基于 PoW 的 Web 服务保护方法

高建 白晓菲 张亮

(复旦大学计算机科学与技术学院 上海 201203)

(上海市数据科学重点实验室 上海 201203)

（上海智能电子与系统研究院 上海 201203）

摘 要 当 Web 服务开放给公共使用时，容易遭受爬虫类的资源滥用和攻击。现有的保护手段一般基于验证码或某种形式的审计，各自存在失效或影响服务质量的场景，并且给跨组织服务组合等合作方式带来了很高的门槛。本文提出一种基于工作量证明的 Web 服务保护方案，从另一种思路出发，在实现有效保护的同时，放行合理的大规模服务请求，降低了进行服务整合的门槛，促进了组织间的合作，起到优化 Web 服务资源分配的作用。

关键词 Web 服务保护；工作量证明；反爬虫；服务组合

中图法分类号 **** DOI 号 *投稿时不提供 DOI 号* 分类号

Protecting Web Services with PoW Systems

GAO Jian BAI Xiao-Fei ZHANG Liang

(School of Computer Science, Fudan University, Shanghai, 201203, China)

(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, 201203, China)

(Shanghai Institute of Intelligent Electronics & Systems, Shanghai, 201203, China)

Abstract Web services are often vulnerable to resource abuse and attacks by web robots, especially when open to public use. Existing protection methods are generally based on CAPTCHA systems or auditing strategies. Their limitations on effectiveness, complexity and/or performance may affect service quality, and might block ad-hoc cooperation via service orchestration. This paper proposes a web service protection scheme using proof-of-work systems. It can produce comparable protection with existing systems, but accepts reasonable robotic requests. Based on the fact that not all robotic traffic cause abuse, we are able to ease the restrictions on ad-hoc service integrations, thus promote cooperation, and ultimately help to rearrange Web services resources.

Key words Web service protrction; Proof-of-Work system; Web crawler detection; Service orchestration

1 引言

对于服务提供者而言，面向公共开放的 Web 服务时，很容易遭受资源滥用和各类攻击 [1]。这些情况不但影响原有服务质量，也常常造成经济损失。相应保护系统的设计和部署并不简单，不同方案在不同环境下表现各异，某些场景中甚至可能引入额外的性能问题和攻击面。

现有的保护方案主要分为两大类：以审计为主的离线方案和以图灵测试为主的实时方案。审计方案认为爬虫等机器流量会带有一定特征，可以进行分析、识别和拦截；图灵测试则要求访问者进行一项机器难以完成的操作，只响应完成测试的访问。

收稿日期：2018年6月29日. 本课题得到… …基金中文完整名称(No.项目号)、… …基金中文完整名称(No.项目号)、… … 基金中文完整名称(No.项目号)资助.高建，男，1994年生，硕士研究生，主要研究领域为*****、****.E-mail: 16210240009@fudan.edu.cn.白晓菲，女，1994年生，硕士研究生,主要研究领域为*****、****.E-mail:16210240001@fudan.edu.cn. 张亮，男，1962年生，博士，教授，中国计算机协会（CCF）会员,主要研究领域为*****、****.E-mail: lzhang@fudan.edu.cn

以一个具体场景为例，一家公司（为了方便讨论，将其称为 Agency）计划实现一个聚合的机票订票服务，完成业务需要频繁向多家航空公司网站请求服务。如果航空公司使用审计策略，则 Agency 很容易因为访问频次过高而被禁止访问；如果航空公司使用图灵测试策略，则需要大量完成验证工作，无论使用机器来对抗还是发起众包来消解，都极大增加系统复杂性和各类成本。

实际场景中，并非所有机器流量都产生资源滥用，尤其在有即时搭建（ad-hoc）的整合服务存在的场景中，现有方法会将这些访问一并拦截，产生显著的合作门槛。现有方法带来的复杂性、“混淆-识别”技术对抗槛等因素，也一定程度上制约了 Web 服务本身在质量和效率等方面的发展。

