外包中间盒中保护隐私的深度数据包检查

星良苑*! ，王新宇*! ，林建雄*和王聪*!
*香港城市大学，中国香港
! 香港城市大学深圳研究院，深圳 518057
{ xinglyuan3-c，jianxilin2-c } @ my.cityu.edu.hk，{ xinywang，congwang } @ cityu.edu.hk。

***摘要*-中间盒对于现代企业网络中的各种高级流量处理至关重要。作为虚拟化服务在云中部署中间盒的最新趋势进一步扩展了中间盒的潜在优势，同时避免了本地维护负担。尽管前景乐观，但设计外包中间盒仍然面临一些安全挑战。首先，许多中间盒处理服务（例如入侵检测）都需要检查数据包有效负载，而由于端到端加密，越来越多地采用 HTTPS 限制了该功能。第二，中间盒使用的许多数据包检查规则实际上是专有的。它们可能包含企业的敏感信息，因此在不受信任的外包环境中配置中间盒时需要强大的保护。在本文中，我们为外包的中间盒提出了一种实用的系统体系结构，以对加密的流量执行深度数据包检查，而不会透露数据包有效载荷或检查规则。我们的第一个设计是一个加密的高性能规则过滤器，该过滤器从数据包有效载荷中获取随机令牌以进行加密检查。然后，我们通过精心定制的技术来详细说明如何全面支持开源的真实规则集。我们正式分析安全强度。Amazon Cloud 的实施表明，我们的系统在每个连接初始化中引入大约 100 毫秒的延迟，对于 500 个并发连接，单个处理吞吐量超过 3500 包/秒。而不透露数据包有效载荷或检查规则。我们的第一个设计是一个加密的高性能规则过滤器，该过滤器从数据包有效载荷中获取随机令牌以进行加密检查。然后，我们通过精心定制的技术来详细说明如何全面支持开源的真实规则集。我们正式分析安全强度。Amazon Cloud 的实施表明，我们的系统在每个连接初始化中引入大约 100 毫秒的延迟，对于 500 个并发连接，单个处理吞吐量超过 3500 包/秒。而不透露数据包有效载荷或检查规则。我们的第一个设计是一个加密的高性能规则过滤器，该过滤器从数据包有效载荷中获取随机令牌以进行加密检查。然后，我们通过精心定制的技术来详细说明如何全面支持开源的真实规则集。我们正式分析安全强度。Amazon Cloud 的实施表明，我们的系统在每个连接初始化中引入大约 100 毫秒的延迟，对于 500 个并发连接，单个处理吞吐量超过 3500 包/秒。然后，我们通过精心定制的技术来详细说明如何全面支持开源的真实规则集。我们正式分析安全强度。Amazon Cloud 的实施表明，我们的系统在每个连接初始化中引入大约 100 毫秒的延迟，对于 500 个并发连接，单个处理吞吐量超过 3500 包/秒。然后，我们通过精心定制的技术来详细说明如何全面支持开源的真实规则集。我们正式分析安全强度。Amazon Cloud 的实施表明，我们的系统在每个连接初始化中引入大约 100 毫秒的延迟，对于 500 个并发连接，单个处理吞吐量超过 3500 包/秒。**

I.简介

中间盒在现代企业网络中无处不在，可提供广泛的专用网络功能[1]-[3]，例如入侵检测，防渗透，防火墙等。然而，维护内部中间盒的基础设施会带来昂贵且昂贵的费用。企业的复杂管理负担[1]。因此，近来的趋势要求将中间盒处理作为虚拟化服务[1]，[3]转移到公共云，同时减轻企业的本地维护负担。这样的外包中间盒可以通过易于管理，成本效益，可伸缩性，容错能力以及其他优势使企业进一步受益。

尽管非常有前途，但是外包中间盒处理也带来了一些新的安全挑战，这些挑战尚待完全解决。首先，将流量重定向到外包由企业定制，并且可以是专有的[6]，[7]，其中包含潜在的敏感信息，例如商业秘密，知识产权等。因此，从企业的角度出发，也强烈要求对这些有价值的规则集进行强有力的保护[7]。] [8] 尤其是在将中间盒部署在外包云环境中时。

大多数现有的中间盒通过一种漫游但受限的方法来处理 HTTPS，只需拦截和解密加密的流量即可[1]，[9]。这种方法除了不合需要地在中间盒处揭示数据包的有效负载外，还很容易构成可能的中间人攻击[10]。最新的称为 BlindBox [5]的设计是第一个使中间盒能够通过 HTTPS 执行 DPI 的设计。但是它仍然不适用于中间盒外包的环境，因为它不考虑在不受信任的环境中保护规则不受中间盒的侵害。我们还指出，由于 BlindBox 的连接设置昂贵，涉及每个端点与中间盒之间的安全的两方计算协议，因此目前尚未准备好进行实际部署。

现有方法的上述缺点促使我们研究一种保护隐私和实用的 DPI 系统。我们的研究旨在使外包中间盒能够对加密流量进行数据包检查，而不会泄露敏感的检查规则或数据包有效载荷。为了应对挑战，我们的第一个见解就是将问题表达为加密令牌匹配。具体来说，可以将流量数据包有效载荷解析并加密为随机令牌。可疑字符串和响应措施 [1] 还可以从数据包检查规则中提取密钥规则对作为键值对，例如（“密码”，“警报”），基于此可以构建加密的规则过滤器以索引那些加密对。通过将加密的交通令牌输入加密的规则过滤器，可以通过现有的可搜索加密技术立即实例化这种蓝图[12]，[13]。

但是，将蓝图变成安全且可用的 DPI 系统仍然遇到不小的障碍。第一个是如何构建具有令人信服的高性能的加密规则过滤器。中间盒能够以低延迟，卓越的吞吐量，内存效率，高速设置等检查数据包内容，对于任何用户来说都是必不可少的。

[1] 通常，现有指令检测系统中的操作包括警报（即，向管理员生成警报），日志（即，记录数据包），丢弃（即，阻止数据包），停止（即，拒绝连接），等[11]。

978-1-4673-9953-1 / 16 / $ 31.00©2016 IEEE

强大的 DPI 系统。直接将现有的可搜索对称加密（SSE）的通用原语（例如[12]，[13]）应用到我们的特定上下文中并不一定能满足我们的所有设计要求。为此，我们建议将 SSE 的安全框架与最近的高性能哈希表设计[14]结合起来，从头开始获得我们的加密规则过滤器。最终的设计包含所有上述性能特征，同时可以证明是安全的。仅当可疑字符串匹配时，操作才会被恢复，以防止入侵和渗透。在整个过程中，除了触发的动作之外，中间盒永远无法学习数据包有效载荷的内容或规则的语义信息。

