(Y^R)）等于式（8）给出的解析方差 Var($\mathcal{R}$)。

由于公式 (8) 包含超越项，因此无法通过解析方法求得 λ的显式闭式表达式。为此，我们采用 MATLAB 提供的数值求根过程，求出目标函数 Var($\mathcal{R}$) –Var(Y^R) 的零点，并求得 $\widehat{\lambda}$。确定 $\widehat{\lambda}$ 后，即可根据 $\rho=\lambda/\mu$ 给出的负载因子关系，轻松计算出 $\widehat{\mu}$ 。本节介绍的方法已通过 BUILDMODEL 子过程纳入算法 2。

D. 统计模型

本节介绍一种统计模型，该模型根据上一节中对响应时间的观测值及估计的系统参数，估算一组已观测到的响应时间属于特定请求类别的后验概率。该模型将由 ESTIMATEPROBA 子过程（算法 2）使用，以实现对公式 (14) 的评估，即贝叶斯分类框架。下文将展示该结果的数学推导。

令 $\Theta\in\{0,1\}$ 为表示当前活跃服务时间分布的（隐藏）类别索引。对于每个类别 $i\in\{0,1\}$，我们使用第五 V-C 节中估计的背景 $\hat{}$排队参数 λ 和 $\widehat{\mu}$，以及该类别特有的服务时间分布参数（γ_i，k_i），对响应时间分布的拉普拉斯-斯蒂尔切斯变换（拉普拉斯变换） $r_i^*(s)$进行参数化。

理论上，单个观测响应时间 y 的条件概率密度函数（概率密度函数）可以通过以下方式获得对在 y处精确计算的响应时间拉普拉斯变换取拉普拉斯逆变换：

$$f_Y(y\mid\Theta=i)=\mathcal{L}^{-1}\{r_i^*(s)\}(y), \quad (10)$$

其中， $\mathcal{L}^{-1}$ 表示拉普拉斯逆变换。

然而，直接计算式（10）会面临严重的数值挑战。这是因为在 M/M/1/PS 系统中，到达的作业有非零概率经历零等待时间，也就是说，其响应时间恰好等于服务时间。因此，理论概率密度函数 $f_Y(y\mid\Theta=i)$ 在最小服务时间边界处存在明显的数学不连续性。在该边界附近直接对概率密度函数的拉普拉斯变换进行逆变换，会引发严重的数值振荡。

为了解决这两项挑战，我们将数值反演的对象从概率密度函数（PDF）转为累积分布函数（CDF） $F_R(y\mid\Theta=i)$。该函数连续、有界，且易于进行数值稳定的反演。随后，我们使用在较小正区间 ϵ 上平滑处理的中心差分数值导数来计算实际的概率密度函数：

$$f_Y(y\mid\Theta=i)\approx\frac{F_R(y+\epsilon\mid\Theta=i)-F_R(y-\epsilon\mid\Theta=i)}{2\epsilon}, \quad (11)$$

其中，累积分布函数通过数值反演计算：

$$F_R(y\mid\Theta=i)=\mathcal{L}^{-1}\left\{\frac{1}{s}r_i^*(s)\right\}(y). \quad (12)$$

基本上，公式（11）是关于 $\partial F_R(x|\Theta)/\partial x$ 的平滑概率密度函数，不会改变给定 Θ时随机变量 R 的均值。

为计算公式（12）中的拉普拉斯逆变换，我们采用集中矩阵指数（CME）方法 [21]。与其他广泛使用的技术相比，该方法以其出色的数值稳定性和光滑的近似效果而著称。我们利用 [22] 中提供的稳健 MATLAB 软件包，计算属于类别 i的 N^A 个观测响应时间 $Y^A=(y_1^A,\ldots,y_{N^A}^A)$ 向量的联合似然：

$$g_{Y^A}(\Theta=i\mid y_1^A,\ldots,y_{N^A}^A)=\prod_{n=1}^{N^A}f_Y(y_n^A\mid\Theta=i). \quad (13)$$