本文提出一类基于工作量证明（Proof-of-Work，或 PoW）的 Web 服务保护方法，作为对现有方法的补充。这一方法允许提供者对服务价值进行建模来控制保护系统，在有效防止资源滥用的前提下，适度放行机器流量，帮助服务提供者在保护强度与合作可能之间作出权衡，搭建出符合自身需求的保护系统，应对复杂的网络环境。

PoW 系统的概念最早由 Dwork 和 Naor 于 1993 年提出 [2]，要求用户进行一种耗时适当的复杂运算，但其答案能被快速验算，将耗用的时间、设备与能源做为担保成本，维持受保护资源的公平分配。通过对这一成本的量化控制，服务提供者可以精确调控保护系统介入的方式，使 PoW 系统成为 Web 服务保护的有效手段。

本文介绍此类中两种具体的方法：其一作为单纯的保护系统，要求用户在请求服务时提供 PoW 证明，确保请求的合理性，以抵御资源滥用；其二在此基础上引入 Merkle 树结构 [3]，允许 PoW 证明在服务提供者之间传递，降低复杂整合服务中各参与者（最终用户和各整合服务提供者）分别开展 PoW 工作产生的较大资源开销。

如果例子中的航空公司使用传统的两类策略，保护系统的抵御能力有一定局限。尤其当 Agency 的技术实力和计算资源规模远高于航空公司时，有效的审计方案将产生很大开销，而图灵测试方案又无法起到应有的保护作用，都有可能对己方服务质量产生不利影响。而如果使用 PoW 方案，航空公司总可以提高工作量要求，限制 Agency 的请求，优先保证服务质量。

如果航空公司使用了 PoW 策略，Agency 一开始的访问只需付出较少的工作量。随着访问频次的增加，航空公司可能会提高工作量要求，直到 Agency 因无法承受而不再增加访问频次。这时，双方已经建立了相对稳定的合作关系，且能自发随着工作量要求和接受能力的变化（如航空公司因系统负载较高而提高难度）而动态调整，增加或减少服务请求。

这类方法的技术贡献主要在于：

- 利用 PoW“难计算、易验证”的属性，服务提供者可以通过调整 PoW 工作的难度，获得杠杆效果，只消耗少量资源即可对抗资源丰富的对手；
- 简化审计系统，使其无需处理全部请求，只参与 PoW 难度调节，减少服务器端资源开销，降低成本，提升服务质量；
- 不使用图灵测试，不产生相应的“混淆-识别”的技术对抗，也不彻底禁止用户自动化调用；
- 最终服务请求方提供的 PoW 证明可以在跨组织服务组合中作为等价物，随着请求传递，在保证各方保护效果的前提下，降低门槛、促进合作。

本文在第 2 节中讨论相关工作；在第 3 节中提出单一服务提供方的问题建模、解决方法和安全性证明；在第 4 节中提出涉及到多个服务提供方协作时的问题建模、解决方法和安全性证明；最后在第 5 节对文章作出总结。

2 相关工作

一般而言，Web 服务的保护主要指各类安全手段和抵御资源滥用的策略（反垃圾、反爬虫系统 [4] 等）。前者是专门的研究领域，不在本文讨论范围内；而后者一般分为两类：以审计为主的离线方案，如日志分析、流量分析 [5]等；以图灵测试为主的实时方案，其中典型代表是验证码 [6]系统。

审计类策略通常使用 Web 后端日志、流量数据等信息，运用语法分析、模式识别、统计和神经网络等手段，识别出产生资源滥用的自动化访问，并根据来源地址或提取的特征进行过滤。此类方法已有诸多大规模应用 [7]，普遍能够将资源滥用的规模降低一个数量级以上 [4]，但系统相对较为复杂，有一定的运行时开销，也给实现和维护工作带来了一定压力。

图灵测试类策略中应用最广泛的是验证码，如CAPTCHA（Completely Automated Public Turing test to tell Computers and Humans Apart）系统[6]，可以有效区分人类和机器的访问请求。但是验证码的大批量识别机制也发展迅速，有基于机器学习的识别方案[8]和廉价的众包人工识别[9]。