第二个障碍是如何通过复杂的检查规则为务实检查提供全面的支持。通常，一些规则指定检查属性，例如，数据包字段或有效负载偏移量。揭示它们可能会损害规则和数据包有效负载的机密性。其他具有多个条件的对象需要在显示操作之前对所有条件进行匹配。因此，要支持它们以及其他特殊要求，不仅需要简单的 SSE 应用程序。因此，我们以深入的方式精心设计了量身定制的技术来分别处理它们，同时仍然确保了对规则和有效负载的强大保护。除其他外，我们的技术亮点之一是采用秘密共享对加密的过滤器设计中嵌入的动作进行加密，在多条件匹配的情况下。结果，当且仅当规则中的所有可疑字符串都匹配时，才会触发该操作。此外，我们还考虑了共享公用字符串的规则（即交叉规则检查）和跨不同连接检查重复字符串的规则（即交叉连接检查）。

此外，我们优化了系统实施，以提高安全性和性能。为了隐藏由相同底层字符串生成的传入令牌的相等性，我们交换空间以获取安全性；也就是说，每个可疑字符串都被复制成多个副本，这些副本随后可以通过不同的标记进行匹配。为了防止令牌相等于其他连接，应为新建立的连接构建一个新鲜的加密过滤器。我们进一步将该过程与连接初始化解耦，以便在异步过程中周期性地预先构建多个过滤器。因此，建立连接后可以立即开始检查。总而言之，我们的贡献如下：

- 我们设计了第一个安全的 DPI 系统，该系统可使外包的中间盒对加密的流量执行深度数据包检查，同时对数据包有效负载和检查规则提供强大的保护。
- 我们提出了一种加密的高性能过滤器，该过滤器具有高吞吐量，快速设置和低内存消耗的特点，并精心设计以广泛支持检查规则，并附有详细的协议说明。我们制定了安全定义，并针对自适应选择关键字攻击证明了所提出的系统
- 我们实施系统原型并将其部署在

图 1：系统架构

亚马逊云。实际规则集用于评估。结果表明，我们的设计可以在大多数选定的入侵检测流量转储中直接检测 90％以上的可疑数据包，并且在具有 500 个并发连接的 Amazon 实例上，每个连接的吞吐量达到每秒 3，600 个数据包。

II. PROBLEM 小号 TATEMENT

图 1 说明了我们的系统架构。它由四方组成：企业网络内部的*客户端*，外部网络中的*主机*，企业管理员维护的*管理服务器*（AS ）以及 在外包环境中作为云或网络服务部署的*中间盒*（*MB* ）。我们还使用术语“端点”来表示建立 HTTPS 连接的*客户端* 和*主机*。

A. 应用方案和信任假设

在介绍我们的威胁模型和系统体系结构之前，我们考虑一个常见的应用场景。企业需要彻底检查传入和传出网络数据包的内容，以防御恶意活动。为了提高成本效率，容错能力和良好的可伸缩性，它订阅了远程中间盒服务，例如入侵检测，防渗透等。同时，客户端将为外部主机建立端到端加密，以防止窃听。在这里，我们假设外包的中间盒（例如基于云的中间盒[3]，[5]）功能强大且具有丰富的计算资源，可以在其中启动多个服务器来处理大量的加密流量并处理大量的并发令牌

另外，企业管理员，即规则创建者，需要保护规则[5]，[7]因为将其保留为明文格式将损害企业的隐私。实际上，企业可以从专业供应商[5]（例如 Symantec）订购规则集，或从开源 DPI 系统（例如 Snort [11]）订购某些规则集。然后，它可以自定义规则以防止数据泄露或调整规则以提高检查准确性[6]。即，规则可以与敏感信息高度相关，例如贸易

机密，知识产权等。这里，我们假设 AS 是值得信赖的。它既不会将规则集公开给 *MB*，也不会授予对端点的规则访问权限

半诚实的中间盒：在这项工作中，我们认为 MB 服务提供商可能是半诚实的[7]，[15]，例如公共云[1]，[3]。它忠实地提供 MB 服务，但打算从所通过的流量中利用敏感信息，并尝试推断专有的检查规则。另一方面，MB 部署在不受信任的环境中它很可能被黑客窃听。因此，规则和流量都应加密以实现深度防御。当前，如何验证外包的MB服务不是我们关注的重点并且仍然有待实际解决，如[16]中所述。我们将在不久的将来探索恶意 MB 的威胁。**端点上的信任假设：**我们假设至少一个端点是诚实的，类似于现有 DPI 系统中的威胁模型[5]，[6]。我们还注意到，检测到两个恶意端点与我们的工作正交。例如，可以通过流量模式分析[17]和特定于应用程序的检测[18]来检测恶意端点之间的隐蔽通道。

B. 系统架构概述

我们提出的系统的概述如图1所示。它的功能分为四个阶段之前介绍了四个参与方。**初始化：**首先，两个端点 *S* 和 *R* 运行标准 SSL 协议以建立加密连接。然后，他们需要分别在 *AS* 处通过加密通道注册密钥 K_s和密钥 K_R。同时，*AS* 构建了一个加密过滤器，该过滤器可对从规则中提取的加密字符串操作对进行安全索引，并将其上传到 *MB*。之后，*MB* 可以通过适当的加密过滤器对此连接进行数据包检查。**预处理：**初始化完成后，一个端点开始发送加密的流量。同时，它将基于预定义的原理将数据包有效载荷解析为一组字符串，并使用 K_s将纯文本字符串转换为随机令牌，这些令牌也将发送到 *MB*。我们注意到检查是双向的。另一个端点按照相同的原理解析数据包有效负载，并使用其自己的密钥 K_R生成令牌。

检查：只要加密的流量和令牌到达，*MB* 就会阻止流量并执行建议的安全DPI协议，以流方式通过加密的筛选器处理令牌。如果令牌正确恢复了过滤器的条目，则 *MB* 将采取相应的措施例如，向 *AS* 发出警报，丢弃数据包等。在检查了数据包中的所有令牌而没有匹配之后，该数据包被视为合法并被允许通过。如果需要验证，则 *MB* 会从一批已处理的令牌中连续计算加密摘要，然后将其发送到另一个端点。

验证：类似于[5]，为了检测另一个端点的不诚实/恶意行为，接收端点将使用 SSL 会话密钥来解密有效负载，然后重建摘要以进行令牌验证。

备注：根据先前的研究要求将中间盒外包作为服务[1]，[3]，[15]，我们假设存在

算法 1 建立加密的过滤器

输入：{（str i，act_f），•••，（str_n，act_n）}：从规则集 R 中提取的字符串操作对；K_1，K_2：私钥；F_1，F_2，P_1，P_2：PRF；τ：负载系数；*d*：每个存储桶中的条目数；*f3*：杜鹃阈值。