令 $\pi_i\equiv\Pr(\Theta=i)$表示观测到类别 i（即观测到 $\Theta=i$）的先验概率，我们应用贝叶斯定理，根据观测值 Y^A（即利用阶段收集的观测值）计算 $\Theta=i$ 的后验概率：

$$\Pr(\Theta=i\mid y_1^A,\ldots,y_{N^A}^A)=\frac{g_{Y^A}(\Theta=i\mid y_1^A,\ldots,y_{N^A}^A)\pi_i}{G}, \quad (14)$$

其中，归一化分母为：

$$G=\sum_{i\in\{0,1\}}g_{Y^A}(\Theta=i\mid y_1^A,\ldots,y_{N^A}^A)\pi_i$$

该后验概率是我们攻击中使用的分类指标。

E. 模型实现细节

所提出模型的实现中，一个重要方面是确保输出在物理意义上合理，尤其是在某些情况不可能发生时。例如，等待时间不可能为负数，因此，因测量问题而产生、并可能作为输入传递给数值反演的任何负值，都会自动设为零，这反映了负等待时间发生的概率为零。同样，如果测量误差导致估计的负载因子为零，我们便完全跳过数值变换反演，而是使用累积分布函数，将观测到的计时数据视为某一类别或另一类别的服务时间。此外，为避免拉普拉斯逆变换产生的概率为零或接近零（否则，即使考虑了所有其他观测值，单个响应时间也可能主导或不成比例地影响最终结果），我们将输出概率限制在区间 $[0.02, 0.98]$ 内。最后，为了考虑经验分析中固有的测量不确定性，我们引入一个阈值ϵ （用于公式（11）），以此定义一个区间，真实响应时间预计落在该区间内。在实验中，我们将此阈值设为 15，因为响应时间通常为数百毫秒，这与我们原始工作中使用的值 [13]一致。

VI. 实验评估

在本节中，我们将在实际环境中展示我们的攻击，并将其效果与最先进的基线方法进行比较。从伦理角度来看，由于我们的攻击是在真实网站上进行测试的，我们谨慎行事，尽量减少潜在危害，并确保收益大于风险。具体而言，我们的所有攻击都只针对自己的账户，未侵犯其他网络用户的隐私。此外，我们已负责任地将所有发现的隐私问题披露给相关联系人，以供其考虑。

A. 实验设置

我们在 Tranco 排名 [23], 中的 20 个热门网站上评估了攻击的有效性，考察其在真实环境中的表现，以及成功攻击部署在互联网中、此前未见过的 Web 应用程序的能力。我们的评估针对跨站计时攻击，其中攻击者的目标是泄露未知请求的类别。我们仅考虑两种请求类别，分别称为正类（已认证）和负类（未认证）。

1) 基线方法：作为第一个对照基线，我们重新实现了最先进的跨站计时攻击 BakingTimer [6], 。BakingTimer 是一种通用攻击策略的具体实现：攻击者收集多组计时观测，并应用统计检验来推断请求类别。更具体地说，攻击者首先收集用于区分两种类别的计时观测组 U_1，U_2。如果统计检验判定 U_1 和 U_2 很可能来自不同的分布，就可以发动攻击。随后，攻击者会收集一组属于某个未知类别的计时观测 D，并尝试泄露该类别。最后，D 会进行一次针对 U_1 和 U_2的统计检验。如果两项检验中恰好有一项通过，攻击者就会按照通过的检验所建议的方式进行预测；否则，统计证据不足以预测任何类别（弃权）。

在 BakingTimer 的具体案例中，D 包含“时间差值”，通过一组响应时间与一组网络往返时间估计值之间的差异来估算服务器端计算时间，这与我们在攻击中采用的方法类似。计算时间差值集合D 需要发送 $2 \cdot |D|$ 个请求。BakingTimer 使用经典的 t 检验，在跨站场景下根据两个目标类别的时间差值 U_1，U_2 来披露网络请求的类别。

作为第二种基线，我们将攻击的一种版本作为对照，该版本基于我们在会议论文 [13], 中提出的随机模型。与新模型不同，它依赖于服务时间概率分布的离散化。具体而言，我们从用于估算每个类别服务时间的观测响应时间数组出发，取最小观测值与最大观测值之间的范围，并将其划分为 q 个连续且互不重叠的区间，即分箱。对于每个分箱，我们统计其中包含的观测值数量，以估算该范围出现的可能性，并用质心及其对应的概率表示每个分箱。这样便可得到每个类别服务时间分布的简化离散近似。随后，我们使用得到的质心和概率来估算系统参数，进而计算一系列响应时间属于给定类别的概率。