3 直接应用 PoW 的方案

本节在单一服务提供者的环境中讨论 PoW 安全性的定义和具体方案：首先给出一份可行的定义；然后阐述在 Web API 请求全文上使用 PoW 的系统的基本方法，并举出对其进行重放攻击的场景；最后给出一种加盐方案，通过证明其不受重放攻击影响，证明其安全性。

需要说明的是，这里对 PoW 系统的形式化与数字货币领域使用的形式化方法略有不同，使用“价值”而非“难度”的概念来描述工作数量，更加直观，便于服务提供者完成建模和部署工作。

3.1 问题定义

服务提供者定义一个价格函数 $P_{u,t}(r)$ 表示在 t 时刻用户 u 使用请求 r 获得其服务需付出的代价，即只要访问者付出足够代价，就将其服务请求视为合理，不认定为资源滥用。这一价格 P 可以根据服务提供者持有的任何信息来进行调节，包括但不限于请求规模、频率、系统负载等。

将用户 u 在 t 时刻已付出的代价记为 $C_{u,t}$，已完成的合理服务请求记为多重集 $R_{u,t}$，t_r 为请求 r 实际发生的时间，则 PoW 保护系统的安全性可以定义如下：

定义 1.（安全性）. 单一服务提供者的 PoW 保护系统的安全性指在任何时刻 t，对所有用户 u，都有 $E(C_{u,t}) = \sum_{r\in R_{u,t}} P_{u,t_r}(r)$。

这里使用 $C_{u,t}$ 的期望是为了与密码学方面的研究工作（如[2]）对接，免除本文中重复开展概率论证明的必要。

定义 1 可以写成增量形式，即任何用户在任何时刻新发出合理的服务请求都必须新付出服务提供者指定的代价。

在开销和价格上同时忽略发起请求涉及的额外开销，只考虑 PoW 要求用户完成的工作，则安全性可进一步拆分成以下三个属性：

1) 用户无法绕过 PoW 保护系统获得服务。这一点需要在软件实现和部署上保障，不在本文讨论范围。
2) 不存在开销的期望小于具体算法声称值的方法允许用户通过 PoW 系统的检验。这一点已经在密码学上有了一定保障[2]，在数字货币领域得到了广泛验证，并不断有新的研究给出更强的保障[10]。
3) 重放安全性，即用户无法使用同一份工作通过 PoW 系统检验两次。

由此可知，本文构造的 PoW 保护系统在结构上只需保证重放安全性即可达成保护目标。重放安全性的形式化定义在下一小节讨论基本方法时给出。

3.2 基本方法

服务提供者选择一个价值函数 $V(proof)$ 估算请求方完成的工作数量，其参数称为一个 PoW 证明，在基本方法中 $proof = (r, n)$。其中 r 为请求全文，n 为请求方计算求得的解，常称为 nonce。

价值函数通常分两个阶段完成：哈希验证阶段 $Hv(proof)$ 返回一个定长二进制串 h 或（当 $proof$ 完全无效时）返回 ε 表示失败；估价阶段 $V'(h)$ 将该二进制串映射到工作数量，并对 $h = \varepsilon$ 的情况输出 0。

请求者使用相应地使用一个工作算法 $W(r, p)$ 求出一个可行的 n 使 $V(r, n)$ 大于当前时刻服务提供者要求的 $P_{u,t}(r)$。工作算法 W 枚举大量 n 的取值并运行一个 $Hs(proof)$ 过程直到其返回的 h 满足 $V'(h) > P_{u,t}(r)$。Hs 过程的大量重复构成了 PoW 的核心，其无法避免的属性已经在上一节中进行了讨论。

在大部分 PoW 算法体系中，$Hv = Hs$，但由于价值函数 V 只包含一次 Hv，其计算开销较小，使该系统在服务器一侧的资源占用保持在较低的水平。某些算法[11]还使用了比 Hs 轻量的 Hv，进一步减轻服务器验证开销。