输出：*F*：加密的过滤器。

1：具有容量|“ff”|的初始哈希表 T_1和 T_2；

2：**F** str i e{ str_1，•••，str_n}，计算 t = F 1（K 1，str_f），$t_1 = P_1$（*t*，1），$t_2 = P_2$（*t*，2），并放置一个到 1 的 d 项中 T_1[T 1]或 T_2[T 2] 如果存在一个空条目。

3：如果没有空的条目存在，随机选择 1 个从条目 T_1[T 1]或 T_2[T 2]，更换内部*起作用*与*行为*，并重新插入*动作*如内示出在步骤 2 *f3* 递归试验。

4：所有操作都插入之后，通过加密它们中的每*行为*®小号，其中所述掩模 S = °F_2（K_2，*STR*），和填充的随机串的空条目。

的 *AS*，它可以是一个现成的网关服务器，并由守信用企业管理员控制建立加密的过滤和重定向企业流量到云中。只有 *AS* 需要了解云。在这种外包设置中，我们还注意到，具有广泛地理覆盖范围的云可以帮助显着减少额外的流量绕行延迟[1]在[1]，[3]之前已经讨论了将流量重定向到外包中间盒的技术在此不再赘述。我们的主要重点是使用精心定制的技术来设计加密过滤器，以涵盖各种检查规则。正如我们在下面提到的，即使这是一项不平凡的任务，也需要同时实现安全性，性能和功能性的思想。

III. 他 P ROPOSED 小号 YSTEM

在本节中，我们将介绍拟议的系统，并解释有关安全性，性能和功能的设计直觉。我们首先提出一个加密的规则过滤器，即我们系统的核心构件。该筛选器使 *MB* 能够对加密的流量执行私有且有效的 DPI，而无需查看数据包有效载荷或检查规则。此外，我们针对各种规则支持量身定制了设计，并以深入的方式介绍了检查过程。

A.加密过滤器

为了使 DPI 能够超过加密流量，一种简单的方法是将规则中指示的可疑字符串加密为 *MB 的*随机令牌，以便以后可以与来自端点的数据包有效载荷的令牌进行令牌匹配。但是，这种方法需要对所有规则令牌进行线性扫描，以检查每个传入令牌[5]。它不具有可伸缩性，因为时间复杂度随着规则数量的增加而增加，这可能成为对带宽不敏感的应用程序的性能瓶颈。

为了提高效率和可伸缩性，我们提出了一种高性能的加密过滤器，该过滤器基于可搜索对称加密的安全框架[12]，[13]和最新的有效哈希表设计[14]之一。基于[12]，[13]中使用的安全技术，我们对内存进行了转换

行动 1	行动 3

高效，高吞吐量的哈希表转换为加密索引，同时保留其原始性能。

与以前的加密索引设计[12] [13]需要在字典中存储加密的字符串和动作不同，我们提出的过滤器通过仅存储加密的动作而仍保持正确性来缩小大小。明确地说，我们首先使用从字符串生成的令牌来寻找对其操作可用的空间，然后将字符串安全地嵌入到随机掩码中以对操作进行加密。结果，仅当掩码和令牌是从相同的可疑字符串派生的时，操作才会恢复。另外，布谷鸟哈希技术使过滤器变得非常紧凑[19]。动作可以在不同的空间之间重新分配，从而使过滤器达到较高的负载率，例如 95 %[14]。**建造：**如前所述，可以从检查规则中提取一对可疑字符串及其响应动作。然后，将所有字符串操作对以随机方式插入到建议的过滤器中，并对其中的所有存储桶进行加密。

算法 1 中介绍了构建加密过滤器的步骤，图 2 中介绍了一个示例性示例。最初，创建了两个哈希表 T_1 和 T_2。对于总共 n 个字符串操作对中的每一个，第一步是将操作 act $_i$ 插入两个存储桶 $T_{i}[t_{i}]$ 和 $T_2[t_2]$ 中的一个中，其中 t_4 和 t_2 通过伪转换。从可疑串-random 功能 $STR_{我}$，即 t = str-$_i$），t_4 = 1），并且 t_2 = P_2（t，2）。为了处理来自不同字符串的哈希冲突，每个存储桶都包含 d 个条目，因此每个插入都将有 2d 个 选择来放置操作。

如果两个存储桶中没有空条目，则触发布谷鸟哈希中的数据重定位。明确地，所述条目中的一个将被随机选择的，并且动作起作用$_{我}$内部将被替换$动作_{我}$。在此之后，行为-是重新插入在从第一步骤中开始一个递归的方式。由于所有的动作被插入，它们经由异或随机掩码，即加密的，$充当_{我}$®小号，其中 S =°F_2（K_2，$STR_{我}$）。基于这种设计，不需要保存 str_i，而采取*行动*如果令牌匹配，$_{我}$仍然可以恢复。详细的协议将在后面的 III-B 部分中显示。最后，其余几个空条目都填充有随机字符串。

关于安全性和可用性的考虑：基于提议的加密过滤器，仅当从可疑字符串中生成令牌时，*MB 才*具有恢复操作的能力。但是直接部署此筛选器并不能提供强大的安全性。首先，当前构造泄漏传入令牌的相等性，无论它们是否匹配，因为它们都是从确定性单向函数生成的。其次，相同的私钥用于生成令牌和加密的过滤器。因此，获得密钥的任何人都可能出于恶意目的（例如 DoS 攻击）而输出令牌。

关于广泛的规则支持，当前用于单字符串匹配的设计仅适用于一小部分检查规则。实际上，还定义了许多其他高级规则来提高检测准确性并捕获复杂的恶意活动[11]。具体来说，这些规则可能包含特定的属性或检查范围，或者需要多条件检查。然而，目前尚不清楚如何通过简单的方式来支持他们

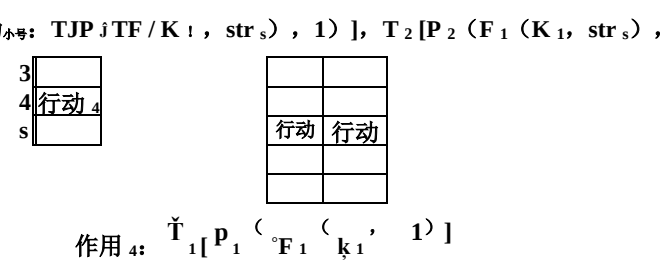

图 2：插入加密过滤器的示例。在此，每个存储桶都有 2 个条目，并且第 $_5$行为的存储桶已满。然后，随机选择动作 $_4$，然后重新放置，然后将动作 $_5$放置在先前放置了*动作*$_4$的条目中。

