Embark：将中间盒安全地外包到云端

加州

大学伯克利分校的 **Chang Lan，Justine Sherry，Raluca Ada Popa 和 Sylvia Ratnasamy；刘志，*清华大学***

https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/lan

本文包含在

第 13 届 USENIX 网络系统

设计和实现研讨会（NSDI '16）的会议记录中。

2016 年 3 月 16 日至 18 日•美国加利福尼亚州圣塔克拉拉

ISBN 978-1-931971-29-4

由 USENIX 赞助的

第 13 届 USENIX

网络系统设计与

实现研讨会论文集（NSDI '16）的开放获取

Embark：将中间盒安全地外包到云端

Chang Lan Justine Sherry Raluca Ada Popa Sylvia Ratnasamy Zhi Liu *
*加州大学伯克利分校*清华大学*

抽象的

对于企业和其他组织来说，将网络处理外包到云中变得越来越普遍。例如，企业可以将防火墙，缓存和深度数据包检查等外包，就像它们将计算和存储外包一样。但是，这对企业机密性构成了威胁，因为云提供商可以访问组织的流量。

我们设计并构建了Embark，这是第一个使云提供商能够支持中间盒外包并同时维护客户机密性的系统。Embark 对到达云的流量进行加密，并使云能够处理加密的流量而无需将其解密。Embark 支持各种中间盒，例如防火墙，NAT，Web 代理，负载平衡器和数据渗透系统。我们的评估表明，Embark 以具有竞争力的性能支持这些应用程序。

1 引言

诸如防火墙，NAT 和代理之类的中间盒已成为现代网络的重要组成部分，但也被广泛认为带来了包括高成本，灵活性和复杂管理在内的重大问题。这些问题导致研究和行业探索的另一种方法：移动中间件的功能进行专门的箱子，进入运行在商用服务器硬件复用软件的应用[53 ， 52,54,29,37,28,27,14,8]。这种被业界称为网络功能虚拟化（NFV）的方法具有许多优势，包括商品基础设施和外包管理的成本优势，统计复用的效率以及软件解决方案的灵活性。在很短的时间，那非那韦已经获得了显著势头，超过 270 个行业参与者[27]和一些新兴产品种类的[1 ， 7,6]。

利用上述趋势，几项工作正在探索一种新的中间箱部署模型，其中第三方将中间箱处理作为*服务提供*。这样的服务可以在公共云被托管[54,13,17] 或在嵌入式一个 ISP 基础设施内的私有云[14 ，11]。该服务模型允许企业等客户从其网络中完全“外包”中间盒，因此承诺了云计算的许多已知优势，例如降低成本和易于管理。

但是，外包中间盒带来了新挑战语言：交通的机密性。如今，为了处理组织的流量，云将流量视为*未加密*。这意味着云现在可以访问潜在敏感的数据包有效载荷和标头。考虑到云员工或黑客记录的数据泄露数量，这令人担忧[23，60]。因此，一个重要的问题是：我们是否可以让第三方在*没有看到企业流量的情况下处理企业的流量？*

为了解决这个问题，我们设计并实现了 Embark [1]，这是第一个允许企业将大量企业中间盒外包给云提供商的系统，同时又保持其网络流量的机密性。Embark 中的中间盒直接对*加密*流量进行操作，而无需对其进行解密。

在以前的工作中，我们设计了一个称为 BlindBox 的系统，用于处理*特定*类别的中间盒的加密流量：深度数据包检查（DPI）[55]-仅检查数据包有效负载的中间盒。但是，BlindBox 不足以进行此设置，因为（1）它的功能受限制，仅支持通常外包的中间盒的数量过少；（2）在某些情况下，性能开销过高。我们在第 2.4 节中详细说明了这些要点。

Embark 支持各种具有实用性能的中间盒。表 1 显示了相关的中间盒和 Embark 提供的功能。Embark 通过结合以下系统和密码创新来实现此功能。

从加密角度来看，Embark 提供了一种称为 PrefixMatch 的新的快速加密方案，以使提供程序能够执行前缀匹配（*例如*，如果 IP 地址在子域 56.24.67.0/16 中）或端口范围检测（*例如*，如果端口在 1000-2000 之间）。PrefixMatch 允许使用与未加密数据相同的运算符将已加密数据包字段与已加密前缀或范围进行匹配：>和前缀相等。同时，比较运算符在加密的数据包字段之间使用时不起作用。在 PrefixMatch 之前，没有一种机制可以提供我们设置中所需的功能，性能和安全性。最接近的实用加密方案是顺序保留加密（OPE）[21，48]。但是，我们显示这些方案比以下方案慢四个数量级

[1] 此名称分别来自“ mb”和“ ark”（中间盒的快捷方式和保护的同义词）。

NAT（NAPT）[57]	（SIP_1，DIP_1，SP_1，DP_1，P_1）=（SIP_2，DIP_2，SP_2，DP_2，P_2） Enc（SIP_1，SP_1）= Enc（SIP_2，SP_2） Enc（SIP_1，SP_1）= Enc（SIP_2，SP_2）（SIP_1，SP_1）=（SIP_2，SP_2）	是的	前缀匹配	
L3 磅（ECMP）[5				
L4 磅[4]	HTTP 代理/缓存[25，4，10]	Match（Request-URI，HTTP Header） =匹配（Enc（Request-URI），Enc（HTTPHeader））| 是的	关键字匹配	方案
	家长过滤器[10]	Match（Request-URI，HTTP Header） =匹配（Enc（Request-URI），Enc（HTTPHeader））| 是的	关键字匹配	前缀匹配
数据渗漏/水印检测[56]	匹配（水印，流） =匹配（Enc（水印），Enc（流））	是的	关键字匹配	
入侵 检测[59，47]	匹配（关键字，流）= 匹配（Enc（关键字），Enc（流））	是的	关键字匹配	
	RegExpMatch（RegExp，Stream） = RegExpMatch（Enc（RegExp），Enc（流））	部分地	关键字匹配	
	运行脚本， 错流分析或其他高级（例如统计）工具	不	--	

表 1：Embark 支持的中间盒。第二列表示足以支持中间盒核心功能的加密功能。附录§A 证明了这一点。“支持”表示 Embark 是否支持此功能，“方案”是 Embark 用于支持该功能的加密方案。**图例：** Enc 表示通用加密协议，***SIP*** =源 IP 地址，***DIP*** =目的 IP，***SP*** =源端口，***DP*** =目的端口，***P*** =协议，***E*** []=范围 ***E***，表示“仅当且仅当”，匹配（***X***，***5***）表示，如果 ***X*** 是的子串***小号***，和匹配是加密等效的匹配。因此，（***SIP***，***DIP***，***SP***，***DP***，***P***）表示描述连接的元组。

PrefixMatch 使它们对于我们的网络设置不可行。同时，PrefixMatch 提供了比这些方案更强的安全性保证：PrefixMatch 不显示加密数据包字段的顺序，而 OPE 显示所有字段之间的总顺序我们专门针对 Embark 的网络设置设计了 PrefixMatch，从而实现了 OPE 方面的改进。

从系统设计的角度来看，Embark 的主要见解之一是保持数据包格式和标头分类算法不变。加密的 IP 数据包的结构与普通 IP 数据包一样，每个字段（例如源地址）都包含该字段的加密值。此策略可确保加密数据包永远不会对现有网络接口，转发算法和错误检查显示为无效。此外，由于 PrefixMatch 的功能，基于标头的中间盒无需修改即可运行现有的高效数据包分类算法[34]，这是软件中间盒[52]中较昂贵的任务之一。此外，即使基于软件部署 NFV 使用某些硬件转发组件，*例如* NIC 多队列流散列[5]，“白盒”开关[12]，以及在 NIC 和交换机错误检测[5，2]; Embark 也与此兼容。

Embark 的统一策略是将相关中间盒的核心功能简化为两个基本歌剧

数据包不同字段上的操作：前缀和关键字匹配，如表 1 所示。这将导致*同时*支持这些中间盒的加密数据包。

我们实施并评估了 Embark on EC2。Embark 支持表 1 中 列出并在附录 A 中详细介绍的各种中间盒的核心功能。在我们的评估中我们表明 Embark 支持表 1 中每个中间盒类别的真实示例。此外，Embark 的吞吐量可忽略不计服务提供商的开销：例如，在加密数据上运行的单核防火墙达到 9.8Gbps，相当于在未加密数据上运行的同一防火墙。我们的企业网关可以在单个内核上以 9.6 Gbps 的速率传输流量。一台服务器可以轻松为中小型企业支持 10Gbps。

2 概述

在本节中，我们将概述 Embark。

2.1 系统架构

Embark 使用与 APLOMB [54]相同的架构，该系统将企业的流量重定向到云以进行中间盒处理。Embark 通过机密性保护增强了该体系结构。

在 APLOMB 设置中，有两个方面：企业和服务提供商或云（SP）。企业运行网关（GW），该网关将流量发送到云中运行的中间盒（MB）。实际上，此云可以是公共云服务（例如 EC2），也可以是在中央办公室（CO）运行的 ISP 支持的服务。

我们在图 1 中说明了来自 APLOMB 的两种重定向设置。图 1（a）中的第一种设置是在企业与外部站点进行通信时发生的：流量先流向云，然后再流回云，然后再发送到 Internet。 。值得一提的是，相对于图 1（a），APLOMB 允许进行节省带宽和延迟的优化：来自 SP 的流量可以直接进入外部站点，而不必通过网关返回 Embark 从根本上不允许这种优化：来自 SP 的流量已加密，并且外部站点无法理解。尽管如此，我们证明§ 6 ，对于基于 ISP 的部署，此开销可以忽略不计。对于同一企业内的流量，其中同一公司拥有的两个网关都知道密钥，我们可以支持如图 1（b）所示的优化。

我们不会进一步研究 APLOMB 设置的细节和动机，而是请读者参考 [54]。

2.2 威胁模型

客户采用云服务可降低成本并简化管理。提供者是众所周知的，可以提供良好的服务。但是，尽管客户信任云提供商可以正确执行其服务，但人们越来越担心云提供商会在提供服务的过程中访问或泄露机密数据。

在大众媒体的报道描述公司出售客户资料给营销[20]，心怀不满的员工窥探或导出数据[16]，而黑客获得对数据的访问云上[60 ，23]。这种类型的威胁被称为“诚实但又好奇”或“被动”的攻击者[33] ：可以信任以处理数据并正确提供服务，但查看数据并窃取或导出数据的一方。Embark 旨在阻止这些攻击者。这种攻击者不同于“主动”攻击者，后者会操纵数据或偏离应该运行的协议[33]我们认为，这样的被动攻击者已经可以访问 *SP 中的所有数据*。这包括 SP 从网关接收的任何流量和通信，任何记录的信息，云状态等等。