用户在请求服务前获取价值函数 V 及价值目标 $P_{u,t}(r)$（后者也可由服务提供者公开的价格函数 P 等其他信息求出），使用（随服务提供的或自己选择的）Hs 实现完成工作，并将 $proof$ 随 HTTP 请求发送给服务器（在这一方法中实际只需额外发送 n）。

服务器计算价值函数 V 并作出对应处理：对工作数量满足要求的，提供正常的服务；对于工作数量不满足要求的，视为攻击者并交由相应系统处理（如累积到一定数量后在防火墙上屏蔽）。

对这一系统进行重放攻击非常简单，用户求得有效的 n 后只需将整个 $proof$ 发送两次，即可获得两次服务，但实际只完成了一份工作。

由于 $proof$ 和具体工作具有直接对应的关系，我们可以借助它来对重放安全性进行定义：

定义 2.（重放安全性）. 单一服务提供者的 PoW 保护系统的重放安全性指其识别曾经通过验证的 $proof$ 且拒绝其再次通过验证的能力。

3.3 加盐的方法

在基本方法中，服务器因为没有保存关于过往或未来 $proof$ 的任何信息，所以会受到重放攻击而无法保证安全性。服务器显然可以记录所有已通过验证的 $proof$ 并拒绝其再次通过，但是这种手段要实现重放安全就必须进行永久存储，每次将新收到的 $proof$ 在其中进行检查的代价会逐渐增大，使这种方法不适合长期使用。

对抗重放攻击的核心思路是让 $proof$ 中 n 以外的部分及时改变。服务提供者可以通过在请求上追加一个能够及时变化的盐来完成，即令 $proof = (r, salt, n)$，其中 $salt$ 为服务器选择的盐。这时工作算法的参数也相应改变，访问者的计算工作变为 $W((r, salt), p)$。每当 $salt$ 改变，旧的 n 无法通过验证，访问者如需再次请求服务，就需要重新运行工作。

定理 1. 加盐的 PoW 保护系统具有重放安全性，当且仅当服务器给同一个盐 $salt$ 提供不超过一次服务。

证明：必要性由 3.2 节末尾的例子可知，以下证充分性。$salt$ 改变则 $(r, salt)$ 改变，那么攻击者无法使用原有的 n 和新的 $salt$ 通过价值函数 V 验证；如果攻击者使用原有的 $salt$ 请求服务，则服务器会因为 $salt$ 失效而拒绝. 证毕.

这一方案要求用户在每次请求服务前（执行 W 过程时）都持有一个有效的盐 $salt$，看似会显著增加服务器需要处理的请求数量，但实际可以通过一些部署技巧进行消解，以下列举两种思路：1) 在每个请求（无论是否有效）的应答中附带一个新的、有效的盐；或者 2) 给盐设置存活时间。由于定理 1 是充要条件，只要在同一请求 r 能再次获取到有意义的服务（如所请求的数据发生变化）前使盐失效即可实现有效保护。

沿用第 2 节中的场景，如果航空公司使用了加盐的 PoW 保护系统，则 Agency 首次请求服务的流程如图 1 所示。

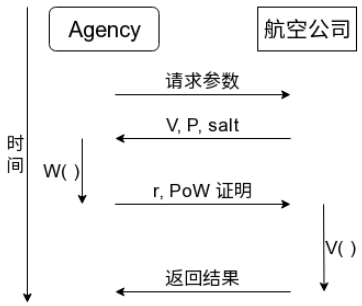

图 1 航空订票场景使用加盐 PoW 保护系统时的交互

如果航空公司在这一请求的回应中附带了一个新的 $salt$，那么 Agency 的下一次请求就无需再请求参数，可以直接执行工作过程 W 并请求服务。

4 带有 Merkle 树的方案

不难发现，无论在多服务提供者的环境中如何定义 PoW 系统的安全性，只要不与定义 1 产生冲突，加盐的方案都能够有效对 Web API 进行保护。

在有复杂的服务组合的场景中，如果参与的每一个调用者都需要进行工作过程 W 的调用来计算 n，就会造成较大的计算资源浪费。

为了消除这种资源浪费对大型系统的不利影响，需要对 PoW 保护系统进行适当的放宽。本节尝试将最终调用者提交的 PoW 证明作为组织间服务调用的等价物，降低服务提供者之间 API 调用的开销。我们首先给出该场景中安全性和重放安全性的一组可行定义，然后介绍在保护系统中使用 Merkle 树 [3]的方法并证明其具有重放安全性。