使用建议的单令牌匹配算法。接下来，我们将为单令牌匹配提出一个安全协议，以解决安全问题，然后提出四种量身定制的设计，以提供全面的 DPI 支持。

B.安全的单令牌检查

该协议启用了单个令牌检查，如下开源规则集[11]中的以下规则所示。应该解决前面提到的两个安全问题。流令牌的等同性必须被隐藏，并且必须针对 DoS 攻击强制执行授权令牌的生成。

Snort 规则 2：警报 tcp $ EXTERNAL_NET $任意- > $ HOME_NET $ 7597（流：到服务器，已建立；内容：“qazwsx.hsq”；）

对于前者，我们建议以空间换取安全。一方面，检查每个单独的连接都需要使用带有新鲜密钥的加密过滤器，因此不同连接中的相同字符串将不再相同。另一方面，每个规则字符串都是重复的，并在过滤器中安全地建立索引，因此即使是相同的可疑字符串也会被随机标记匹配。特别是，我们引入了一个由预定义安全阈值 C 界定的增量计数器 c，该阈值与重复字符串串联在一起，如下所示：

$$\{str \| 0, \bullet\bullet\bullet, str \| C - 1\}$$

因此，t 等于 F_4（K_4， str ‖ c），而 s 等于 F_2（K_2，str ‖ c），而对于动作插入，t_4 和 t_2 的计算方法相同。和加密。如果 C 足够大，则在一个连接中重复的字符串将映射到不同的标记。否则，计数器 c 将重置为 0 。即使在那种情况下，由于混淆了出现在有效载荷中的出现的字符串的分布，所以安全强度仍然得以提高。

同时，我们通过广播加密[20]实施对端点的访问控制机制，这是受可搜索对称加密方案[12]中多客户端支持的启发。仅授权端点可以生成有效令牌。并且，如果检测到恶意端点，它将被有效地吊销。

详细协议：给定密钥生成函数 KGen （1^k），广播加密方案 BE（Enc ，Dec ，Add ），规则集 R ，提出的算法 Build ，针对单个连接的单令牌匹配协议如下所示：：•我的化：

1) *AS* 生成{K x，K 2，K 3} KGen（1 k），其中 *k* 是安全参数，并构建 F Build（K，K 2，R，*C*）。
2) 对于两个注册端点 S 和 *R*，*AS* 计算状态信息 *ST* BE。Enc（K 3，**P**，r），其中 **P** = { *S*，*R*，*MB* }，和- *[R* 是一个 ķ 位随机串。然后，它生成 *K* s BE。加（K 3，S）和 *K* R BE。加(K 3，R)。
3) {F，R，*ST* }被发送到 *MB*，并[KP b ķ 2，K 小号，*ST*，*C* }和{ ķ 1，*K* 2，ķ - *[R*，*ST*，*C* }被发送到小号和- [R 分别。
4) S 和 R 运行标准的 SSL 密钥交换协议。

- P 重-处理：

1) S 解析数据包的有效负载，并将每个字符串转换为令牌(t，s），其中 t = *F* 1（*K* 1，str ‖ c），s = *F* 2（*K* 2，*str* \\ *c* ），则为每个不同的字符串缓存 *c*，并在出现另一个重复的字符串时递增 *c*。
2) S 计算 r BE。分解（K 小号，ST），以及一个G（R，T ‖ S），其中 *G* 是由 r 键控的伪随机置换。此后，S 将 *a* 发送到 MB
3) S 用 SSL 密钥加密流量并将其发送到 MB 。

- 我 NSPECTION：

1) MB 保留加密的流量，并为每个接收到的 *a* 计算 t ‖ s = G -1（r，*a* ）。然后，它生成吨 x = *P* 我（*T*， 1），吨 2 = *P* 2（*T*，2），并进行 E®小号为每个条目 ê在 Ť 我[吨我]和 Ť 2 [T 2] 。
2) 只有当涉及到一个可疑的字符串的令牌匹配，则动作的行为将被经由 XOR 操作回收，或令牌标记合法。当数据包中的所有令牌均合法时，该数据包将被允许通过。

- V ERIFICATION：

1) MB 连续地通过密码散列函数，例如，SHA256 生成用于间歇传递令牌的安全消化，并与加密的流量将它们发送 ř 。
2) R 解密流量，根据与 S 相同的原理生成令牌，并计算摘要以进行验证。
3) 如果摘要不相同，则认为 S 是恶意的。兆字节通知并撤销小号：ST ' BE。Enc（K 3，**P**，r '），其中 **P** = {R，MB }，并且 r '是一个新鲜的 k 位随机字符串。

注意：用于解析有效载荷的现有原理，例如基于定界符的分段[5]和基于窗口的 *n-gram* 分析[11]，可以在预处理过程中应用于端点，以便于对二进制和文本数据进行有效检查。在本文中，我们为实验实现了这两个原理，更详细的讨论可以在[5]中找到。我们还指出，上面列出的协议只是基本的操作概述在实践中，根据实际将流量重定向到外包中间盒的不同选择例如简单的流量退回，IP 或 DNS 重定向，可能会在协议级别上进行进一步的变化。

C. 具有属性的安全检查

实际上，检查规则通常包括给定可疑字符串的指定属性。它们可以分类为分组字段或分组有效载荷中的位置范围。对于这些规则，单令牌匹配将不再起作用。一种简单的方法是揭示属性

如 BlindBox [5]所采用的那样，将其设置为 MB，以便 MB 可以在令牌匹配后执行后检查。

但是，我们观察到将它们暴露于 *MB* 会违反规则的保密性和流量有效载荷的机密性。例如，在下面的规则中显示“ http_stat_code”将告诉 *MB* 该规则正在检查 HTTP 的雕像代码。由于相应的消息空间是有限的，因此规则字符串和匹配的有效负载可能会受到影响。

Snort 规则＃746：警报 tcp $ HTTP_SERVERS $ $ HTTP_PORTS- > $ EXTERNAL_NET $任意（流：to_client，已建立；内容：“ 403”；http_stat_code;）

此外，揭露以下规则中指出的检查位置也会威胁企业的隐私，因为这种检查总是检查特定网络协议和应用程序的脆弱性[11]。如果令牌匹配，则 *MB* 会准确告知端点运行哪些应用程序或协议。

Snort 规则＃3235：警报 tcp $ EXTERNAL_NET $任意- > $ SMTP_SERVERS $ 25（流：to_server，已建立；内容：“ EMF”；深度：4；偏移量：40；）