此外，在原始模型中，我们设置了一个阈值 δ，以考虑到由于测量误差，精确的服务时间质心不太可能与观测值完全一致。该阈值定义了一个小区间，在此区间内的观测值被视为与零等待时间对应的观测值相当。除了这两个额外的超参数外，新旧模型的接口相同，因此我们对两个模型都使用按照第 IV 节所述流程收集的同一组观测响应时间数组。

2）超参数：对于 BakingTimer，我们将 t 检验的显著性水平设为 0.05，这是标准的推荐做法；然后将所得的 p 值与该阈值进行比较，以判断是否具有统计显著性。请求数量设为 20，因为原论文报告称，数量在 10 到 50 [6] 之间时，性能都令人满意。我们选择 20 个请求，作为攻击效果与现实性的折中，因为受害者可能只会在攻击者的页面上停留很短时间。请注意，攻击中实际使用 20 个请求时，需要发送 40 个请求，分成 20 对，以计算时间差。

对于我们的攻击，我们将概率阈值 α 设为 0.90，也就是说，我们要求预测类别的概率至少为 90%；同时将 HTTP 请求的最大数量设为 20，与 BakingTimer 相同，以便进行公平比较。原始模型所需的其他参数则设为与会议论文中相同的值 [13],，即阈值 δ 设为 0.05，分箱数量

设为 4。

B. 真实环境中的实验

在跨站计时攻击中，探索由攻击者的浏览器发起，而利用则通过受害者的浏览器进行。为开展这些实验，我们使用了两台独立的计算机，一台用于受害者，另一台用于攻击者；两台计算机均连接到同一有线网络，并通过互联网向目标网站发送请求。攻击者的目标是通过收集响应时间，判断目标用户是否已登录 Web 应用程序。

我们将攻击的有效性与 Baking- Timer [6] 以及基于原始随机模型 [13]的攻击变体进行比较。我们将研究重点放在 Tranco 排名 [23] 前 20 的、可访问私有区域的网站上，并评估它们对跨站计时攻击的脆弱性。我们按照网站受欢迎程度从高到低手动浏览，直到找到登录表单；如果在合理时间内未找到登录表单，或无法轻松完成注册（例如网站要求付费订阅），则舍弃该网站。对于每个网站，我们都会注册一个新账户（如果尚未拥有账户），然后登录。访问私有区域后，我们会从浏览器中删除所有 SameSite 属性未设为 None 的 Cookie，并手动检查我们是否仍处于该网站的登录状态。未通过此项人工测试的网站不易受到跨站计时攻击，因为无法通过跨站请求 [24]建立其会话。或许令人意外的是，在分析的 20 个网站中，有 13 个使用停用了 SameSite 属性的 Cookie 来建立会话。我们针对每个可能存在漏洞的网站进行 16 次攻击尝试（已认证状态下 8 次，未认证状态下 8 次），以比较不同攻击方法的性能，共进行了 208 次攻击尝试。

在一天中的不同时段对每个网站进行攻击，以考虑日常运营期间服务器负载的波动。表 I 汇总了所有受测网站的数据，报告了每种攻击产生的正确预测、错误预测和弃权数量。当攻击在达到允许的最大请求次数后，因统计证据不足而无法预测任何类别时，就会发生弃权。

总体而言，基于新模型的攻击优于原始模型和 BakingTimer。与原始模型相比，新模型的弃权次数更少（-15），错误也更少（-4），因此正确分类多出 19 次。相比之下，与 BakingTimer 相比，新模型的错误分类次数更多，弃权次数则更少。不过，这种劣势只是表面上的，因为 BakingTimer 的有效性受到其大量弃权的严重限制：它在 50% 的情况下不提供预测。此外，在新模型给出分类而 BakingTimer 弃权的 59 个实例中，新模型有 37 次预测正确（63%）。这表明，新模型所带来的额外分类并非