4.1 问题定义

对 3.1 节中安全性的定义进行调整，加入对服务提供者的表述，可以得到多提供者环境中安全性的一个可行定义：

定义 3.（多提供者安全性）. 多服务提供者 PoW 保护系统的安全性，指其中包含的每一个单一服务提供者的 PoW 保护系统都具有安全性。即在任何时刻 t，对所有服务提供者 sp 和所有用户 u，都有 $E(C_{sp,t}) = \sum_{r \in R_{sp,u,t}} P_{sp,u,t_r}(r)$。

相似地，也可以给出对应的安全性的定义：

定义 4.（多提供者重放安全性）. 多服务提供者 PoW 保护系统的重放安全性，指其中包含的每一个单一服务提供者的 PoW 保护系统都具有重放

安全性。即其中每个服务提供者都具有识别曾经通过验证的 $proof$ 且拒绝其再次通过验证的能力。

以上定义是可行的，由 3.1 节中服务提供者定义价格函数 P 的方式可知，单个服务提供者只要求调用者付出相应代价，并不关心系统的其他参与者，即可以允许同一个 PoW 证明被用于请求其他提供者的服务。

4.2 具体方案

取一个常用哈希算法 H，将 $(H(r,salt),H(node))$ 作为一个请求单元，则本节的系统中使用的 Merkle 树节点定义为由请求单元组成的非空集合，其中 $H(node)$ 通过哈希完成对一个子节点的引用。这一集合需要实际实现为数组以便完成哈希操作。

由于树结构上有以下公理：

公理 1. 节点 n 在树 T 中当且仅当 n 是 T 的根节点，或其父节点在 T 中。

这时服务请求者可以从 $H(r,salt)$ 所在的节点开始逐级给出父节点直到树根，记作 $[node]$，证明 $H(r,salt)$ 记录在 Merkle 树 T。三元组 $(H(r,salt),H(root(T)),[node])$ 称为Merkle证明，其中 $root(T)$ 为对应 Merkle 树的根节点。

由于 Merkle 树中已经通过哈希记录了请求和盐，上一节中的 PoW 证明无需重复记录，即令 $proof=(H(root(T)),nonce)$。使服务器先进行 Merkle 验证，则：

定理 2. 使用 Merkle 树的 PoW 系统具有多提供者重放安全性，当且仅当每个服务器对 $H(root(T))$ 相同的 Merkle 证明只接受不超过一次。

证明：必要性显然，以下证充分性。由于 $H(root(T))$ 直接出现在 $proof$ 中，攻击者无法使用原有的 n 和新的 $H(root(T))$ 通过价值函数 V 验证. 证毕.

如果 $proof$ 中的盐已经失效，就无法通过 PoW 验证。利用这一属性，服务器可以清理旧的 $H(root(T))$，无需永久记录，即：

推论 1. 服务器对每个通过 PoW 验证的请求记录根节点哈希值 $H(root(T))$，至收到该请求前发出的所有盐都已失效为止，拒绝后续使用同一 $H(root(T))$ 的 Merkle 证明，整个 PoW 保护系统具有多提供者重放安全性。

这时最终用户请求服务的过程可以描述如下：

1. 预先获取价值函数 V 和服务价格 $P_{sp,u,t_r}(r)$
2. 向服务器请求盐 $salt$ 和一个子节点 c 的引用 $H(c)$
3. 以 r、$salt$ 和 $H(c)$ 组成一个请求单元，与其他请求单元（如果有），一起构造根节点 $root(T)$
4. 调用工作算法 W 求得 n
5. 将 PoW 证明和 Merkle 证明随各个请求发送给服务器