为了确保对规则的强力保护，我们对令牌生成进行了定制，在令牌生成中，检查属性被连接到规则字符串。因为现在通过伪随机函数将属性与字符串一起转换，所以在检查仍然可以正常运行的同时对其进行了保护。明确地，令牌生成如下：

（t，s）= （*F* i（*K* i，*str* ‖ c ‖ field），F 2CK 2，*str* ‖ c ‖ field））

值*字段*是“http_stat_code”为前者的例子规则。关于范围在有效载荷为后者的例子的位置，所有可能的位置应在值来指定的*字段*，我。e。，{ *t* 乞求 ‖ s beg，* * *，*t* endW *s* end}，其中

（*t* i，*s* i ）=（F i（K i，str ‖ c ‖ i），F 2（K 2，str ‖ c ‖ i）），i€ *[beg*，*end] beg* 等于到*偏移量*，并结束等于*求+深度-的 sizeof* （*STR* ）。因此，总*深度-的 sizeof* （*STR* ）重复的行为 s 应该存储在过滤器中。同时，要求端点以相同的方式生成令牌，可以检测到相应的可疑字符串。

D. 安全的多条件检查

一些规则同时检查多个字符串以减少误报[11]。原因是许多漏洞或恶意行为是在多种情况下发生的。因此，我们的系统还应该支持多次令牌检查，如下所示。

Snort 规则＃127：警报 tcp $ EXTERNAL_NET $任何- > $ HOME_NET $ 21（流程：to_server，已建立；内容：“ RETR”；内容：“ passwd”；）

一种简单的方法是将规则 *id* 附加到其操作上，即 *id* \\ *act*。恢复的 *ID* 将告诉您匹配的令牌是否与同一规则相关。在这里，我们假设 *id* 是化名，它不提供有关规则内容的任何信息。但是，仅加密 *id* \\ *act* 不符合安全保证。如果有任何令牌

如果匹配，*MB* 将知道该操作，而不必要地显示了更多信息。在这里，我们的目标是与以前的设计具有相同的安全强度。即，仅当规则中的所有相关令牌都匹配时，响应操作才会恢复。否则，将不会显示任何内容。

通过秘密共享进行加密：为了克服上述安全问题，我们采用了一种有效的（n，n）秘密共享方案[21]。具体来说，给定具有 *n* 个字符串的规则，其动作将被视为机密。然后 N - 1 随机串*{ p* ₁，•••，*P* ₙ₋₁ ₙ*}*与相同位长度生成行为，和 p ₙ = P ₁ ®，•••，©p ₙ₋₁ ₙ®行为，其中每个 p ₄是行为的一部分。因此，基于从可疑字符串之一产生的令牌 t 来放置每个份额。仅当获得所有 n 股时，才会按照以下方式恢复行为：

行动= P 我®，•••，©p ñ

结果，n 个份额的任何严格子集都无法解密 act 。这种设计保证，当且仅当所有可疑字符串都匹配时，份额才会被收回并且规则操作将被显示。否则，该规则将保密。同时，采用的秘密共享与单字符串检查的规则兼容。在这种情况下，份额将是行动本身。我们强调，每个份额 p 仍通过随机掩码 s 保护。

检查：为了适应基于秘密共享的加密，应相应更新 *MB* 处的检查。算法 2 中说明了该实现。给定传入令牌（t，s），MB 将计算 t ᵢ和 t ₂，如果令牌通过 XORing s 与存储桶 T ₁ [t 中的每个条目 e 匹配，则尝试恢复份额。ᵢ]和 T ₂ [t ₂] 。同时，MB 利用内存中的关联映射 M 执行解密。特别地，将 XOR 运算后的结果解析为 x ‖ y ，其中 *x* 的 id 长度相同，y 的每个份额 p 长度相同。然后，一对（X，Y）被插入到中号。如果 x 存在于 M 中，则表明 x 可以是有效的 id，而 y 可以是份额之一。在这种情况下，MB 执行 y©y ' ，其中 y '是以前的份额。当所有必需的股份进行异或运算，动作行为 将被收回并执行。

E. 对其他规则支持的安全检查

跨规则检查：通常，某些规则可能包含相同的可疑字符串，因为某些敏感关键字出现在不同的上下文或活动中。例如，"密码"属于 Snort 社区规则[11]的十多个不同规则。为了支持对相同可疑字符串的跨规则检查，我们基于加密过滤器的设计提出了一种新颖的构造。高级思想是引入另一个计数器来区分给定字符串的不同规则，并使用标志来指示最后一个匹配规则。

回想一下，P ᵢ（t，1）和 P ₂（t，2）事先是根据令牌 t 计算出来的，以寻找过滤器中的可用条目。取而代之的是，计算 P ᵢ（t 1 ‖ c ᵣ）和 P ₂（t，2 ‖ c ᵣ），其中 *c ᵣ*是与每个不同的可疑字符串关联的计数器，以区分相关规则。此外，产生新的遮罩 P ₃（s，cᵣ）以进行加密，其中 P ₃为 PRF。通过给出（t，s），*MB* 可以递增 c ᵣ 进行检查，但不知道何时

算法 2 通过秘密共享检查加密

输入：{t ‖ s}：流令牌；***M*：**关联图。
输出：{act}：触发的动作。

1：对于每个传入令牌（t，*s*），计算 t ₁ = *P- ᵢ ft*，1）和 t ₂ = P 2（t，2）；
2：执行 e®s，Ve in T ₁ [t ₁]和 *T ₂ [t ₂]*，并将结果解析为 x ‖ y，其中 | x | = | id | 和 | y | = | p |；
3：插入（*X*，*Y*）到中号。如果 *X* 存在，执行 *ÿ* ® *Y*"其中 *Y*"是与相关联的先前值 X 。
4：一旦有效动作 *行为*已经由 XOR 操作恢复时，*动作* 被执行。

停止。因此，我们引入一个标志并将其附加到动作份额以及 id 上。因此，当正确恢复条目中的标志时，*MB* 将知道它是否应该继续。每个条目的构造如下：

P 3（s，c ᵣ）©（id ‖ p ‖ f lag），标志 G {valid，null}

该标志被分配为有效或 null ，其中有效意味着需要检查更多与令牌匹配的规则，而 null 意味着这是最后一个与令牌匹配的规则。对于具有唯一字符串的规则，该标志设置为 null 。**交叉连接检查：**交叉连接检查是入侵检测的另一种常用方法，例如，检测如下所示的暴力登录。

```
Snort 规则＃1633：警报 tcp $ EXTERNAL_NET $任意- > $ HOME_NET $ 110
（流：to_server，已建立；内容：" USER"；计数 30，秒 30；）
```