表 I：我们的攻击分别结合新模型和原始模型时，与 BakingTimer 的比较。

答案	新模型	原始模型 [13]	BakingTimer [6]
正确	132	113	95
错误	32	36	10
弃权	44	59	103

以牺牲可靠性为代价，而是在多数情况下给出了正确预测。事实上，在全部 208 次攻击尝试中，新模型有 63% 的预测正确，而原始模型和 BakingTimer 的正确率分别仅为 54% 和 46%。值得注意的是，如前所述，这些改进是在使用更少超参数、投入更少调优工作的情况下实现的。

图 3 展示了对于每个网站，各方法产生的预测总数，并将结果分为正确（蓝色）和错误（橙色）。三种方法在 Adobe 和 Skype 上的表现都很差，这表明这些网站可能不易受到此类攻击，因为不同类别请求之间的响应时间可能过于接近。对于其余网站，新模型的性能要么与最先进水平和原始模型中表现最佳的方法相当，要么在减少弃权数量的同时大幅增加正确预测的数量。各网站的结果与上文观察到的总体趋势一致：总体而言，新模型的弃权数量最低，或与弃权数量最少的基线相当。此外，弃权数量的减少带来的是更多正确预测，而非更多错误预测。事实上，在所有网站上，就正确预测数量而言，新模型要么与表现最佳的基线持平，要么产生更多正确分类。这进一步证实了此前观察到的趋势：新模型支持的额外分类在大多数情况下都是正确的。唯一的例外是 LinkedIn，在该网站上，表现最佳的基线优于新模型，少一次弃权、少两次错误预测，并多出三次正确预测。尽管如此，新模型的表现仍然优于原始模型。

VII. 相关工作

远程计时攻击受到了研究界的广泛关注。Felten 和 Schneider 证明，可以利用 Web 缓存带来的性能提升来确定目标用户是否访问过某个网页 [4]。后续研究同样利用 Web 缓存重建目标用户的地理位置 [25]。我们的工作并不直接适用于基于缓存的计时攻击，因为我们的框架所建模的计时侧信道是服务器端计算所需的时间。由服务器端计算时间促成的远程计时攻击，其先驱是 Bortz 和 Boneh，他们针对 Web 应用程序提出了不同的攻击方法 [5]。后续研究进一步推广了

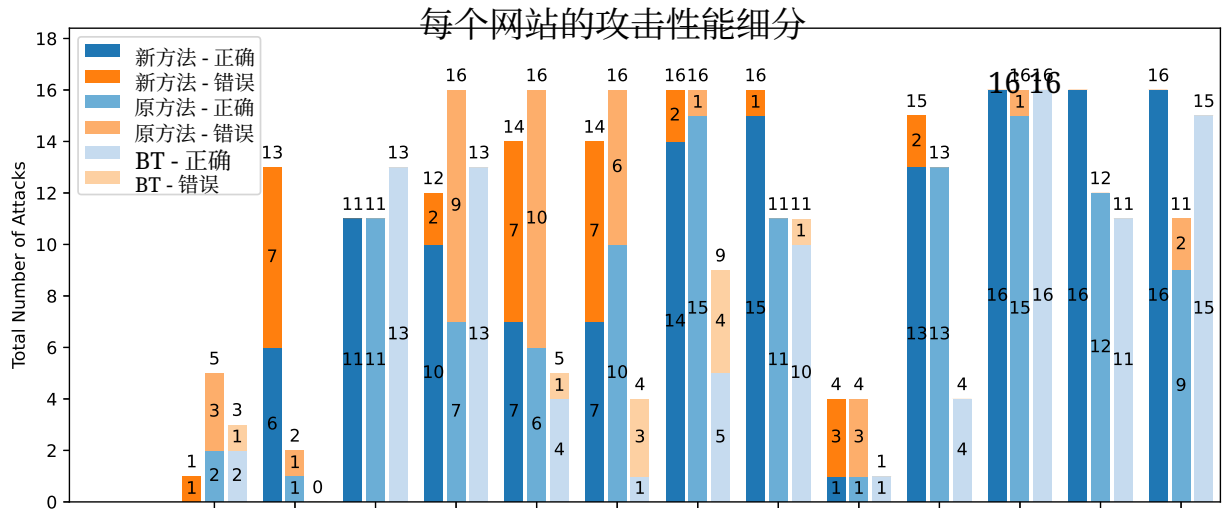


图 3：不同攻击（新方法、原始方法和 BakingTimer）在各网站上的性能 .