服务器验证请求后可以根据价值函数 V 的输出，选择是否将该份证明用于事先在子节点 c 中记录的服务请求。这时只需将 c 的某个子节点 c' 加到 $[node]$ 中并以对应的请求替换掉 $H(r,s)$，就可以组成新的 Merkle 证明，用于请求预先选定的其他服务。

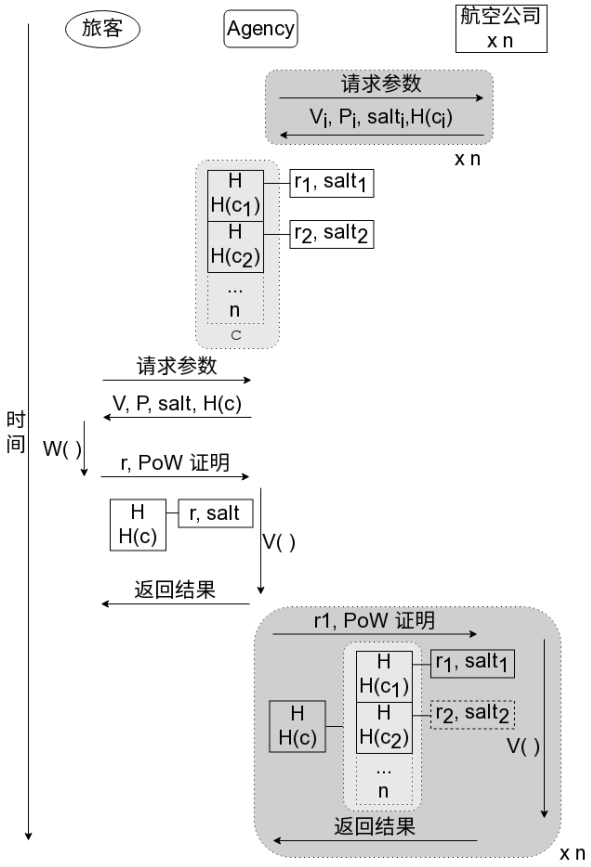

图 2 航空订票场景使用带有 Merkle 树的 PoW 保护系统时的交互

在先前的场景中，Agency 可以事先向若干航空公司请求参数，并准备好一棵包含若干请求的 Merkle 树。待最终用户（旅客）向 Agency 请求服务时，将这棵树的引用作为参数交给用户。Agency 可以进一步在用户提交的 Merkle 证明上追加节点，使对应的 PoW 证明能够用于向对应的航空公司请

求服务，如图 2 所示。

如果航空公司要求的价值低于 Agency 向用户要求的价值，那么每个 PoW 证明都能成功传递；反之，Agency 则需要筛选出价值符合航空公司要求的 PoW 证明。这一操作与数字货币矿池非常相近，从期望上来看，只要 Agency 收取的价值总和超过向每一个航空公司付出的价值，Agency 的整合服务就能够持续运转而不需要进行 PoW 计算工作。

由于树节点的引用中加了盐，这一整合服务调用过程中所有请求都只对直接相关的参与者（请求者和提供者）可见。系统的参与者还可以在哈希函数 H 的值域中生成随机数，作为节点引用填入 Merkle 树中，进一步对元数据进行保护。

与 3.3 节中相似，服务器仍然可以在应答中附带新的盐和子节点引用，但因为环境发生了较大变化，放宽重放安全性的策略可能会对系统的其他部分产生作用，是否影响安全性还有待进一步研究。

5 实现与部署

作为一个软件系统，PoW 能够投入使用需要满足功能、性能等各方面的属性，仅完成证明是不够的。本节讨论该系统在实现和部署时需要注意的问题及应对策略，供后续研发和部署工作参考。