此规则将计算" USER"的出现次数，并在" USER"在 30 秒内出现 30 次以上时触发可疑登录活动的警报。为了实现跨不同连接的检查，我们进一步引入了通用消息身份验证代码 mac，该代码随附有用于相等性检查的动作，即 mac = P ₄（r ᶜ，str）‖ time，其中 P ₄为 PRF，*r ᶜ*为所有连接都同意的随机性，时间是时间间隔。因为所有连接的 r ᶜ都相同，所以 P ₄（r ᶜ，str）将与相同的字符串 str 匹配。当 MAC 被回收，*MB* 会启动一个定时器，并且记录的出现 P ₄（R ᶜ，STR）。在这里，我们假定由于云的无限功能，*MB* 处理的令牌处理的吞吐量可以与网络流量的吞吐量相媲美。否则，应调整时间以容忍检查带来的延迟。

讨论：当前，我们的设计不包括对复杂检查的支持，即正则表达式和脚本。与 BlindBox [5]中的处理类似，在解密流量后，需要对此类操作进行后期处理。但是与 BlindBox 允许 *MB* 解密流量不同，我们的设计选择是将警告和可疑数据包发送回 *AS* 出于安全考虑，谁将强制端点交出用于解密的 SSL 密钥，否则它将终止连接。特别是，我们定义了一个名为" pcre"的新操作，并使用正则表达式将其替换为规则中的前一个操作。只要恢复了" pcre"，数据包就会被发回。我们确实承认这是我们系统的局限性，但是我们认为只有很小的一部分

匹配的部分数据包需要进行复杂的检查，例如，在指令检测流量转储中不到1%，我们的实验中显示的数据包超过3000万

F. 初始化优化

如前所述，每个连接的初始化都需要新的私钥来构建加密的筛选器，以便隐藏不同连接之间令牌的相等性。尽管花费短时间，例如，具有超过3000条规则的规则集大约需要100毫秒，但是对于端点而言，建立加密连接仍然是额外的费用。为了消除初始化期间的开销，我们利用 *AS* 为多个传入连接预先构建了一组加密过滤器。明确地，将函数 Build 与初始化解耦，并在异步过程中定期为 *MB* 构建筛选器。对于实施，*AS* 记录已建立的连接数，并在连接数达到使用的过滤器数之前准备另一组新的过滤器。结果，对输入连接的检查变得无缝连接到期后，*MB* 将驱逐过时的筛选器。

IV. 小号 ECURITY 一 NALYSIS

在本节中，我们将进行严格的安全性分析，以证明*MB*在通过多个不同的连接执行DPi时无法学习流量有效载荷和规则内容。具体来说，我们将证明第 III-B 节中描述的单字符串检查协议 *P* 对于适应性对手是安全的，然后表明其余设计仍然可以保证规则和有效负载的机密性。

首先，我们从[12]，[13]逐字拟定基于仿真的安全定义。给定MB的定义清晰的视图，即状态函数L，我们证明 *P* 对自适应选择关键字攻击是L安全的。也就是说，除了对可疑标记采取的行动外，P 不会产生任何有关有效负载和规则的语义信息而这些可疑标记是由检查“自然揭示”的。定义 1 中的安全性定义以任何概率的多项式时间对手 *A* 都无法区分真实游戏中的真实加密过滤器和模拟过滤器，真实随机标记和模拟标记之间**分别为 Real** A （k）和模拟游戏 **Ideal** AS *（k ）*。

***定义* 1.实数** A（k）：*质询者调用* KGen（1^k）*以输出私钥{* K_1，K_2 *}。对手 A 为挑战者选择规则集 R ，以通过* Build（Ki，K_2，*R）创建一个加密的过滤器 F ，而 A 自适应地发送从一组数据包中提取的多项式字符串。之后，挑战者用相应的令牌响应 A ，而 A 通过 F 处理令牌。最后，A 输出一点。*

理想的 A，小号（K）：甲*选择-* **[R** ，*和一个模拟器小号生成 ˚F 基于* L（**- [R ）**。*然后，A 自适应地发送从一组数据包中提取的多项式字符串。之后，S 用模拟令牌响应 A ，而 A 在 F 上处理令牌。最后，A 输出一点。*

在提出证明之前，我们将 *MB* 的视图形式化：L（**R**）=（| F |，| e |，*d*，{t ‖ s}$_q$，*{ act }*$_m$），其中| F | 是加密过滤器的大小，| e | 是过滤器条目的比特长度，d 是条目的一个桶的数量，{吨‖ S}$_q$是 *q* 被自适应地生成的令牌，和*{行为}*$_*$的*米* 动作中回收 q 令牌。然后，我们有以下定理。

***定理* 1.** *如果 F_1，F_2，P_1和 P_2为 PRF ，则* P *是针对自适应选择关键字攻击的 L-安全性。*

*证明。*首先，S 模拟具有与实 F 相同大小的过滤器 *F*，不同之处在于过滤器中的每个条目都存储一个| e |。位随机字符串。在 *F 中，*每个存储桶还包含 d 个条目。回想一下，F 中的每个条目都是用随机掩码加密的，或者是用随机填充填充的。由于伪随机性˚F $_x$，˚F $_2$，P $_x$和 P $_2$，小号不能区分˚F 从˚*F* 。然后 S 模拟令牌和检查结果，即动作。对于第一个令牌 t ‖ s，如果没有从 2 个哈希表中访问的 2 个存储桶中恢复任何操作，并且结果仍然是随机字符串，则 S 将生成随机字符串 t ‖。“和操作随机预言^*h*，取代的 PRF 找到两个水桶˚*F* 。*S* 取消掩盖存储桶中的条目，并且也不会显示任何操作。因此，A 无法区分真实令牌和结果与模拟令牌。如果操作 *行为*被回收，小号 操作^ h 生成 s'，这将掩盖随机选择的条目之一：*act* = s'®e 。对于随后的令牌，如果令牌出现在之前，则 S 使用先前模拟的相同令牌，或遵循模拟第一个令牌的相同方式。因此，A 无法区分模拟令牌和结果与真实令牌。毕竟，**实数** A （k）和**理想数** AS（k）的输出是无法区分的。 □

应用基于对称密钥的可搜索加密将使 *MB* 知道重复的令牌，因为它们是通过确定性单向函数生成的。如建议的那样，我们的设计通过交换空间来提高安全性。对于 与计数器关联的 *C* 副本，每个规则字符串都是重复的。然后，*MB* 将看到在连接中出现的 C 个重复字符串中生成的不同令牌。我们注意到，如果规则要求进行跨规则检查，则恢复的通用 MAC 某些可疑的字符串将揭示相等性，即使对于不同的标记也是如此。但是，功能是必需的，例如，计算跨连接的重复可疑字符串。而且，无与伦比的代币的平等性仍然受到保护。