我们假设网关是由企业管理的，因此是受信任的。他们不会泄漏信息。

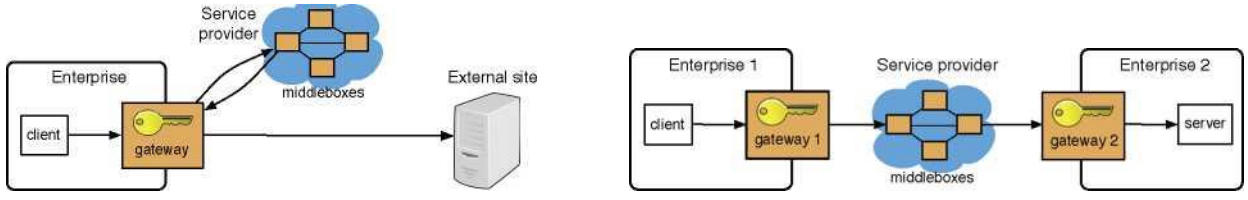
一些中间盒（例如入侵或渗透检测）对于两个端点之间的通信具有自己的威胁模型。例如，入侵检测假设端点之一可能行为不当，但其中最多一个行为不当 [47]。我们保留这些威胁模型不变这些应用程序依靠中间盒来检测这些威胁模型中的攻击。由于我们假设中间盒能够正确执行其功能，而 Embark 保留了这些中间盒的功能，因此这些威胁模型与 Embark 中的协议无关，因此我们不再赘述。

2.3 加密概述

为了保护隐私，Embark *对*通过服务提供商（SP）*的流量*进行*加密*。Embark 会加密每个数据包的标头和有效负载，因此 SP 不会看到此信息。我们对标头进行加密，因为它们包含有关端点的信息。

Embark 还为云提供商提供了一组*加密规则*。通常，诸如防火墙规则之类的标头策略是由本地网络管理员生成的。因此，网关知道这些规则，并且这些规则可能会或可能不会从云中隐藏。另一方面，DPI 和过滤策略可能是企业专用的（如在渗透策略中），双方都知道（如在公共黑名单中）或仅由云提供商知道（如在专有的恶意软件签名中）。在第 4.2 节中，我们将在给定这些不同的信任设置的情况下讨论如何加密，生成和分发规则。

如图 1 所示，网关具有密钥 k；在具有两个网关的设置中，它们共享相同的密钥。在建立时，网关使用 k 生成一组加密规则，并将其提供给 SP。之后，网关使用 Embark 的加密方案对进入服务提供商的所有流量进行加密。SP 的中间盒处理加密的流量，将流量与加密的规则进行比较。处理后，中间盒将产生加密的流量 SP 将其发送回给服务器。



（a）企业与外部站点的沟通　　（b）企业对企业的沟通

图 1：系统架构。在网关处使用 Embark 加密的 APLOMB 和 NFV 系统设置。箭头指示从客户端到服务器的流量。响应流量遵循相反的方向。

网关。网关使用密钥 *k* 解密流量。

在整个过程中，SP 的中间盒仅处理加密的流量，而从不访问解密密钥。除了 Embark 的加密功能外，网关还可以使用安全隧道协议（例如 SSL 或 IPSec）来保护与 SP 的通信。

数据包加密。一个关键思想是*逐字段*加密数据包。例如，一个加密的数据包将包含一个源地址，该地址是对原始数据包的源地址的加密。我们确保加密的大小与原始数据相同，并将任何其他加密的信息或元数据放在数据包的选项字段中。Embark 使用三种加密方案来保护每个字段的隐私，同时允许与云中的加密规则进行比较：

- 传统 AES：提供强大的安全性，没有计算功能。
- KeywordMatch：允许提供者检测数据包中的加密值是否等于加密规则；不允许将两个加密值相互比较。
- PrefixMatch：允许提供程序检测加密值是否在规则值范围内，例如 128.0.0.0/24 中的地址或 80-96 之间的端口。

我们在讨论这些加密算法§3 。

例如，我们使用 PrefixMatch 加密 IP 地址。例如，这允许防火墙检查数据包的源 IP 是否属于已知由僵尸网络控制的前缀-但无需了解实际的源 IP 地址是什么。根据表 1 所示的中间盒功能分类，我们选择适合每个字段的加密方案。在同一表中，我们将中间盒分类为仅在 L3 / L4 标头上运行，仅在 L3 / L4 标头和 HTTP 标头上运行，或对整个数据包进行操作，包括连接字节流（DPI）中的任意字段。我们在重温详细说明各个主题§5 。

所有加密的数据包都是 IPv6，因为 PrefixMatch 需要超过 32 位才能对加密的 IP 地址进行编码，并且由于我们希望将来越来越多的服务提供商默认使用 IPv6。这是一个微不足道的要求，因为在网关处很容易从 IPv4 转换为 IPv6（并反向转换）[42] 。客户群可能会继续使用 IPv4，并且将网关连接到提供商的隧道可能是 v4 或 v6。

例子。图 2 显示了通过云中三个示例中间盒的数据包的端到端流每个中间盒在一个加密字段上运行。假设初始数据包是 IPv4。首先，网关将数据包从 IPv4 转换为 IPv6 并对其进行加密。现在，options 字段包含一些辅助信息，这些信息将帮助网关以后解密数据包。数据包通过防火墙，防火墙尝试将标头中的加密信息与其加密规则进行匹配，并决定允许该数据包。接下来，渗透设备检查为 DPI 加密的数据中是否存在任何可疑（加密）字符串，并且没有发现任何字符串，它允许数据包继续进行 NAT。NAT 将源 IP 地址映射到其他 IP 地址。回到企业，网关对数据包进行解密，NAT 编写的源 IP 除外。它将数据包转换回 IPv4。

2.4　架构含义和与 BlindBox 的比较

与 BlindBox 相比，Embark 提供了更广泛的功能和更好的性能。关于功能，BlindBox [55] 在包的加密有效载荷上启用基于相等性的操作，该操作支持某些 DPI 设备。但是，这不包括诸如防火墙，代理，负载平衡器，NAT 之类的中间盒，以及还检查包头的那些 DPI 设备，因为它们需要与包头兼容的加密和/或需要执行范围查询或前缀匹配。

性能的提高来自 Embark 不同的体系结构设置，这提供了一系列有趣的机会。在 BlindBox 中，两个任意用户端点通过 HTTPS 的修改版本进行通信。BlindBox 需要 97 秒才能执行初始握手，必须为每个新连接执行此握手。但是，在 Embark 环境中，由于网关与云提供商之间的连接是长期存在的，因此只能在网关处执行一次此交换。因此，不存在每个用户连接的开销。

第二个好处是增加了可部署性。在 Embark 中，网关加密流量而在 BlindBox 中

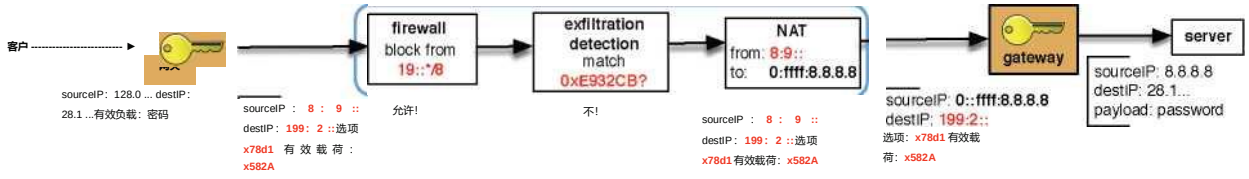


图 2：通过一些中间盒的数据包流示例。粗体红色表示加密的数据。

最终主机会这样做。因此，由于不需要修改最终主机，因此可部署性得到了改善。

最后，安全性可以通过以下方式提高。BlindBox 有两种安全模型：一种较强的模型用于检测“完全匹配”子字符串的规则，另一种较弱的模型用于检测作为正则表达式的规则。规则越多，每个连接的建立成本就越高。由于 Embark 中没有每个连接的开销，因此我们可以负担更多的规则。因此，我们将许多正则表达式转换为一组完全匹配的字符串。例如，/ hello [1-3] /等效于“ hellol”，“ hello2”，“ hello3”上的精确匹配项。尽管如此，许多正则表达式仍然过于复杂-如果潜在的精确匹配集太大，我们将其保留为正则表达式。正如我们在展示§6，这种方法将使用较弱的安全模型所需的规则数量减半，从而在较强的安全模型中启用了更多规则。

在本文的其余部分，我们不探讨这些架构优势，而是着眼于踏上新的能力，使我们能够外包一个*完整的*一套中间件的。

2.5 安全保证

我们在扩展的文件中形式化并证明 Embark 的总体保证。在此版本中，我们仅提供高级描述。Embark 隐藏标头和有效负载数据的值但显示中间盒处理所需的一些信息。信息披露是信息的工会揭示 PrefixMatch 和关键字匹配，如详细§3 。Embark 揭示了该功能所不仅仅具有的必要功能，但它接近于此必要功能。例如，防火墙会在不了解 IP 地址或前缀的值的情况下，了解加密的 IP 地址是否与加密的前缀匹配。DPI 中间盒了解某个字节偏移量是否与 DPI 规则集中的任何字符串匹配。

3 密码构建块

在本节中，我们介绍 Embark 所依赖的构建基块。对称密钥加密（基于 AES）是众所周知的，因此在此不进行讨论。取而代之的是，我们简要讨论 KeywordMatch（由[55]引入，我们向读者介绍有关细节），并更广泛地讨论 PrefixMatch，这是我们为此设置设计的一种新的加密方案。在描述这些方案时，我们将加密器称为网关密钥为 k 的实体，并以服务提供者（SP）的身份对加密数据进行计算。

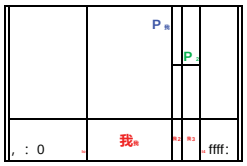
3.1 关键字匹配

KeywordMatch 是一种加密方案，SP 可以使用该方案检查加密规则（“关键字”）是否相等地匹配加密字符串。例如，给定规则“ malicious”的加密以及加密字符串的列表[Enc （“ alice”）， Enc （“ malicious”），Enc（“alice”）]，SP 可以检测到该规则与第二个字符串，但是它对第一个和第三个字符串一无所知，即使它们彼此相等也是如此。KeywordMatch 提供了典型的可搜索安全性保证并对此进行了深入研究：在较高级别上，给定了加密字符串的列表和加密关键字，SP 不会了解有关加密字符串的任何信息，除非哪些字符串与关键字匹配。字符串的加密是*随机化*，因此不会泄漏两个加密字符串是否彼此相等，除非它们当然都与加密关键字匹配。我们使用[55]中的方案，因此不再赘述。

3.2 PrefixMatch

许多中间盒在 IP 地址或端口号的*前缀*或*范围内*执行检测（即，数据包分类）。为了说明 PrefixMatch，我们使用 IP 地址（IPv6），但该方案也适用于端口和其他值域。例如，网络管理员可能希望阻止访问由 MIT 托管的所有服务器，在这种情况下，管理员将阻止访问前缀 0 :: ffff：18.0.0.0/104，*即* 0 :: ffff：18.0.0.0 /104-0::ffff:18.255.255.255/104。PrefixMatch 使中间盒能够判断加密的 IP 地址 v 是否在加密范围[s_1,e_1]，其中 s_1 = 0 :: ffff：18.0.0.0/104 和 e_1 = 0 :: ffff：18.255.255.255/104。同时，中间盒不学习 v，s_1 或 e_1 的值。