价值函数 V 需要调用 Hv 函数，随着请求的增加，会积累显著的资源开销。为了防止在此处构造 DoS 攻击，PoW 保护系统必须与黑名单系统相连——连续多次 PoW 验证失败的用户需要被及时加入黑名单，以防其发送大量随机生成的无效 PoW 证明，大量触发验证过程，消耗服务器资源。

理想情况下，包括 PoW 在内的服务保护系统与服务本身的业务逻辑应该完全解耦，以免影响服务本身的操作和维护属性。为了实现这一目标，PoW 系统需要部署在前置的反向代理设施中。企业中出于负载均衡等需求，通常已部署 nginx[1]或 haproxy[2]等软件在此位置，则 PoW 系统的服务器端组件可以实现为相应软件的插件，并配置为优先于其他插件调用，以防缓存等功能使部分请求漏过。如第 3、4 节所述，PoW 系统需要短期存储少量数据提供重放安全性，这些数据应当存储在独立的数据库（如 redis[3]）中，以防对其他业务产生管理上或者性能上的串扰。

PoW 系统使用的算法及其参数、盐生成和管理方法、服务价值模型等要素都有很大的调整余地。服务提供者可以仅考虑保护效果，选择简易的配置（如 $Hv = Hs = H = SHA256$、$V'(h) = \frac{1}{h}$、随访问频率指数增长的服务定价 P）；也可以根据自身的具体需求仔细调整配置，提高保护效果和服务质量。其中，H 的选择对系统影响相对较小，V' 和 P 主要依靠服务提供者的建模工作选择，因而不在本文讨论范围之内；Hs 和 Hv 则直接来自于 PoW 系统，其资源开销等属性对系统的诸多指标有较大影响，我们在表 1 中列出一些常见的 PoW 算法体系及其主要特点，以便后续研究和开发工作作出合适的选择。

表 1 常见 PoW 算法及特点

算法	特点
SHA256	部署简单，有标准库 内存开销极低 有成熟的 ASIC 实现，对相应敌手效果大幅弱化
X11[4]	子算法可靠，方案成熟 内存开销较低 有 ASIC 实现，对相应敌手效果弱化
equihash[11]	有 ASIC 抗性 内存开销大 验证与工作中的单次尝试不同，即 $Hs \neq Hv$
scrypt[12][13]	有一定 ASIC 抗性 内存开销中等 验证与工作中的单次尝试不同，即 $Hs \neq Hv$

6 讨论

总体来看 PoW 系统用作 Web 服务保护是轻量且有效的，只需一个基本的键值数据库就能够有效完成工作，甚至可以无需使用持久化存储。与图灵测试系统类似，PoW 系统将一个“做与不做”的选择交给对手，但在“做”的一侧，两种系统的处理是差异很大的——验证码无法快速增加难度，使对

[1] NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy, https://www.nginx.com/

[2] HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer, https://www.haproxy.org/

[3] Redis, https://redis.io/

[4] Dash_hash/dash.c at master - dashpay/dash_hash, https://github.com/dashpay/dash_hash/blob/master/dash.c

手可以通过购买众包等方案继续完成访问，但 PoW 系统可以快速提升服务价格 ，提供更有效的保护。

在一些攻击场景中，这个“做与不做”的选择也可以协助其他系统完成一定程度的识别和防御：在较高的服务价格下，如果攻击者不完成 PoW 计算工作，会被 PoW 验证过程筛出而拒绝访问；如果攻击者继续完成 PoW 计算，又会显著拖慢攻击进程，提高攻击代价。尤其在需要多个请求才能完成攻击时，PoW 系统也能够起到辅助保护作用。

相应的，PoW 系统也存在一些已知的局限性。其一在于，特定场合（如设计上存在大规模突发访问的服务）中对服务价格函数 P 的设计比较困难，有潜在可能影响到 3.1 节中假设的可满足性；其二在于，PoW 作为密码学应用，向运行环境中引入了新的因素，如果设计、实现或部署不当，可能产生额外的攻击面。