V. É XPERIMENTAL Ë 估值

实验设置：为了进行评估，我们选择了两个开源规则，即 Snort 默认规则集[2] 和 ETOpen 规则集[3]，以及两个入侵检测流量转储，即 DARPA99[4]和 iCTF08[5]，其总大小超过 3 x $10^{7个}$数据包。我们在 C ++中实现 Middlebox 模块和 Endpoint 模块，并在 C ＃中实现带有 UI 的规则解析器。然后，将中间盒模块部署到 Amazon Cloud 上不同实例模型上，即“ c4.2xlarge”，“ c4.4xlarge”和“ c4.8xlarge”，然后安装

[2] Snort 稳定版本：在线 https://www.snort.org/downloads。
[3] ETOpen 规则集：在线，网址为 https://www.snort.org/downloads。
[4] DARPA 数据集：在线，网址为 http://www.ll.mit.edu/ideval/data/。
[5] iCTF 通信量：在线，网址为 http://ictf.cs.ucsb.edu/pages/the-2008-ictf.html。

（a）初始化时间　　（b）检查量　　（c）代币开销　　（d）引入的延迟

图 3：我们提出的系统的性能评估

iCTF08	0。7%	93。8%	6。2%	0。2%	96。4%	3。6%
DARPA99	0。2%	15。3%	84。7%	0。3%	97。6%	2。4%

表 I：支持匹配的可疑数据包的统计信息。“ match”：匹配的报文在流量转储中所占的百分比；“支持”：我们的设计直接支持的总匹配数据包中的数据包百分比；“ pcre”：需要正则表达式的数据包的百分比。

具有 Intel Core i7 CPU 和 16GB RAM 的 Macbook Pro 上的终结点模块。

有效性评估： Snort 和 ETOpen 的设计直接支持的规则百分比分别为 69％和 52％，不包括“ pcre”规则。如表 I 所示，在大多数情况下，我们的设计可以处理 90％以上的匹配数据包，其中其余数据与“ pcre”规则匹配，需要对其进行解密以进行后处理。但是，在选定的流量转储中，可疑匹配的数据包总数不到 1％即使某些数据包需要在 *AS 处*解密，它们仅占总流量的一小部分。请注意，检查准确性取决于 DPI 系统如何解析数据包有效负载，在纯文本域中对此进行了深入研究。类似于 BlindBox [5]，我们采用基于定界符和基于窗口的字符串分段。

性能评估：首先，我们评估由加密过滤器引入的成本。在表 II 中，显示了从规则集中提取的字符串操作对的总数。为了隐藏检查位置，会为每个可疑字符串生成多个字符串操作对，并带有多个可能的位置，因此，对的数量要比规则的数量大得多。回想一下，每个加密的条目都存储 *id* \\ *act* \\ *flag* \\ *mac* 其中我们设计中的每个段的长度为 4 个字节，并且该条目被 16 字节的随机掩码掩码，因此每个条目的长度为 16 字节。 。负载率 95％，针对具有近 2 万条规则的 ETOpen 加密的过滤器的成本为 1.9 MB，而针对具有约 *3 K* 规则的 Snort 的加密过滤器的成本仅为 129 KB。建立连接也是从 AS 到 *MB* 的带宽消耗。图 3-（a）显示了每个连接的初始化时间，该时间主要由加密过滤器的建立时间决定。在这里，我们根据不同的规则集报告成本，即 Snort 大约为 0.1 s，ETOpen 小于 2 s。如前所述，此过程可以与连接初始化分离，因为可以预先构建许多过滤器因此，*MB* 可以执行

规则集	# 条规则	/ 7 p; urs	过滤器尺寸
喷鼻息	3240	7678	129 KB
公开赛	19528	115503	1945 KB

表 II：每个连接的加密过滤器的空间和带宽消耗。负载系数设置为 95％。建立连接后立即进行检查。

回想一下，我们建议对重复的字符串操作对进行索引，以隐藏重复的标记。为了了解开销，我们用文本内容分析了来自两个流量转储的数据包，即，对单个数据包中不同字符串的最大出现次数进行计数。结果表明，在总共 10268 个不同的字符串中，有 91％出现在一个数据包中的次数不超过 3 次。然后，将每个可疑字符串的重复数设置为 3 ，即 Sec 中定义的计数器 c 的阈值。III-B。因此，过滤器的尺寸变为 2 时间更大。将来，我们将探索最大程度地减少开销，同时尽可能保护令牌分配。

数据包处理的吞吐量如图 3-（b）所示。我们在三种 AWS 实例模型下测量吞吐量。由于我们的加密过滤器实现了快速并发的查找，因此处理一个令牌的等待时间小于 10 s，并且吞吐量高达每秒 63 x 10 ⁶令牌。在流量转储中，每个 HTTP 数据包的平均令牌数为 35 。对于“c4.8xlarge”实例保持 500 个连接，用于一个连接的吞吐量可以达到 3 ， 600 每秒的数据包。为 3 000 连接，每个连接的吞吐量仍然达到 600 每秒数据包。并且使用多个实例将进一步提高吞吐量。

为了使 *MB* 能够对加密的流量执行 DPI，要求端点解析数据包内容并生成令牌以进行检查。生产和传输这些令牌确实会带来计算和带宽开销。在这里，我们实现了基于定界符和基于窗口的字符串分割，并将滑动窗口的大小设置为 6 个字节。我们还截断 HMAC-SHA1 的输出作为 10 个字节和 16 个为令牌字节*吨*和掩模小号分别。我们还观察到带有属性的规则通常用于检查网络协议的报头[11]，因此我们的客户端需要为报头中的每个字符串生成三个令牌以确保正确性，即带有分组字段的令牌，令牌。带有偏移量，令牌仅带有字符串。对于其他字符串，例如 HTTP 正文中的字符串，仅生成一个带有该字符串的令牌。

如图 3-（c）所示，超过 90％的数据包引入的带宽开销在 3 倍到 6 倍之间变化。

原始数据包大小的条款。图 3-（d）报告了网络等待时间，包括给定数据包的令牌生成时间，传输时间和处理时间。我们在两个带宽设置（即 20 Mbps 和 1 Mbps，延迟为 100 ms）下评估每个数据包引入的延迟。对于高速网络，几乎所有数据包引入的延迟都小于 100 毫秒。在恶劣的网络条件下，超过 85％的数据包的延迟仍小于 200 毫秒，因为大部分 HTTP 数据包的有效载荷大小很小，每个数据包仅输出数十个令牌。简而言之，我们的系统可以执行具有实际性能的安全 DPI。