有人可能会问，PrefixMatch 是否必要，或者可以使用与第 2.4 节中的某些（但不是全部）正则表达式相同的扩展技术来使用 KeywordMatch 。要检测某个 IP 地址是否在范围内，可以枚举该范围内的所有 IP 地址并执行相等性检查。但是，将这种技术用于诸如防火墙规则之类的常见网络范围的开销是过高的。对于我们自己的部门网络，这样做会将仅 97 个基于范围的规则的 IPv6 和 IPv4 防火墙规则集转换为



2 [238]精确匹配规则; 仅查看 IPv4 规则仍将导致 38M 完全匹配规则。因此，为了提高效率，我们需要一种新的范围匹配方案。

要求。要支持表 1 中的中间盒并满足我们的系统安全性和性能要求，在设计 PrefixMatch 时需要满足以下要求。首先，PrefixMatch 必须允许在加密值 Enc （*v*） 与范围[*s* i ， *e* 1]的加密端点 W 和 *e* f 之间进行直接顺序比较（即，使用</>）。这允许使用现有的包分类算法，例如尝试，基于区域的四叉树，FIS 树或基于硬件的算法[34]，运行不变。

其次，为了支持如表 1 所示的 NAT 功能， Enc （*v*） 必须*在流中具有确定性*。回想一下，流是源 IP 和端口，目标 IP 和端口以及协议的 5 元组。此外，与两对（IP 1，端口 1）和（IP 2，端口 2）相对应的加密必须是可插入式的：如果两对不同，则其加密也应该不同。

第三，为了安全起见，我们要求除上述功能所需的值外，不要泄漏值 *v*。请注意，踏上的中间件并不需要知道两个加密值之间的顺序的 Enc （*v* 我）和的 Enc （*v* 2），但仅比较端点; 因此，PrefixMatch 不会泄漏此类订单信息。PrefixMatch 还为范围的端点提供保护：SP 不应了解其值，SP 不应了解间隔的顺序。此外，请注意，NAT 不需要 Enc （*v*） 确定性跨流程；因此，PrefixMatch 隐藏作为不同流一部分加密的两个 IP 地址是否相等。换句话说，PrefixMatch 在流之间是随机的。

最后，对于典型的中间盒线速，加密（在网关上执行）和检测（在中间盒上执行）都应该是实用的。我们的 PrefixMatch 在<0 中加密。5 *p* 每人 S 值（如我们在讨论§6），以及检测是相同的基于正则中间件</>运算符。

功能性。PrefixMatch 将一组范围或前缀 *P* 1，...，*P* n加密为一组加密的前缀。前缀 *Pi* 的加密 由一个或多个加密的前缀 *P*，1...，*P* ifIi 组成. 此外，PrefixMatch 将值 *v* 加密为加密值 Enc （*v*） 。这些加密具有这样的性质，对于所有的*我*，

v G *P* i Enc （*v*） G *P*，1 U- *U P* ^。

换句话说，加密保留前缀匹配。

例如，假设加密 *P* = 0 :: ffff：18.0.0.0/104 会生成一个加密的前缀 *P* = 1234 :: / 16，加密 *v* 1 = 0 :: ffff：18.0.0.2 会导致 *v* T = 1234：db80：85a3：0：0：8a2e：37a0：7334，并且加密 *v* 2 = 0 :: ffff：19.0.0.1 导致 *v* 2 = dc2a：108f：1e16：992e：a53b：43a3：00bb：d2c2。我们可以看到 *7* 吗? ģ *P* 和 *7* 2ģ *P*。

图 3：使用 PrefixMatch 进行前缀加密的示例。

3.2.1 方案

PrefixMatch 由两种算法组成：EncryptPrefixes 用于加密前缀/范围 EncryptValue 用于加密值 *v*。

前缀的加密。PrefixMatch 将一组前缀或范围 *P* 1 = [*s* 1， *e* 1] ，...， *P* n = [*s* n， *e* n *]作为输入*，其端点的大小为 len 位。PrefixMatch 将每个前缀加密为一组加密的前缀：这些前缀的长度为_ len 个前缀。正如我们在下面讨论的选择前缀_ LEN 取决于进行加密前缀的最大数量。例如，前缀_ len = 16 足以满足典型的防火墙规则集。

考虑所有端点 *小号*我 和 *Ë* 我递增的顺序如在图布置在轴上。3.添加在该轴线上的端点 *P* 0，最小和最大的可能的值，0 和 2 len 个- 1. 考虑所有每个连续对的端点形成的非重叠间隔。每个间隔都具有以下属性：该间隔中的所有点都属于同一组前缀。例如，在图 3 中，有两个要加密的前缀：*P* 1 和 *P* 2 。PrefixMatch 计算间隔 *I* 0，... *I* 4。正好在一个端点上重叠的两个或多个前缀/范围定义了一个元素间隔。例如，考虑对这两个范围[13 :: / 16,25 :: / 16]和[25 :: / 16,27 :: / 16]进行加密；他们定义了三个间隔：[13 :: / 16，25 :: / 16-1]，[25 :: / 16、25 :: / 16]，[25 :: / 16 + 1,27 :: / 16] 。

每个间隔都属于一组前缀。让前缀(*I*)表示间隔 *I* 的前缀。例如，前缀 （*l* 2） = { *P* 0 *P* 1， *P* 2 } 。

现在，PrefixMatch 为每个时间间隔分配一个加密的前缀。加密的前缀是一个简单的*随机* 大小的数前缀_ len 个。除了属于相同前缀的间隔之外，每个间隔都会获得不同的随机值。例如，在图 3 中因为前缀 （*I* 0） = 前缀 （*I* 4） ，所以间隔 *I* 0 和 *I* 4 接收相同的随机数。

当一个前缀与另一个前缀部分重叠时，它将具有多个加密的前缀，因为它被分成多个间隔。例如，为 *I* 1 分配了一个随机数 0x123c，为 *I* 2 分配了 0xabcc。图 3 中 *P* 1 的加密将是该对（123 *c* :: / 16，*abcc* :: / 16）。

由于加密是随机前缀，因此加密不会显示原始前缀。此外，与相同前缀集相关的间隔接收到相同的加密数字的事实隐藏了一个加密值匹配的地方，如下所述。例如，对于与 *P* 1 或 *P* 2 不匹配的 IP 地址 *v*，云

因为 I_0 和 I_4 接收相同的加密，所以提供程序将不会知道它与 $P_1 \cup P_2$ 的左边还是右边匹配。它得知的唯一信息 v 是 v 不匹配或者 P_1 或 P_2。

现在，我们介绍 EncryptPrefixes 过程，该过程对于前缀或范围是相同的。

EncryptPrefixes（P_1，…，P_n，前缀_len，len）：

1：令 s_i 和 e 为 p 的端点。 //$p = [s_i, e_i]$

2：分配 P_0 [0，$2^{LEN}-1$]

3：按升序对 UiP_i 中的所有端点进行排序

4：如上所述，从端点构造非重叠间隔 I_0，…，I_m。对于每一个间隔 A，计算前缀^-），前缀列表 $P_{我_1}$，。。。$P_{我_k}$ 包含*我我*。

5：让 I_0，…，I_m 分别是大小前缀_len 的不同随机值。

6：对于所有 I，J 与 *我*<Ĵ 如果前缀^-）=前缀（*我 J*），设定*我 J 我我*

7：P_i 的加密为 P_i

{*我 J* /前缀_len 个，forall 的 Ĵ ST $P_{我}$ġ 前缀（*我 J*）}。加密的前缀按值排序输出（作为随机化的一种方式）。

8：输出 P_1，…，P_n 和*间隔图*[$I_i I_i$]

价值加密。要加密的值 v，PrefixMatch 所处的一个时间间隔 *予*，使得 v ġ *予*。然后，它在由 EncryptPrefixes 计算的间隔图中查找 I 并将 I 设置为 v 加密的前缀。这确保了加密的 v 和 v 匹配 I /前缀_len。v 的后缀是随机选择的。唯一的要求是它是确定性的。因此，后缀是根据伪随机函数[32]选择的，即 prf 后缀_LEN，在一个给定的种子播种种子，其中后缀_LEN = LEN -前缀_len 个。如下所述，网关使用的种子取决于连接的 5 元组（SIP，SP，DIP，DP，P）。

例如，如果 v 为 0 :: ffff：127.0.0.1，并且为匹配间隔分配的前缀为 *abcd* :: / 16，则在上述加密范围内，可能的加密为 Enc（v）= *abcd*：*ef01*：2345：6789：*abcd*：*ef01*：2345：6789。请注意，加密不会保留除 v 匹配的间隔以外的任何有关 v *的*信息，因为后缀是随机选择（伪）的。特别是给定两个值 v_1 和 v_2 匹配相同间隔的加密顺序是任意的。因此，PrefixMatch 不会显示顺序。

EncryptValue（seed，v，后缀_len，间隔图）：

1：运行二进制搜索间隔地图上定位间隔*我*使得 v ġ *予*。

2：在间隔图中查找 I。

3：输出

Enc（v）= *7* 小时 prf SUe 'ci ^{xJen}（v） （1）

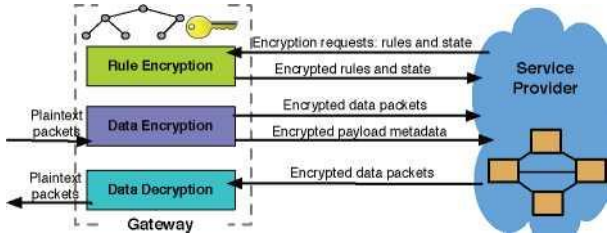


图 4：云与网关服务之间的通信：规则加密，数据加密和数据解密。

将加密值与规则进行比较。确定加密值是否匹配加密前缀很简单加密会保留前缀，并且中间盒可以使用常规的<//>运算符。因此，可以在防火墙上运行常规的数据包分类，而无需进行任何修改。比较匹配相同前缀的不同加密值是没有意义的，并返回一个随机值。

3.2.2 安全保证

PrefixMatch 隐藏使用 EncryptPrefixes 和 EncryptValue 加密的前缀和值。PrefixMatch 揭示了在云提供商处启用功能所需的匹配信息具体而言，云提供商可以学习间隔的数量以及每个间隔中哪些前缀重叠，但是没有有关这些间隔的大小，顺序或端点的其他信息而且，对于每个加密的值 v，它都学习包含 v 的前缀的索引（这是该方案所需的功能），但是没有其他有关 v *的*信息。对于任何两个加密值，Enc（v）和 Enc（v'），云提供商仅在将它们作为同一流的一部分进行加密时才知道它们是否相等（这是 NAT 所需的功能），但它不会学习有关其价值或顺序的任何其他信息。因此，与保留顺序的加密相比，PrefixMatch 泄漏的信息更少，从而揭示了加密前缀/范围的顺序。

由于 EncryptValue 植入每个连接标识符中，因此攻击者无法跨流关联值。本质上，每个流有一个不同的密钥。特别是，即使 EncryptValue 在流中是确定性的，也可以在流之间随机分配：例如，由于每个流的种子不同，因此在不同流中对相同 IP 地址的加密是不同的。