他们的思想，以涵盖现代浏览器引入的其他攻击向量 [26], [27] ，并提出了旨在更有效地侵犯用户隐私的新型计时攻击 [28], [6]。我们的工作与现有攻击之间的一个关键区别在于，使用形式化建模来处理远程计时攻击的复杂性，详见第三节。事实上，我们的实验评估表明，像 BakingTimer [6] 这样的经验型最先进攻击的有效性并不理想。即使针对真实网站进行测试，我们的攻击也能更好地适应目标 Web 应用程序负载因子的变化，并且精确度显著更高。

唯一一篇更严谨地探讨远程计时攻击形式化理论的论文，是 Crosby 等人的开创性研究 [1]。作者指出，测量响应时间时，必须考虑服务器负载因子；他们将这一影响评估为测量值中的加性抖动效应。然而， [1] 中提出的场景与我们的场景存在显著差异。首先，该研究假设每个类别的服务时间都是已知的常数。对于其场景中所考察的观测值类型，这一假设或许合理，但对于 Web 服务而言则不太适用，因为处理时间可能随系统状态而变化，例如受到资源争用和工作负载变化的影响 [29]。其次，该研究将抖动建模为服务时间的加性分量；与我们在此考虑的轮询调度相比，这一假设更适用于先入先出调度 [30]。轮询调度更能准确描述当今 Web 服务器中调度器的实际行为 [31]。

VIII. 结论

在本文中，我们提出使用随机模型来提高远程计时攻击的有效性，并在 [13]所介绍的工作基础上进行了改进。我们的方法从目标 Web 应用程序的排队模型出发对其发起攻击，并使用贝叶斯统计推断所关注的特定网络请求所属的未知类别。与此前的经验方法相比，我们的攻击更加精确，并能提供定量的概率界。作为未来的工作，我们希望尝试除本文所考虑的 M/M/1/PS 模型之外的其他类型排队模型。特别是，我们计划进一步研究微服务架构带来的挑战。此外，我们还将探索通过引入更复杂的排队网络框架来增强分析模型的可能性。事实上，支持不同类型的模型对于推广到不同的 Web 应用程序和环境十分重要，例如，攻击者可以执行初步测试，以确定目标 Web 应用程序最适合采用哪种模型。我们还计划通过扩大实验规模并测试我们的方法在不同干扰因素下的稳健性，来改进实验评估，例如评估地理位置对攻击有效性的影响。

致谢

本研究获得Agenzia per la cybersicurezza nazionale“网络安全领域第 XL 周期博士研究促进计划”的支持——CUP N.H71J24001710005。文中观点仅代表作者立场，不代表资助机构。

参考文献

[1] S. A. Crosby、D. S. Wallach 和 R. H. Riedi，《远程计时攻击的机遇与局限》，ACM Trans. Inf. Syst. Secur.，第12卷，第17:1–17:29页，2009年。[2] R. Rivest、A. Shamir 和 L. Adleman，《获取数字签名和公钥密码系统的方法》，Comm. of the ACM，第21卷，第120–126页，1978年。[3] D. Brumley 和 D. Boneh，《远程计时攻击切实可行》，Comput. Networks，第48卷，第701–716页，2005年。