在多提供者的环境中，定义 3 和 4 是比较严格的，会导致同一份 $proof$ 完全无法用于向同一个服务提供者请求两次服务。考虑到较为复杂的整合服务中确实可能有不同参与者各自发起这样的请求，这种属性会对服务组合产生一定的限制。我们尚不明确这种限制会对具体场景产生怎样的影响，或者是否存在更合理的安全性定义能够放松这种限制。这些疑问还需要等待后续研究工作来给出解答。

7 结论

开放公共访问的 Web 服务常常需要进行防止资源滥用的保护，而现有的保护系统未能在保护效果、服务质量和合作能力等指标上达成一致。

本文提出了一类将 PoW 系统用于公共 Web 服务保护的方法，在不削弱保护效果的条件下，降低组织间服务整合的门槛。我们对两种具体方法的有效性进行了证明，并简要讨论了实现、部署策略和已知的局限性。其中加盐的方法较为简单、易于实现且提供更好的保护效果，但在服务组合大量出现的复杂系统中可能产生资源浪费；使用 Merkle 树的方法相对复杂，在保护上进行了适当的放松，防止了这种浪费的发生，并降低了合作门槛，促进了 Web 服务资源的优化分配。

PoW 系统用于 Web 服务保护的方案尚不成熟，存在一些子问题有待后续研究，主要包括：如何对服务价值进行有效建模建模、多服务提供者环境中的安全性是否应该进一步放宽、Merkle 树方法中是否存在潜在攻击面等。

PoW 系统允许服务提供者将其服务价值模型投入到 Web 服务保护系统中，适当放行机器流量换取合作机会，给互联网服务提供者提供新的思路。

参 考 文 献

[1] Giles CL, Sun Y, Councill IG. Measuring the web crawler ethics. Proceedings of the 19th international conference on World wide web. Raleigh, USA, 2010: 1101-1102

[2] Dwork C, Naor M. Pricing via processing or combatting junk mail. Proceedings of the Annual International Cryptology Conference, Santa Barbara, USA, 1992:139-147

[3] Merkle RC. A digital signature based on a conventional encryption function. Conference on the Theory and Application of Cryptographic Techniques, Santa Barbara, USA, 1987:369-378

[4] Doran D, Gokhale SS. Web robot detection techniques: overview and limitations. Data Mining and Knowledge Discovery. 2011, 22(1-2):183-210.

[5] Geens N, Huysmans J, Vanthienen J. Evaluation of web robot discovery techniques: a benchmarking study. Industrial Conference on Data Mining, Leipzig, Germany, 2006:121-130

[6] Von Ahn L, Blum M, Hopper NJ, Langford J. CAPTCHA: Using hard AI problems for security. Proceedings of Eurocrypt. Warsaw, Poland, 2003: 294-311

[7] Park K, Pai VS, Lee KW, Calo SB. Securing Web Service by Automatic Robot Detection. Proceedings of the USENIX Annual Technical Conference, General Track, Boston, USA, 2006: 255-260

[8] Bursztein E, Aigrain J, Moscicki A, Mitchell JC. The End is Nigh: Generic Solving of Text-based CAPTCHAs. InWOOT 2014 Aug 19.

[9] Motoyama M, Levchenko K, Kanich C, McCoy D, Voelker GM, Savage S. Re: CAPTCHAs-Understanding CAPTCHA-Solving Services in an Economic Context. In USENIX Security Symposium 2010 Aug 11 (Vol. 10, p. 3).

[10] Dwork C, Goldberg A, Naor M. On memory-bound functions for fighting spam. Proceedings of the Annual International Cryptology Conference, Santa Barbara, USA, 2003: 426-444

[11] Biryukov A, Khovratovich D. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. Ledger. 2017, 2:1-30

[12] Percival, Colin, and Simon Josefsson. The scrypt password-based key derivation function. No. RFC 7914. 2016.

[13] Alwen J, Chen B, Pietrzak K, Reyzin L, Tessaro S. Scrypt is maximally memory-hard. Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 2017:33-62

投稿专题：服务质量与服务安全、隐私与信任