我们在扩展的文件中形式化并证明 PrefixMatch 的安全性保证。

4 企业网关

网关有两个作用。首先，它将流量重定向到云/从云重定向到中间盒以进行处理。其次，它为云提供了规则集的加密。每个网关都静态配置为在单个服务提供商存在点将流量隧道传输到固定 IP 地址。从逻辑上讲，网关可以视为三种服务：

规则加密服务，从企业到云的管道（数据加密）以及从云到企业的管道（数据解密）。所有这三个服务共享对 PrefixMatch 间隔图和私钥 *k 的访问*。 图 4 说明了这三种服务以及它们向云提供商发送的数据以及从云提供商发送的数据。

我们在设计网关时要牢记两个目标：**格式兼容性**：在将纯文本流量转换为加密流量时，加密数据的结构应使该*流量*对执行处理的中间盒而言*像普通 IPv6 流量一样出现*。格式兼容性使我们不仅可以在中间盒软件中而且在诸如 NIC 和交换机之类的硬件组件中保持快速路径操作不变。这样可以在云上获得良好的性能。

可伸缩性和低复杂度：网关应仅执行便宜的每数据包操作，并且应可并行化。网关只需要少量的配置。

4.1 数据加密和解密

如表 1 所示，我们将中间盒分类为 Header 中间盒，它们仅在 IP 和传输头上运行；DPI 中间盒，可在连接字节流中的任意字段上运行；和 HTTP 中间盒，它们对 HTTP 标头中的值进行操作（这些是 DPI 中间盒的子类）。我们讨论如何加密/解密各种数据，以满足中间盒的要求，如下所示。

4.1.1 IP 和传输头

使用 PrefixMatch 逐字段对 IP 和传输头进行加密（*例如*，输入数据包中的源地址导致输出数据包中的加密源地址字段）。我们将 PrefixMatch 用于这些字段，因为许多中间盒对前缀和值的范围进行分析-例如，防火墙可能会阻止来自受限制 IP 前缀的所有连接。

加密与 PrefixMatch 的 Encrypt-值的值，所述网关种子与加密种子 $= \mathrm{PRF}_k$ (*SIP，SP，DIP，DP，P*)，两者使用符号表的密钥和连接信息的功能 1.注意在具有两个网关的系统设置中，网关生成相同的加密，因为它们共享 *k*。

加密 IP 地址时，两个不同的 IP 地址一定不能映射到相同的加密，因为这会破坏 NAT。为避免此问题，Embark 中的加密 IP 地址必须为 IPv6，因为将两个 IP 地址分配给同一加密的可能性很小原因是每个加密的前缀都包含大量可能的 IP 地址。假设我们有 *n 个*不同的防火墙规则，*m 个*流和 len 位空间，则发生冲突的可能性大约为：

$$\frac{-米^{2}(2\tilde{N}+1)}{1-e^{2^{|\mathrm{en}}+1}}\quad(2)$$

因此，如果 len = 128（在使用 IPv6 时就是这种情况），那么在实际情况下该概率可以忽略不计。

加密端口时，由于 port 字段仅为 16 位，因此可能会发生冲突。但是，只要 IP 地址不冲突，就不会破坏 NAT 的功能，因为 NAT（以及其他需要插入的中间盒）同时考虑了 IP 地址和端口。例如如果我们有两个流的源 IP 和源端口分别为 *(SIP，SP* $_1$) 和 *(SIP，SP* $_2$*)* 且 *SP* $_1$ = *SP* $_2$，则两个流中 SIP 的加密将有所不同，因为加密是播种在连接的 5 元组中。正如我们在附录 A 中讨论的那样，对于 Embark，NAT 表可以更大，但实际上因素很小。

解密。PrefixMatch 是不可逆的。为了启用数据包解密，我们将标头字段的 AES 加密值存储在 IPv6 选项标头中。网关收到要解密的数据包时，如果中间盒（*例如* NAT）尚未重写这些值，则它将从选项标头中解密这些值并还原它们。

格式兼容性。我们对 IP 和传输标头的修改将加密的前缀匹配数据放回与原始存储的未加密数据相同的字段中；因为规则和加密数据之间的比较依赖于<>，就像未加密的数据一样，这意味着*在中间盒*上对 IP 和传输头进行比较的操作*完全保持不变*。这确保了与现有软件*和硬件* 快速路径操作的向后兼容性。由于在生产中间盒中对每个数据包的操作进行了严格的优化，因此即使我们进行了更改，这种兼容性也可以确保在云上获得良好的性能。

格式兼容性的另一个挑战是可解密 AES 数据的放置位置。一种选择是定义我们自己的数据包格式，但这可能会导致与现有实现不兼容。通过将其放在 IPv6 选项标头中，可以配置中间盒以忽略此数据。[2]

4.1.2 有效载荷数据

连接字节流使用 KeywordMatch 加密。与 PrefixMatch 不同，所有流中的数据都使用相同的密钥 *k* 加密。原因是 KeywordMatch 是随机的，并且不会泄漏跨流的相等模式。

这使 Embark 支持 DPI 中间盒，例如入侵检测或防止渗透。这些设备必须检测是否存在

[2] 一个常见的误解是中间盒与 IP 选项不兼容。商业中间盒通常知道 IP 选项，但是许多管理员 将设备*配置为*在启用某些类型的选项的情况下过滤或丢弃数据包。

连接字节流中*任意位置*的加密规则字符串的完全匹配。由于此加密的有效负载数据是通过*字节流*传输的，*因此*我们需要生成跨越“包”有效负载之间的加密值。我们用于加密 DPI 的可搜索加密方案要求对流量进行*标记化*， 并沿着滑动窗口对一组固定长度的流量子串进行加密-例如，恶意词可能会被标记为“ malici”，“ alicio”“ liciou”，“ icious”。如果将“恶意”一词划分为两个数据包，除非我们在网关处重建 TCP 字节流，否则我们可能无法正确地对其进行标记。因此，如果在云上启用了 DPI，我们将完全这样做。

在重建并加密了 TCP 字节流之后，网关通过“扩展”辅助通道传输加密的字节流，该通道仅执行 DPI 操作的那些中间盒进行检查此通道未路由到其他中间盒。我们将此通道实现为网关和中间盒之间的持久 TCP 连接。传输中的字节流与其流标识符相关联，以便 DPI 中间盒可以区分不同流中的字节流。DPI 中间盒处理从扩展通道以及包含数据包的主通道接收到的数据包；我们在[55]中详细阐述了这种机制。 因此，如果入侵防御系统在扩展通道中找到签名，则它可以切断或重置主通道的连接。

解密。有效负载数据已使用 AES 加密并放回数据包有效负载中-与 PrefixMatch，KeywordMatch 一样不可逆，我们需要此数据在网关进行解密。因为扩展通道对于解密不是必需的，所以它不会传输回网关。

格式兼容性。对于仅检查/修改数据包头的中间盒，加密有效负载没有影响。通过将加密的字节流放在扩展通道中，多余的流量可以被路由通过，并被不需要此数据的中间盒忽略。

如[55]中所述，用于检查有效载荷的 DPI 中间盒必须进行修改以检查主通道旁的扩展通道。DPI 设备一般用软件实现，这些修改都是直接和增加开销的限制（正如我们在看到§6 ）。

4.1.3 HTTP 标头

HTTP 标头是有效负载数据的特例。诸如 Web 代理之类的中间盒不会从数据包有效载荷中读取任意值：它们读取的唯一值是 HTTP 标头。由于它们需要检查 TCP 字节组，因此可以将它们归类为 DPI 中间盒。但是，由于完整 DPI 的限制对于支持，我们将这些值与其他有效负载数据进行了特殊处理：我们使用确定性形式的 KeywordMatch 加密整个（未令牌化的）HTTP URI。

普通关键字匹配允许比较加密值和规则，但不能比较一个值和另一个值；确定性 KeywordMatch 也允许比较两个值。尽管相对于 KeywordMatch 而言，这是较弱的安全保证，但是必须支持需要对不同 URI 之间进行比较的 Web 缓存。因此，高速缓存了解不同 URI 的频率，但无法立即了解 URI 值。这是我们在较弱设置中加密的唯一字段。我们将此加密值放在扩展通道中。因此，我们的 HTTP 加密具有与其他 DPI 设备相同的格式兼容性属性。

像其他 DPI 任务一样，这需要解析整个 TCP 字节流。但是，在某些情况下，我们可以无状态提取和存储 HTTP 标头。只要禁用 HTTP 流水线并且数据包 MTU 为标准大小（> 1KB），则必填字段将始终连续出现在单个数据包中。鉴于 SPDY 使用持久连接和流水线请求，因此这种无状态方法不适用于 SPDY。

解密。使用存储在有效负载中的数据按正常方式解密数据包；IP 选项已删除。

4.2 规则加密

给定用于中间盒类型的规则集，网关将使用 KeywordMatch 或 PrefixMatch 对此规则集进行加密，具体取决于该中间盒所使用的加密方案，如表 1 所示。例如，防火墙规则使用 PrefixMatch 进行加密。由于加密因此，一些规则集拓展，我们在评估§ 6 多少。例如，包含使用 PrefixMatch 映射到两个加密前缀的 IP 前缀的防火墙规则将变为两个规则，每个加密前缀一个。网关应适当地生成规则，以解决单个前缀映射到加密前缀的事实。例如，假设有一个中间盒，该中间盒计算到前缀 P 的连接数。P 映射到 2 个加密的前缀 P_1 和 P_2。如果原始中间盒规则是“如果 v 在 P 然后计数器+ + ”，网关应该产生“如果 v 在 P 我或 v 在 P_2 然后计数器++ ”。

防火墙和 DPI 服务的规则来自各种来源，并且对于允许或不允许谁知道规则的规则，可以有不同的策略。例如，渗透检测规则可以包括客户可能希望对云提供商保密的公司产品或未发布项目的关键字。另一方面，许多 DPI 规则是 DPI 供应商的专有功能，可以允许提供者学习规则，但不能允许客户（网关）学习规则。Embark 支持三种不同的 KeywordMatch 规则模型，

允许客户和提供者共享自己愿意的规则：（a）客户知道规则，而提供者不知道；（b）提供者知道规则，而客户不知道；或（c）双方都知道规则。PrefixMatch 规则仅支持（a）和（c）-网关*必须*知道规则才能正确执行加密。

如果允许客户端知道规则，则他们会对它们进行加密-生成 KeywordMatch，AES 或 PrefixMatch 规则-并将其发送到云提供商如果允许云提供商也知道这些规则，则客户端将发送这些加密后的规则，并在明文中注明；如果不允许云提供商，则客户端仅以随机顺序发送加密的规则。

如果不允许客户端（网关）知道规则，则我们必须以某种方式允许云提供商使用客户端的密钥来学习每个规则的加密。这是通过使用 Yao 的乱码电路[65]和遗忘的传输[40]的经典组合来实现的如 BlindBox [55]最初应用的那样。与在 BlindBox 中一样，只有在规则由受信任的第三方（例如 McAffee，Symantec 或 EmergingThreats）签署后，此交换才能成功-云提供商在没有签名的情况下应该无法生成自己的规则，因为这将允许云提供商从客户端流量中读取任意数据。与 BlindBox 不同，此规则交换仅发生一次-网关初始化规则时。完成此设置后，来自企业的所有连接都将使用网关上的相同密钥进行加密。