[4] E. W. Felten 和 M. A. Schneider，《针对网络隐私的定时攻击》，载于 CCS 2000：第7届 ACM计算机与通信安全会议论文集，希腊雅典，2000年11月1日至4日。ACM，2000年，第25–32页。[5] A. Bortz 和 D. Boneh，《通过计时暴露 Web 应用程序中的私人信息》，载于第16届国际万维网会议论文集（WWW 2007），加拿大阿尔伯塔省班夫，2007年5月8日至12日。ACM，2007年，第621–628页。[6] I. S´anchez-Rola、D. Balzarotti 和 I. Santos，《Bakingtimer：服务器端请求处理时间的隐私分析》，载于第35届年度计算机安全应用会议论文集（ACSAC 2019），美国波多黎各圣胡安，2019年12月9日至13日。ACM，2019年，第478–488页。[7] W. Wang、G. Casale、A. Kattepur 和 M. Nambiar，《QMLE：从排队数据中对服务需求进行统计推断的方法》，ACM Trans. Model. Perform. Evaluation Comput. Syst.，第3卷，第17:1–17:28页，2018年。[8] F. Baccelli、B. Kauffmann 和 D. Veitch，《排队论中的逆问题与互联网探测》，Queueing Syst. Theory Appl.，第63卷，第59–107页，2009年。[9] D. A. Menasc´e、V. Almeida 和 L. W. Dowdy，《Performance by Design》，培生，2004年。[10] L. Kleinrock，《排队系统》第1卷：理论。Wiley-Interscience，1975年。[11] M. Harchol-Balter，《计算机系统性能建模与设计：排队论实践》，剑桥大学出版社，2013年。[12] S. Bozzolan、D. Olliaro、S. Calzavara、A. Marin、G. Balbo 和 M. Sereno，《Artifacts》，https://github.com/Asterius27/SecPerf-Artifacts/tree/journal extension，2025年，包含本文所有相关 Artifacts 的代码库。[13] ——，《远程计时攻击的随机模型》，Proc. Priv. Enhancing Technol.，第2025卷，第545–559页，2025年。[14] D. Akhawe、A. Barth、P. E. Lam、J. C. Mitchell 和 D. Song，《迈向网络安全的形式化基础》，载于第23届 IEEE计算机安全基础研讨会论文集（CSF 2010），英国爱丁堡，2010年7月17日至19日。IEEE计算机学会，2010年，第290–304页。[15] J. Cao、M. Andersson、C. Nyberg 和 M. Kihl，《使用 m/g/1/k*ps 排队系统对网络服务器性能建模》，载于第10届国际电信会议（ICT）论文集，第2卷。IEEE，2003年，第1501–1506页，第2卷。[16] C. U. Smith 和 L. G. Williams，《Performance Solutions：创建响应迅速、可扩展软件的实用指南》。Addison-Wesley Professional，2001年。[17] H. H. Liu，《Oracle数据库性能与可扩展性：定量方法》。Wiley-IEEE计算机学会出版社，2011年。[18] J. von Kistowski、N. Herbst、S. Kounev、H. Groenda、C. Stier 和 S. Lehrig，《负载强度曲线的建模与提取》，ACM Trans. Auton. Adapt. Syst.，第11卷，第23:1–23:28页，2017年。[19] E. G. Coffman、R. R. Muntz 和 H. Trotter，《处理器共享系统的等待时间分布》，Journal of ACM，第17卷，1970年。[20] J. A. Morrison，《处理器共享系统的响应时间分布》，SIAM应用数学杂志，第45卷，167，1985 152pp. –。[21] G. Horv´ath、I. Horv´ath、S. A. Almousa 和 M. Telek，《使用集中矩阵指数分布进行数值拉普拉斯逆变换》，Perform. Evaluation，第137卷，第1–22页，2020年。[22] G. Horvath，《基于CME的数值逆拉普拉斯变换方法》，https://it.mathworks.com/matlabcentral/fileexchange/71511-a-cme-based-numerical-inverse-laplace-transformation-method，2024年，MathWorks。[23] V. L. Pochat、T. van Goethem、S. Tajalizadehkhoob、M. Korczy´nski 和 W. Joosen，《Tranco：一种经强化以抵御操纵、面向研究的顶级网站排名》，载于第26届年度网络与分布式系统安全研讨会（NDSS 2019），美国加利福尼亚州圣迭戈，2019年2月24日至27日。互联网协会，2019年。[24] S. Khodayari 和 G. Pellegrino，《SameSite 的现状：研究 SameSite Cookie 的使用情况、有效性与适用性》，载于第43届 IEEE安全与隐私研讨会（SP 2022），美国加利福尼亚州旧金山，2022年5月22日至26日。IEEE，2022年，第1590–1607页。[25] Y. Jia、X. Dong、Z. Liang 和 P. Saxena，《我知道你去过哪里：通过浏览器缓存发动的地理位置推断攻击》，IEEE Internet Comput.，第19卷，第44–53页，2015年。[26] T. van Goethem、W. Joosen 和 N. Nikiforakis，《时钟仍在滴答：现代网络中的定时攻击》，载于第22届ACM SIGSAC计算机与通信安全大会论文集，美国科罗拉多州丹佛，2015年10月12日至16日。ACM，2015年，第1382–1393页。