规则更新。对于 PrefixMatch，规则更新需要谨慎处理。添加新的前缀/范围或删除现有范围可能会影响现有前缀的加密。原因是新前缀可以与现有前缀重叠。在最坏的情况下，所有规则的加密都需要更新。

旧规则的加密发生更改的事实带来了两个挑战。第一个挑战是中间盒状态的正确性。考虑具有转换表的 NAT，转换表包含活动连接的端口和 IP 地址。使用 EncryptValue 对 IP 地址进行加密取决于前缀列表，因此在更新规则后，可能会对 IP 地址进行不同的加密，从而与 NAT 表不一致。因此，NAT 状态也必须更新。第二个挑战是竞争条件：如果在旧规则集下加密的数据包仍在流动时，中间盒采用新的规则集，则这些数据包可能会被错误分类。

为了保持一致的状态，网关首先为新的前缀集运行 EncryptPrefixes。然后，网关向云宣布即将进行的更新，中间盒将其当前状态发送到网关。网关通过产生新的加密来更新此状态，并将新状态发送回中间盒。在这段时间内，网关继续基于旧前缀和中间盒对流量进行加密根据旧规则对其进行了处理。一旦所有中间盒都具有新状态，网关就会向云发送一个信号，表明它将“交换”新数据。云将在此信号之后缓冲传入的数据包，直到管道中所有正在进行的数据包在云上完成处理为止。然后，云向所有中间盒发出信号，以“交换”新规则和状态。最后，它开始处理新数据包。对于[51]中定义的每包一致性，缓冲时间受流水线的数据包处理时间限制，通常为数百毫秒。但是，为了确保每个流的一致性，缓冲时间受流的生存期的限制。如此长时间的缓冲是不可行的。在这种情况下，如果云具有备份中间盒，则可以使用迁移避免方案[43]来保持一致性。请注意，对中间盒的所有更改都在*控制平面中*。

5 中间盒：设计与实现

Embark 支持表 1 中列出的一组中间盒的核心功能。表 1 还列出了 Embark 支持的功能。在附录 A 中，我们回顾了每个中间盒的核心功能，并解释了为什么表 1 中的功能足以支持这些中间盒。在本节中，我们重点介绍中间盒的实现方面。

5.1 标头中间盒

在 IP 和传输头上运行的中间盒仅包括防火墙，NAT 和 L3 / L4 负载平衡器。防火墙是只读的，但是 NAT 和 L4 负载平衡器可能会重写 IP 地址或端口值。对于标头中间盒，读取和写入操作的每个数据包操作均保持不变。

对于读取操作，防火墙从网关接收一组加密规则，并像正常流量一样直接将它们与加密数据包进行比较。因为 PrefixMatch 支持 <和>，所以防火墙可以使用任何标准分类算法[34]。

对于写操作，中间盒从地址池中分配值。它在规则生成阶段从网关接收这些加密的池值。这些加密规则标记有保留给重写值的特殊后缀。当网关收到具有这样重写值的数据包时，它将从池中还原纯文本值，而不是从 options 标头中解密该值。

中间盒在写入后可以照常重新计算校验和。

5.2 DPI 中间盒

我们像 BlindBox [55]中那样修改执行 DPI 操作的中间盒。中间盒通过加密的扩展通道（而不是数据包有效载荷本身）进行搜索，并且如果在扩展中发现黑名单，则阻止或记录连接。登船

也提高了如在讨论了正则表达式规则的设置时间和安全§2.4 。

5.3 HTTP 中间盒

家长过滤器和 HTTP 代理从扩展通道读取 HTTP URI。如果父母过滤器观察到列入黑名单的 URI，它将丢弃属于该连接的数据包。

Web 代理需要对所有中间框 Embark 支持进行最多的修改；尽管如此，我们的代理取得了良好的业绩，我们将在讨论§6 。 代理缓存 HTTP 静态内容（例如，图像）以提高客户端性能。当客户端打开新的 HTTP 连接时，典型的代理将捕获客户端的 SYN 数据包并打开与客户端的新连接，就像代理是 Web 服务器一样。然后，代理在后台打开与原始 Web 服务器的第二个连接，就好像它是客户端一样。当客户端发送对新内容的请求时，如果内容在代理的缓存中，则代理将从那里提供内容。否则，代理会将请求转发到 Web 服务器并缓存新内容。

代理具有加密文件路径到加密文件内容的映射。当代理在端口 80 上接受新的 TCP 连接时，代理会从扩展通道中提取该连接的加密 URI，并在缓存中查找它。确定性加密的使用使代理能够使用不变的快速搜索数据结构/索引，例如哈希映射。我们有两种可能的情况：命中或未命中。如果存在缓存命中，则代理通过现有的 TCP 连接从缓存发送加密的文件内容。即使无法解密 IP 地址或端口，代理仍然可以接受连接，因为网关会透明地加密/解密头字段如果存在缓存未命中，则代理将打开一个新连接，并将加密的请求转发到 Web 服务器。回想一下，流量在转发到 Web 服务器之前会反弹回网关，以便网关可以解密标头字段和有效负载。相反，来自 Web 服务器的响应数据包由网关加密并由代理接收。然后，代理缓存并向后发送加密的内容。内容分成数据包。数据包有效载荷是按数据包加密的。因此，网关可以正确解密它们。数据包有效载荷是按数据包加密的。因此，网关可以正确解密它们。数据包有效载荷是按数据包加密的。因此，网关可以正确解密它们

5.4 局限性

如表 1 所示，Embark 支持多种中间盒的核心功能，但并非所有中间盒功能都可以设想外包。现在我们讨论一些例子。首先，对于入侵检测，Embark 不支持无法在一定数量的关键字匹配中扩展的正则表达式，在流量上运行任意脚本[47]或将研究的不同流相关联的高级统计技术

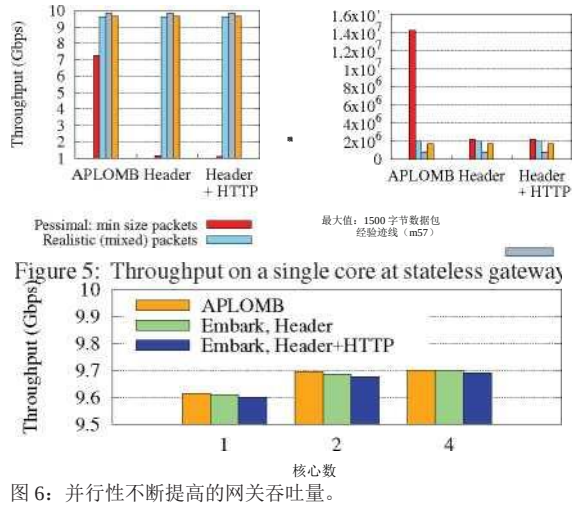


Figure 5: Throughput on a single core at stateless gateway

图 6：并行性不断提高的网关吞吐量。

在研究文献中[69]。

其次，Embark 不支持应用程序级中间盒，例如 SMTP 防火墙，应用程序级网关或代码转换器。这些中间盒以特定于应用程序的方式解析流量-KeywordMatch 不支持这种解析。第三，Embark 不支持端口扫描，因为端口的加密取决于关联的 IP 地址。支持所有这些功能是我们未来工作的一部分。

6 评估

现在，我们从性能的角度研究 Embark 是否实用，并查看由于加密和重定向导致的开销。我们在带有 2.6GHz Xeon E5-2650 内核和 128GB RAM 的现成 16 核服务器上使用 BESS（伯克利可扩展软件交换机，以前为 SoftNIC [35]） 构建了网关。网络硬件是单个 10GbE Intel 82599 兼容网卡。我们在我们的研究实验室中部署了原型网关，并通过该网关将来自 3 个服务器的测试流量重定向到该网关。这三台客户端服务器的硬件规格与我们用作网关的服务器相同。我们在 Amazon EC2 上部署了中间盒。对于大多数实验我们使用由 Pktgen 生成的合成工作量[63]；对于指定经验跟踪的实验，我们在 IPv4 中都使用了 m57 专利跟踪[26]和 ICTF 2010 跟踪[62] 。

对于基于 BlindBox 的 DPI 处理，我们仅针对 Embark 在 BlindBox 之上进行的改进提供实验结果，并向读者介绍[55]，以获得详细的 DPI 性能。

6.1 企业绩效

我们首先评估 Embark 在企业的间接费用。

6.1.1 网关

一个典型的企业需要多少台服务器才能将流量外包给云？ 图 5 显示了加密要发送到云的流量时的网关吞吐量，

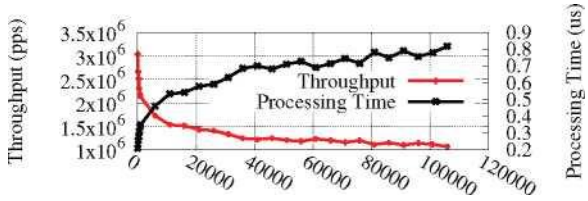


图 7：吞吐量随着 PrefixMatch 规则数量的增加而增加。

首先使用常规重定向（在 APLOMB [54]中使用），然后使用 Embark 的 L3 / L4 标头加密，最后使用 L3 / L4 标头加密以及无状态 HTTP /代理加密。对于禁用了有效负载加密（DPI）的经验性流量跟踪，Embark 平均每个核心 9.6Gbps；而对于每个核心而言，Embark 平均为 9.6Gbps。对于全尺寸数据包，它可以达到 9.8Gbps 以上。在可伸缩性实验中（图 6 ）拥有 4 个专用于处理的核心，我们的服务器可以以高达 9.7Gbps 的速度转发经验流量，同时对标头和 HTTP 流量进行加密。HTTP 开销和 L3 / L4 开销之间没有什么区别，因为 HTTP 加密仅在 HTTP 请求上发生-数据包的一小部分。启用 DPI（未显示）后，吞吐量下降到每个核心 240Mbps，这表明企业将需要为网关分配至少 32 个核心。*网关的吞吐量和延迟如何随着 PrefixMatch 的规则数量而扩展？* 在§ 3.2 ，我们讨论了 PrefixMatch 如何存储排序间隔；每个数据包加密都需要对间隔进行二进制搜索。因此，随着间隔图的大小变大，我们可以期望需要更多的时间来处理每个数据包，并且吞吐量会降低。我们在图 7 中测量了这种影响。在 y_1 轴上，我们以 0 到 100k 的规则数量显示了网关上每个数据包吞吐量的汇总。这里的代价是对数的，这是二进制搜索的预期性能。从 0-10k 规则开始，吞吐量从 3Mpps 下降到 1.5Mpps；在此之后，其他规则的性能损失逐渐减少。添加额外的 90k 规则会将吞吐量降至 l.IMpps。在 y_2 轴，我们测量每个数据包的处理时间，*即*网关加密数据包的时间；处理时间遵循相同的对数趋势。

PrefixMatch 是否比现有的订单保存算法更快？ 我们将 PrefixMatch 与 BCLO [21]和 mOPE [48]这两种著名的顺序保留加密方案进行了比较。表 2 显示了结果。我们可以看到 PrefixMatch 比这些方案快四个数量级。

手术	BCLO	忧郁	前缀匹配
加密 10K 规则	9333 ^s	6640 ^s	0.53 ^s
加密 10 万条规则	9333 ^s	8300 ^s	0.77 *克小号*
解密	169 ^s	0.128 ^s	0.128 ^s

表 2：PrefixMatch 的性能。

PrefixMatch 的内存开销是多少？ 使用未优化的 C ++对象，在内存中存储 10k 规则需要 1.6MB，在内存中存储 100k 规则需要 28.5MB 该开销可以忽略不计。

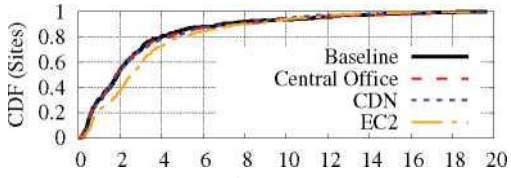


图 8：不同部署下的页面加载时间。

6.1.2 客户绩效

我们使用网络性能来了解 Embark 的端到端用户体验。图 8 显示了通过我们的测试台加载的 Alexa top-500 网站的 CDF。我们假设三个不同的服务提供商比较基线（直接下载）：在中央办公室（CO）中托管服务的 ISP，内容分发网络和传统云提供商（EC2）。网关的平均 RTT 为 60 q 分别为 s，4ms 和 31ms。我们在 EC2 上部署了 Embark，并在实验中使用了此部署，但是对于 CO 和 CDN，我们在测试台中使用延迟和服务器来模拟部署。在实验中，我们运行了 NAT，防火墙和代理（缓存为空）的管道。由于 Embark 使用了“反弹”重定向，因此所有页面加载时间都会增加一小部分。在中位数情况下，ISP /中央办公室的增加时间少于 50ms，CDN 的增加时间少于 100ms，使用 EC2 的增加时间不到 720ms；因此，基于 ISP 的部署将逃避人类的理解[39]，但是 CDN（或云部署）可能会引入人类可察觉的开销。

6.1.3 带宽开销

我们评估了两个成本：由于我们的加密和元数据而导致的带宽增加，以及由于“反弹”重定向而导致的带宽成本增加。

Embark 加密增加了多少发送到云的数据量？ 由于三项加密成本，网关夸大了流量大小：

- 如果企业使用 IPv4，则要从 IPv4 转换为 IPv6，每包成本为 20 字节。如果企业默认使用 IPv6，则无需支付任何费用。
- 如果启用了 HTTP 代理，则每个请求平均在其他加密数据中平均有 132 个字节。
- 如果 HTTP IDS 启用，有在最坏的情况有 5 x 头顶上的所有 HTTP 有效载荷[55]。

我们使用 m57 跟踪来了解这些开销将如何在企业中合计。在上行链路中，从网关到中间盒服务提供商，由于仅标头网关的加密成本，流量将增加 2.5%。业务将通过 4.3 增加 x 在上行链路上进行的网关支持 DPI 中间件。

使用 Embark 可使网关和云之间的带宽增加多少？该带宽将增加企业的网络成本多少？ Embark 在与中间盒服务提供商之间来回发送所有网络流量以进行处理

应用	基准吞吐量	上载吞吐量
IP 防火墙	9.8Gbps 的	9.8Gbps 的
NAT	3.6Gbps 的	3.5 Gbps
负载均衡器 L4	9.8 Gbps	9.8Gbps 的
网络代理	1.1Gbps	1.1Gbps
入侵检测系统	85 Mbps	166Mbps [55]

表 3：经验型工作负载的中间盒吞吐量。

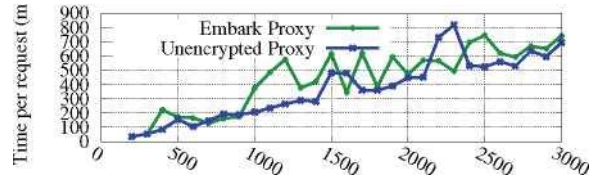


图 9：每页访问时间与代理上并发连接数的关系。

将该流量发送到整个 Internet。

在 ISP 上下文中，客户端的中间盒服务提供者和网络连接提供者是相同的，并且人们可能期望中继到中间盒和从中间盒来回的流量的成本将被合并到一个服务“包”中；考虑到在中心办公室部署的延迟优势（如图 8 所示），我们期望基于 ISP 的部署是部署 Embark 的最佳选择。

在云服务设置中，客户端必须支付第三方 ISP 来向云传输数据或从云传输数据，然后再第三次支付该 ISP 来通过网络实际传输数据。使用当前的美国带宽定价[24，38，61]，我们可以估计多少外包将增加整体带宽成本。多站点企业通常会提供两种网络成本：Internet 访问和域内连接。互联网访问通常具有较高的带宽，但较低的 SLA；流量还可以在共享以太网发送[24 ，61]。 域内连接通常在公司站点之间具有较高的 SLA 和较低的带宽的专用虚拟以太网链路。由于退回重定向是通过“便宜”链接进行的，因此在公开销售数量下，仅采用标头加密对带宽成本的总体影响在 15％至 50％之间；如果使用 DPI 加密，则成本会增加 30-150％。

6.2 中间盒

现在，我们评估每个中间盒的开销。

是否因 Embark 而降低了中间盒的吞吐量？

表 3 显示了我们实现的应用程序所承受的吞吐量。IP 防火墙，NAT 和负载平衡器都是“仅标头”中间盒；显示的结果比较了同一数据平面上的数据包处理，一次是使用加密的 IPv6 数据，一次是使用未加密的 IPv4 数据。可以观察到任何开销的唯一中间盒是 NAT-仅减少了 2.7％。

我们重新实现了 Web 代理和 IDS，以启用它们需要对加密数据进行字节流感知的操作。我们将 Web 代理实现与 Squid [10] 进行了比较，以表明 Embark 可以实现竞争性能。无论有没有加密数据 Web Proxy 都能维持相同的吞吐量，但是，正如我们稍后将介绍的那样，每次缓存命中确实有更长的服务时间。IDS 编号将 Snort（基准）与 BlindBox 实现进行比较；这不是苹果与苹果的比较，因为 BlindBox 的执行效果几乎完全相同匹配 Snort 匹配正则表达式的地方。

接下来，我们为防火墙，代理和 IDS 提供了一些特定于中间盒的基准。

防火墙：*Embark 是否支持典型防火墙配置中的所有规则？由于加密，规则集会“扩展”多少？*

我们用机构的网络管理员提供给我们的三个规则集和来自新兴威胁[3]的 IP 防火墙规则集对防火墙进行了测试。 我们能够使用范围和关键字匹配加密对所有规则进行编码。加密后，三个规则集的大小没有变化，而来自“新兴威胁”的另一个规则集的大小则从 1363 扩大到 1370，增加了 0.5％。因此，我们得出的结论是，它对防火墙性能的影响可以忽略不计。

代理/缓存：表 3 中显示的吞吐量不是用于衡量代理性能的典型指标。一个更好的代理度量标准是代理可以同时处理多少个连接，以及它为每个客户端提供什么服务时间。在图 9 中，我们绘制了服务时间与并发连接数的关系，发现 Embark 的平均时间要比未加密的代理平均高出每页数十到数百毫秒。这不是由于计算成本，而是由于加密的 HTTP 标头值在与主要数据连接不同的通道上传输的事实。Embark 代理需要在这两个流之间进行同步。这种同步成本增加了服务时间。**入侵检测：**我们的 IDS 基于 BlindBox [55]。每当客户启动新连接时，BlindBox 都会产生大量的“设置成本”。但是，借助 Embark，网关和云将维持一个长期的持久连接。因此，当初始配置网关时，只需支付一次此设置费用。Embark 还可以启发式地将规则集中的正则表达式扩展为完全匹配的字符串。这样有两个好处：

*(1) 端到端性能改进。*如果 BlindBox 初始握手为 97s[55] 以打开新连接并生成加密的规则，则 Embark 下的最终主机永远不会支付此费用。取而代之的是，网关支付一次性的设置费用，然后，最终主机仅执行 3-5 个 RTT 的正常 TCP 或 SSL 握手。在我们的测试平台上，这大约需要 30 到 100 毫秒，具体取决于站点和协议-

改善了 4 个数量级。

*(2) 安全性提高。*使用来自 Snort 的 IDS 规则集，我们将正则表达式转换为完全匹配的字符串，如第2.4节所述。在 BlindBox 中，与正则表达式相比，可以以更高的安全性支持完全匹配规则。有了 10G 内存，我们就可以将这个规则集中大约一半的正则表达式转换为有限数量的精确匹配字符串。其余的导致太多可能的状态我们使用两个规则集来对此进行评估[3,9]。对于第一个规则集，BlindBox 会针对 33%的规则求助于较低的安全级别，但是 Embark 只会要求 11.3%的安全级别。对于第二个规则集，BlindBox 将为 58%的规则使用较低的安全性，而 Embark 只会为 20.2%的安全性使用较低的安全性。同时，Embark 不支持较低的安全级别，因此 Embark 根本不支持其余的正则表达式规则。

还值得注意的是，在以下三种情况之一中，以这种方式进行正则表达式扩展会使一次性设置非常慢：网关可能看不到规则的情况。原因是，在这种情况下，Embark 运行了第 4.2 节中讨论的乱码规则交换协议，其规则数量的减少是线性的。在一台机器上，由于关键字数量众多，通往服务器初始设置的网关将花费 3,000 多个小时来生成一组加密规则。幸运的是，这种安装成本很容易并行化。此外，在第 4.2 节中讨论的其他两种规则交换方法中不会发生此设置成本，因为它们每个关键字仅依赖一个 AES 加密，而不是乱码电路计算，后者的成本要高出六个数量级。

7 相关工作

中间件外包：安普仁[54]是外包企业的中间件云，我们更详细地讨论了实际的服务§2 。

数据保密性：在云数据的机密性得到了广泛的认为是重要的问题研究人员提出了软件解决方案[18]，web 应用程序 [30 ，50]，文件系统[19,36,31]，数据库[49 ，46] ，和虚拟机[68]。 CryptDB [49] 是最早对加密数据进行计算的实用系统之一，但是它的加密方案和数据库系统设计不适用于我们的网络设置。

关注流量处理，与 Embark 关系最密切的工作是在第 2.4 节中讨论的 BlindBox [55] 。mcTLS [41]提出了一种协议，其中客户端和服务器可以共同授权中间盒来处理加密流量的某些部分。与 Embark 不同，中间盒可以访问*未加密的数据*。最近的一篇论文[67]提出了一种用于外包中间盒的系统体系结构，以专门对加密流量执行深度数据包检查。

跟踪匿名化和推断：一些专注于*脱机* 处理的系统允许对匿名数据进行一些分析[44、45]；它们不适合 Embark，不适合在线处理。Yamada 等人[64]展示了如何仅通过使用数据大小和数据包的时序对 SSL 加密的数据包执行一些非常有限的处理，但是他们无法对连接数据的内容进行分析。