[27] S. Lee、H. Kim 和 J. Kim，《使用应用程序缓存识别跨源资源状态》，载于第22届年度网络与分布式系统安全研讨会（NDSS 2015），美国加利福尼亚州圣迭戈，2015年2月8日至11日。互联网协会，2015年。[28] N. Gelernter 和 A. Herzberg，《跨站搜索攻击》，载于第22届ACM SIGSAC计算机与通信安全大会论文集，美国科罗拉多州丹佛，2015年10月12日至16日。ACM，2015年，第1394–1405页。[29] K. Gardner、M. Harchol-Balter、A. Scheller-Wolf、M. Velednitsky 和 S. Zbarsky，《冗余-d：d 种冗余选择的威力》，《运筹学》，第65卷，第1078–1094页，2017年。[30] P. G. Harrison 和 N. M. Patel，《通信网络与计算机体系结构的性能建模》（International Computer S，第1版。美国：Addison-Wesley Longman Publishing Co., Inc.，1992年。[31] M. Harchol-Balter、N. Bansal、B. Schroeder 和 M. Agrawal，《在 Web 服务器中实现 SRPT 调度》，卡内基梅隆大学计算机科学学院，技术报告，2000年。



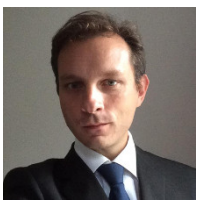
Simone Bozzolan 于2024年获得威尼斯卡福斯卡里大学计算机科学硕士学位，目前正在该校攻读计算机科学博士学位。他的研究聚焦于网络安全，尤其关注身份验证与授权系统。他致力于研究分析、检测和缓解 Web 应用程序漏洞的技术，并结合实证方法与形式化方法，提升基于Web的系统的安全性与韧性。



Diletta Olliaro 是瑞典皇家理工学院的博士后研究员，在TECoSA研究中心开展工作。2025年，她以优等成绩获得威尼斯卡福斯卡里大学计算机科学博士学位。她的研究聚焦于排队论、乘积形式解模型，以及更广泛而言，网络和电信系统的随机建模，尤其关注评估和提升其性能与可靠性。



Stefano Calzavara 是意大利威尼斯卡福斯卡里大学计算机科学副教授。他的研究处于形式化方法与软件安全的交叉领域，并应用于网络安全和人工智能安全。他在这些领域发表了70余篇论文，包括发表于 IEEE S&P、ACM CCS、NDSS、USENIX Security 和万维网大会等顶级会议的论文。他曾在研究领域担任多个重要职务，包括担任 IEEE CSF（2022年和2023年）的程序委员会主席，以及安全领域主席 TheWeb Conference（2024年）以及 USENIXSecurity（2026年和2027年）的副主席。他目前担任《计算机安全期刊》主编。他的研究曾获得 IEEE S&P 2025年和 ETAPS 2013 年最佳论文奖。



Andrea Marin 于2009年获得威尼斯大学计算机科学博士学位。他是该校计算机科学教授。他是同行评审国际期刊和会议论文集上100多篇技术论文的（共同）作者。目前，他的研究重点是运用随机建模技术评估计算机系统的性能和可靠性。他是IEEE高级会员。



Gianfranco Balbo 于1979年获得美国普渡大学计算机科学博士学位，目前是都灵大学信息学系荣休教授；他于2016年11月从正教授职位上退休。他的研究兴趣包括计算机系统性能评估、系统生物学、排队网络、随机Petri网和排队论。他是广义随机Petri网（GSPN）形式体系的提出者之一，该体系是在他开展有关性能研究期间开发的

并行计算机的性能。其研究成果发表于100余篇会议和期刊论文以及10部著作（作为作者或编辑）。Gianfranco Balbo是ISI高被引研究者，也是都灵科学院成员。



Matteo Sereno 于1987年获得意大利萨莱诺大学计算机科学学士学位，并于1992年获得意大利都灵大学计算机科学博士学位。他目前是意大利都灵大学计算机科学系的正教授。其研究兴趣包括通信网络、分布式系统、排队网络和随机Petri网模型的性能评估与建模、博弈论，以及它们在计算机和电信系统中的应用。