加密方案： Embark 的 PrefixMatch 方案类似于保留订单的加密方案[15]， 但是没有现有的方案同时提供我们所需的性能和安全性保持顺序的加密（OPE）方案，例如[21，48]是>比 PrefixMatch 慢 10000 倍（§6 ） 和另外泄漏加密的 IP 地址的顺序。另一方面，OPE 方案更为通用，适用于更广泛的场景。另一方面，PrefixMatch 是针对我们的特定方案而设计的。

Boneh 等人的加密方案。[22]能够检测加密值是否与范围匹配并提供与 PrefixMatch 类似的安全保证；同时，它比已经比 PrefixMatch 慢的 OPE 方案慢几个数量级。

致谢

我们感谢牧羊人 Srinivasan Seshan 和匿名审阅者的深思熟虑。我们也感谢 VMware Research 的 Dahlia Malkhi 和 Ittai Abraham 对 PrefixMatch 的宝贵反馈。

中间盒的足够属性

在本节中，我们在表讨论 IP 防火墙，NAT，L3 / L4 负载平衡器的核心功能 1，以及为什么在表的列 2 中列出的属性 1 是足够用于支撑这些中间件的功能。由于这些属性的足够明显，因此我们省略了对表中其他中间框的讨论。Embark 专注于这些中间盒的核心（“教科书”）功能的原因是，这些中间盒上存在变型和不同的配置而 Embark 可能不支持其中的某些功能。

A.1 IP 防火墙

来自不同供应商的防火墙可能具有明显不同的配置和规则组织，因此我们需要提取防火墙的通用模型。我们使用了[66]中 定义的模型，该模型描述了 Cisco PIX 防火墙和 Linux iptables。在此模型中，防火墙由多个访问控制列表（ACL）组成。每个 ACL 包含规则列表。规则可以以*（谓词，操作）*的形式解释，其中*谓词*描述与该规则匹配的数据包，而*动作*描述对匹配的数据包执行的操作。谓词定义为源/目标 IP 地址和端口范围以及协议的组合。可能采取的行动包括*“接受”*和*“拒绝”*。

让 Enc 表示通用加密协议，而 $(SIP[], D//^{1}\|SP[], DP[], P)$ 表示规则的谓词。具有 5 元组 *(SIP，DIP，SP，DP，P) e* (*SIP* []，*DIP* []，*SP* []，*DP* []，*P) 的任何数据包都*匹配该规则。我们对元组和规则都进行加密。加密的以下属性对于防火墙是足够的。

(SIP，DIP，SP，DP，P) e (SIP[]，DIP[]，SP[]，DP[]，P) 和

Enc (SIP，DIP，SP，DP，P) e

Enc (SIP[]，DIP[]，SP[]，DP[]，P)

（3）

A.2 NAT

典型的 NAT 将一对源 IP 和端口转换为一对外部源 IP 和端口（往返），其中外部源 IP 是网关的外部地址，并且可以任意选择外部源端口。本质上，NAT 维护从一对源 IP 和端口到外部端口的映射 NAT 具有以下要求：1）相同的对应映射到相同的外部源端口；2）不同的线对不应映射到相同的外部源端口。为了满足它们，下列属性就足够了：

$$(SIP_1, SP_1) = (SIP_2, SP_2)$$

$$>\text{的 Enc}'\ SIP_1, SP_1) = \text{的} Enc\ (SIP_2, SP_2),$$

$$\text{Enc}\ (SIP_1, SP_1) = \text{Enc}\ (SIP_2, SP_2)$$

$$(SIP_1, SP_1) = (SIP_2, SP_2)。$$

但是，我们可以放松 1）到：属于相同 5 元组的源 IP 和端口对应映射到相同的外部端口。放宽此要求后，仍然可以保留 NAT 的功能，但由于同一对映射到不同的端口，因此 NAT 表可以更快地被填充。但是，我们认为这种扩展实际上很小，因为主机上的应用程序很少使用相同的源端口连接到不同的主机或端口。足够的属性将变为：

$$(SIP_1, DIP_1, SP_1, DP_1, P_1) = (SIP_2, DIP_2, SP_2, DP_2, P_2) \wedge \text{Enc}\ (SIP_b SP_1) = \text{Enc}\ (SIP_2, SP_2) \quad (6)$$

和

$$\text{Enc}\ (SIP_1, SP_1) = \text{Enc}\ (SIP_2, SP_2)$$

$$(SIP_1, SP_1) = (SIP_2, SP_2)。$$

A.3 L3 负载均衡器

L3 负载平衡器维护服务器池。它根据 L3 连接信息为传入数据包选择服务器。L3 负载平衡的常见实现是在交换机中使用 ECMP 方案。它保证通过散列 5 元组将相同流的数据包转发到同一服务器因此，L3 负载均衡器的足够属性是：

$$(SIP_1, DIP_1, SP_1, DP_1, P_1) = (SIP_2, DIP_2, SP_2, DP_2, P_2) \&$$

$$Enc\ (SIP_1, DIP_1, SP_1, DP_1, P_1) = \text{Enc}\ (SIP_2, DIP_2, SP_2, DP_2 \quad 。$$

（8

A.4 L4 负载均衡器

L4 负载平衡器[4]或 TCP 负载平衡器还维护服务器池。它充当接受客户端连接的 TCP 端点。接受来自客户端的连接后，它将连接到服务器之一，并在客户端和服务器之间转发字节流。加密方案应确保两个相同的 5 元组具有相同的加密。此外，两个不同的 5 元组不应具有相同的加密，否则 L4 负载均衡器将无法区分这两个流因此，支持 L4 负载均衡器的足够属性是：

$$(SIP_1, DIP_1, SP_1, DP_1, P_1) = (SIP_2, DIP_2, SP_2, DP_2, P_2) \&Enc\ (SIP_1, DIP_1, SP_1, DP_1, P_1) =$$

$$\text{Enc}\ (SIP_2, DIP_2, SP_2, DP_2)$$

（9

B PrefixMatch 的形式属性

在本节中，我们显示 PrefixMatch 如何支持表 1 中指示的中间盒。首先，我们正式列出 PrefixMatch 保留的属性。如 3.2 中所述，PrefixMatch 通过保证属性 3 来保留防火墙的功能。此外，PrefixMatch 还确保以下属性：

$(SIP_1, DIP_1, SP_1, DP_1, P_1) = (SIP_2, DIP_2, SP_2, DP_2, P_2)$ $Enc(SIP_1, DIP_1, SP_1, DP_1, P_1) = Enc(SIP_2, DIP_2.SP.DP.P$（10）下列语句*很有可能成立：*

$$Enc(SIP_1) = Enc(SIP_2) \quad SIP_1 = SIP_2 \tag{11}$$

$$Enc(DIP_1) = Enc(DIP_2) \quad DIP_1 = DIP_2 \tag{12}$$

$$Enc(SIP_1, SP_1) = Enc(SIP_2 SP_2) \quad (SIP_1, SP_1) = (SIP_2, SP_2) \tag{13}$$

$$Enc(DIP_1, DP_1) = Enc(DIP_2, DP_2) \quad (DIP_1, DP_1) = (DIP_2, DP_2) \tag{14}$$

$$Enc(P_1) = Enc(P_2) \quad P_1 = P_2 \tag{15}$$

我们讨论这些特性如何意味着所有的充分性§一个如下。

NAT 我们将显示等式（10）-等式（15）暗示等式（6）-等式（7）。给定 $(SIP_y DIP_{1,} SP_b DP_{1,} P_1) = (SIP_2, DIP_2, SP_2, DP_2, P_2)$，由公式。（10），我们有 $Enc(SIP_b SP_1) = Enc(SIP_2, SP_2)$。因此，等式（6）成立。类似地，给定 $Enc(SIP_b SP_1) = Enc(SIP_2, SP_2)$，通过等式（13），我们具有 $(SIP_1, SP_1) = (SIP_2, SP_2)$。因此，式（7）也成立。注意，如果我们不放松等式（6）中的性质，我们将无法获得这样的证明。

L3 负载均衡器通过等式（10），可保持等式（8）的左右方向。通过式（11）∽式（15），式（8）的左右方向也成立。

L4 负载均衡器通过等式（10），可保持等式（9）的左右方向。通过式（11）∽式（15），式（9）的左右方向也成立。

参考

[1] Brocade 网络功能虚拟化。http://www.brocade.com/zh /产品服务/软件-网络/网络功能-虚拟化.html 。

[2] Cisco IOS IPv6 命令。http: // www。cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/command/ipv6-cr-book/ipv6-s2.html 。

[3] 新兴 Threats.net 打开规则集。http://rules.emergingthreats.net/ 。

[4] HAProxy。http://www.haproxy.org/。

[5] 英特尔 82599 10 GbE 控制器数据表。http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/82599-10-gbe-controller-datasheet.pdf。

[6] 网络边缘服务产品。https: //www.juniper.net/us/en/products-services/network-edge-services/ 。

[7] 电信网络功能虚拟化。http://www.dell.com/learn/us/ en / 04 / tme-telecommunications- solutions-telecom-nfv / 。

[8] OPNFV：加速开放的平台 NFV。https://www.opnfv.org/sites/ opnfv / files / pages / files / opnfv_ whitepaper_103014.pdf 。

[9] Snort v2.9 社区规则。https: //www.snort.org/downloads/ community / community-rules.tar.gz 。

[10] Squid：优化 Web 交付。http: // www。squid-cache.org/ 。

[11] Telefdnica NFV 参考实验室。http: // www.tid.es/long-term-innovation/network-innovation/telefonica- nfv-reference-lab 。

[12] 什么是白盒开关？https: // www。sdxcentral.com/resources/white-box/what-is-white-box-networking/ 。

[13] ZScaler。http://www.zscaler.com/。

[14] AT&T Domain 2.0 愿景白皮书。https://www.att.com/Common/ about_us / pdf / AT&T%20Domain%202。0%20Vision%20White%20Paper.pdf ，11 月 2013。

[15] R. Agrawal，J。Kiernan，R。Srikant 和 Y. Xu。保留数字数据加密的顺序。在 *2004 年 ACM SIGMOD 国际数据管理大会上的会议记录，* SIGMOD '04，第 563-574 页。ACM，2004 年。

[16] Ars Technica。呼叫中心员工窃取客户数据后，AT&T 被罚款 2500 万美元。http://arstechnica.com/tech-policy / 2015/04 / att-fined-25-illion-after-call-center-Employees-stole-customers-data /。

[17] Aryaka。广域网优化。http: //www.aryaka.com/ 。

[18] A. Baumann，M。Peinado 和 G. Hunt。使用避风港将应用程序从不受信任的云中屏蔽出来。在 *第 11 届 USENIX 操作系统设计和实现会议的会议记录中，* OSDI'14，第 267-283 页 USENIX 协会，
2014。

[19] M. Blaze。UNIX 的加密文件系统。在 *第一届 ACM 计算机和通信安全性会议论文集，* CCS '93，第 9-16 页。ACM，1993 年。

[20] 彭博商业。RadioShack 出售与州结算后的客户数据。http://www.bloomberg.com/news/ article / 2015-05-20 / radioshack- receive-approval-to-sell-name- to-standard-general 。

[21] A. Boldyreva，N。Chenette，Y。Lee 和 A. O'Neill。保留顺序的对称加密。在 *第 28 届年度国际密码学进展国际会议论文集：密码技术的理论和应用，* EURO-CRYPT'09，第 224-241 页。施普林格出版社，2009 年。

[22] D. Boneh，A。Sahai 和 B. Waters。具有短密文和私钥的完全防串扰叛国者追踪。在 *第 24 届国际密码技术理论与应用国际会议论文集中，* EURO-CRYPT'06，第 573-592 页。施普林格出版社，2006 年。

[23] PR Clearinghouse。数据泄露年表。http://www.privacyrights.org/数据泄露。

[24] 康卡斯特。小型企业互联网。http: //business.comcast.com/internet/ business-internet / plans-pricing 。

[25] I. Cooper，I。Melve 和 G. Tomlinson。Internet Web 复制和缓存分类法。IETF RFC 3040，2001 年 1 月。

[26] 数字语料库。m57 专利方案。http://digitalcorpora.org/corpora/ 场景/ M57-专利的场景。

[27] 欧洲电信标准协会。NFV 白皮书。https：//portal.etsi。org / nfv / nfv_white_paper.pdf

[28] SK Fayazbakhsh，L。Chiang，V。Sekar，M。Yu 和 JC Mogul。使用 FlowTag 在动态中间盒操作的情况下实施网络范围的策略。在 *第 11 届 USENIX 网络系统设计和实现会议论文集，* NSDI'14，第 533-546 页。USENIX 协会，2014 年

[29] A. Gember-Jacobson，R。Viswanathan，C。Prakash，R。Grandl，J。Khalid，S。Das 和 A. Akella。OpenNF：实现网络功能控制的创新。在 *SIGCOMM 举行的 2014 年 ACM 会议论文集中，* SIGCOMM '14，第 163-174 页。ACM，2014 年

[30] DB Giffin，A。Levy，D。Stefan，D。Terei，D。Mazieres JC Mitchell 和 A. Russo。冰雹：保护不受信任的 Web 应用程序中的数据隐私。在 *第 10 届 USENIX 操作系统设计和实现大会会议记录中*，OSDI'12，第 47-60 页。USENIX 协会，2012 年。

[31] E.-J. Goh，H。Shacham，N。Modadugu 和 D. Boneh。SiRiUS：保护远程不受信任的存储。在 *《第十届网络和分布式系统安全性研讨会论文集》中，* NDSS '03，第 131-145 页互联网协会（ISOC），2003 年 2 月。

[32] O. Goldreich。*密码学基础：第 I 卷基本工具*。剑桥大学出版社，2001 年。

[33] M. Goodrich 和 R. Tamassia。*计算机安全简介*。皮尔逊（Pearson），2010 年。

[34] P. Gupta 和 N. McKeown。数据包分类算法。*IEEE Network*，15（2）：24-32，2001 年 3 月。

[35] S. Han，K。Jang，A。Panda，S。Palkar，D。Han 和 S. Ratnasamy。SoftNIC：用于增强硬件的软件 NIC。加州大学伯克利分校 EECS 系 UCB / EECS- 2015-155 技术报告，2015 年 5 月。

[36] M. Kallahalla，E。Riedel，R。Swaminathan，Q。Wang 和 K Fu。Plutus：在不受信任的存储上可扩展的安全文件共享。在*第二届 USENIX 文件和存储技术会议论文集中，* FAST '03 第 29-42 页。USENIX 协会，2003 年。

[37] J. Martins，M。Ahmed，C。Raiciu，V。Olteanu，M。Honda，R。Bifulco 和 F。会慈 ClickOS 和网络功能虚拟化的艺术。在*第 11 届 USENIX 网络系统设计和实现会议的论文集中，* NSDI'14，第 459-473 页。USENIX 协会，2014 年。

[38] Megapath 。以太网数据加。http：//www.megapath.com/promos/ethernet- dataplus /。

[39] RB 米勒。人机对话事务中的响应时间。在 *1968 年 12 月 9 日至 11 日的会议记录中，秋季联合计算机会议，第一部分* AFIPS '68（秋季，第一部分），第 267-277 页。ACM，1968 年。

[40] M.NaorandB.Pinkas。高效的遗忘传输协议。在*第十二届 ACM-SIAM 离散算法年度研讨会论文集中，* SODA '01，第 448-457 页。工业和应用数学协会，2001 年。

[41] D. Naylor，K。Schomp，M。Varvello，I。Leontiadis，J。Blackburn，DR L6pez，K。Papagiannaki，P。Rodriguez Rodriguez 和 P。Steenkiste。多上下文 TLS（mcTLS）：在 TLS 中启用安全的网络内功能。在 *2015 年 ACM 数据通信特别利益小组会议论文集中，* SIGCOMM '15，第 199-212 页。ACM，2015 年。

[42] E. Nordmark。无状态 IP / ICMP 转换算法（SIIT）。IETF RFC 2765，2000 年 2 月。

[43] S. Palkar，C。Lan，S。Han，K。Jang，A。Panda，S。Ratnasamy，L。Rizzo 和 S. Shenker。E2：NFV 应用程序框架。在*第 25 届操作系统原理研讨会论文集中，* SOSP '15，第 121-136 页，纽约，纽约，美国，2015 年。ACM。

[44] R. Pang，M。Allman，V。Paxson 和 J. Lee。魔鬼和数据包跟踪匿名化。*SIGCOMM 计算机通信评论，* 2006 年 1 月 36（1）：29-38。

[45] R. Pang 和 V. Paxson。数据包跟踪匿名化和转换的高级编程环境。在 *2003 年应用程序，技术，体系结构会议上，和计算机通信协议，* SIGCOMM '03，第 339-351 页。ACM，2003 年。

[46] V. Pappas，F。Krell，B。Vo，V。Kolesnikov，T。Malkin SG Choi，W。George，A。Keromytis 和 S. Bellovin。盲人先知：可扩展的专用 DBMS。在 *2014 年 IEEE 安全与隐私研讨会上，* SP '14，第 359-374 页。IEEE 计算机协会，2014 年。

[47] V. Paxson。兄弟：实时检测网络入侵者的系统。*计算机网络，*31（23-24）：2435-2463，Dec。1999 年。

[48] RA Popa，FH Li 和 N. Zeldovich。用于保留订单编码的 IdealSecurity 协议。在 *2013 年 IEEE 安全与隐私专题研讨会论文集中，* SP '13，第 463-477 页。IEEE 计算机协会，2013 年。

[49] RA Popa，CMS Redfield，N。Zeldovich 和 H. Balakrishnan。CryptDB：使用加密查询处理保护机密性。在*《第二十三届 ACM 操作系统原理研讨会论文集》中，* SOSP '11，第 85-100 页。ACM，2011 年。

[50] RA Popa，E。Stark，J。Helfer，S。Valdez，N。Zeldovich MF Kaashoek 和 H. Balakrishnan。使用 Mylar 在加密数据之上构建 Web 应用程序。在*第 11 届 USENIX 网络系统设计和实现会议论文集，* NSDI'14，第 157-172 页。USENIX 协会 2014 年。

[51] M. Reitblatt，N。Foster，J。Rexford，C。Schlesinger 和 D。沃克。网络更新的抽象。在 *ACM SIGCOMM 2012 年会议的应用程序，技术，体系结构和计算机通信协议中，* SIGCOMM '12，第 323-334 页。ACM，2012 年。

[52] V. Sekar，N。Egi，S。Ratnasamy，MK Reiter 和 G. Shi。整合的中间盒体系结构的设计和实现。在*第 9 届 USENIX 网络系统设计和实现会议的会议记录中，* NSDI'12，第 24-24 页 USENIX 协会，2012 年。

[53] V. Sekar，S。Ratnasamy，MK Reiter，N。Egi 和 G. Shi。中间盒宣言：在中间盒部署中实现创新。在*第十届 ACM 网络热点话题研讨会论文集，* HotNets-X，第 21：1-21：6 页。ACM，2011 年。

[54] J. Sherry，S。Hasan，C。Scott，A。Krishnamurthy，S。Ratnasamy 和 V. Sekar。制作中间盒

另一个人的问题：网络处理作为云服务。在 *ACM SIGCOMM 2012 计算机通信的应用程序，技术，体系结构和协议会议上，* SIGCOMM '12，第 13-24 页。ACM，2012 年。

[55] J. Sherry，C。Lan，RA Popa 和 S. Ratnasamy。BlindBox：基于加密流量的深度数据包检查。在 *Proceedingsofthe2015ACM 数据通信特别兴趣小组会议上，* SIGCOMM '15，第 213-226 页。ACM，2015 年。

[56] G. Silowash，T。Lewellen，J。Burns 和 D. Costa。通过流量检查通过加密的 Web 会话检测和防止数据泄露。卡内基梅隆大学软件工程学院 CMU / SEI-2013-TN-012 技术报告，2013 年。

[57] P Srisuresh 和 KB Egevang。传统 IP 网络地址转换器（传统 NAT）。IETF RFC 3022，2001 年 1 月。

[58] D. Thaler 和 CE Hopps。单播和多播下一跳选择中的多路径问题。IETF RFC 2991，2000 年 11 月。

[59] Snort 项目。Snort 用户手册，2014 年。版本 2.9.7。

[60] Verizon 。 2015 年数据泄露调查报告。 http://www.verizonenterprise.com/ DBIR / 2015 / 。

[61] Verizon。 高速互联网套餐。 http://www.verizon.com/ 小型企业/产品/企业-互联网/宽带套餐/ 。

[62] G. Vigna。ICTF 数据。https：//ictf.cs。ucsb.edu/

[63] 威尔斯。 Pktgen 。 https： // 开头 pktgen 。 readthedocs.org/ 。

[64] A. Yamada，Y。Saitama Miyake，K。Takemori，A。Studer 和 A. Perrig。加密 Web 访问的入侵检测。在 2007 年 *第 21 届高级信息网络和应用程序国际会议研讨会上。*

[65] AC-C。瑶 如何生成和交换秘密。在 *"第 27 届计算机科学基础年度研讨会论文集"中，* SFCS '86，第 162-167 页。IEEE 计算机协会，1986 年。

[66] 袁 L，麦 J.Su，陈 H.C.-N 。Chuah 和 P. Mohapatra 。FIREMAN：用于 FIREwall 建模和分析的工具包。在 *2006 年 IEEE 安全与隐私研讨会上，* SP '06，第 199-213 页。IEEE 计算机协会，2006 年。

[67] X. Yuan，X。Wang，J。Lin 和 C. Wang。隐私保护外包中间盒中的深度数据包检查。在 *2016 年 IEEE 计算机通信大会上，* INFOCOM'16,2016。

[68] 张峰，陈洁，陈厚和 B。CloudVisor：使用嵌套虚拟化对多租户云中的虚拟机进行保护改造。在 *《第二十三届 ACM 操作系统原理研讨会论文集》中，* SOSP '11，第 203-216 页。ACM，2011 年。

[69] Zhang Y 和 V. Paxson。检测垫脚石。在 *《USENIX 安全研讨会第 9 届会议记录-第 9 卷，* SSYM'00，第 13-13 页》中。USENIX 协会，2000 年。