六。[R 心花怒放 w ^ ORK

大多数现有的入侵检测系统无法对加密流量进行全面分析[6]。先前的中间盒对路径[1]，[22]中间的流量进行解密，这会破坏有效负载的机密性，并且可能构成中间人攻击[10]。另一项工作[17]对加密流量进行统计分析，以提取端点活动的特征和特征，但是这些机制无法检测到复杂的语义攻击，因此限制了入侵检测系统的能力。

最近，一个名为 BlindBox [5]的中间盒设计启用了 HTTPS 流量上的 DPI 服务。一种改进的设计[23]扩展了 BlindBox 以支持更广泛的中间盒功能，并且还考虑保护使用基于云的中间盒服务的企业的隐私。与这些设计不同，我们的设计通过以定制方式处理不同类型的规则来确保对规则集提供更精细的保护。辅助信息（如检查位置，字段等）也受到保护，可以利用这些信息来破坏规则和有效负载的机密性。另一方面，新提出的防火墙设计[7]在过滤非加密流量时混淆了防火墙规则。另一个安全的中间盒设计[15]也旨在保护中间盒服务提供商的流量和规则。但是以上设计使用了沉重的加密工具，例如多线性映射[7]和同态加密[15]。因此，尚不清楚它们是否可以达到我们设计所能达到的相同水平的实际性能。

我们提出的设计还与大量可搜索的加密方案（仅举几例）[12] [13]有关。他们研究了有关如何对加密文档启用私人关键字搜索的问题。但是，如上所述，直接应用它们并不能提供对检查规则的全面支持，也不能提供具有高吞吐量和内存效率的安全设计。

七。Ç ONCLUSION

在本文中，我们设计了一个系统，该系统使外包的中间盒能够执行数据包检查，同时保护数据包的内容和检查规则。我们首先将问题表述为加密的字符串匹配，然后提出一个加密的过滤器，该过滤器安全地存储从规则中提取的加密的字符串操作对。之后，端点解析数据包内容并生成随机令牌，因此中间盒可以在过滤器上对其进行处理以进行检查。我们的设计支持广泛的检查规则，对真实规则集和流量的评估表明，我们的系统可以有效地检测大多数可疑数据包。将来，我们将研究在加密流量上处理正则表达式规则的方法，并研究有效的机制来验证 Middlexbox 的行为。

一个 CKNOWLEDGMENT

这项工作得到了香港研究资助局（项目编号 CityU 138513）中国自然科学基金会（项目编号 61572412）和 AWS 的教育研究项目资助的部分支持。

[R EFERENCES

[1] J. Sherry，S。Hasan，C。Scott，A。Krishnamurthy，S。Ratnasamy 和 V. Sekar，“使中间盒成为其他人的问题：将网络处理作为云服务”，*Proc。ACM SIGCOMM*，2012 年。

[2] A. Gember，R。Grandl，J。Khalid 和 A. Akella，“*Proc。中的*软件定义中间盒网络框架的设计和实现”。*ACM SIGCOMM*，2013 年。

[3] Proc。中的 H. Jamjoom，D。Williams 和 U. Sharma，“不要称它们为中间盒，而应称它们为中间管道”。*ACM HotSDN*，2014 年。

[4] Google，“ HTTPS 作为排名信号”，http：// googlewebmastercentral。blogspot.hk/2014/08/https-as-ranking-signal.html，2014 年。

[5] J. Sherry，C。Lan，RA Popa 和 S. Ratnasamy，“盲箱：用于加密流量的深度包检查”，*Proc。ACM SIGCOMM*，2015 年。

[6] K. Skarfone 和 P. Mell，“入侵检测和预防系统指南”，*美国国家标准技术研究院，网址为：http://ffcsrc.nist.gov/publications/nistpubs/800-94/SP800-94 .pdf*，2007 年。

[7] Shi。J.，Y。Zhang 和 S. Zhong，“保护隐私的网络功能外包”，arXiv 预印本 arXiv：1502.00389，2015 年。

[8] AR Khakpour 和 AX Liu，*Proc 中的*“迈向基于云的防火墙的第一步”。*IEEE SRDS*，2012 年。

[9] Z. Zhou 和 T. Benson，*Proc。中的*“建立 HTTPS 和具有 QoS2 的 MiddleBox 的安全场所”。*ACM HotMiddlebox 的照片*，2015 年。

[10] LS Huang，A。Rice，E。Ellingsen 和 C. Jackson，“ *Proc 中的*“在野外分析伪造的 SSL 证书”。*IEEE S&P*，2014 年。

[11] Snort，“开源入侵防御系统”，https：//www.snort。组织/，2015 年。

[12] R. Curtmola，JA Garay，S。Kamara 和 R. Ostrovsky，“可搜索对称加密：改进的定义和有效的结构”，《*计算机安全学报*》，第 1 卷。19 号 5，第 895-934 页，2011 年。

[13] D. Cash，J。Jaeger，S。Jarecki，C。Jutla，H。Krawczyk，M.-C。Rosu 和 M. Steiner，*Proc 中的*“大型数据库中的动态可搜索加密：数据结构和实现”。*NDSS*，2014 年。

[14] B. Fan，D。Andersen，M。Kaminsky 和 M. Mitzenmacher，“布谷鸟过滤器：实际上比开花效果好”，*Proc。ACM CoNEXT*，2014 年。

[15] L. Melis，HJ Asghar，ED Cristofaro 和 MA Kaafar，“外包网络功能的私有处理：可行性和构建”，Cryptology ePrint 档案，2015/949 年报告，2015 年。

[16] SK Fayazbakhsh，MK Reiter 和 V. Sekar，*Proc 中的*“可验证的网络功能外包：需求，挑战和路线图”。*ACM HotMiddlebox 的报告*，2013 年。

[17] A. Yamada，Y。Miyake，K。Takemori，A。Studer 和 A. Perrig，“ *Proc 中的*用于加密 Web 访问的入侵检测”。*IEEE 高级信息网络和应用研讨会* 2007 年。

[18] E. Bertino 和 G. Ghinita，“关于检测和防止内部人员泄露数据的机制：主题演讲文件”，载于 *Proc。ASIACCS*，2011 年。

[19] R. Pagh 和 F. Rodler，“布谷鸟哈希”，*J。算法*，第 1 卷。51 号 2，第 122-144 页，2004 年。

[20] A. Fiat 和 M. Naor，*Proc 中的*“广播加密”。*CRYPTO*，1994 年。

[21] B. Schneier，《*应用密码学：协议，算法和源代码》，C。John* Wiley& Sons，2007 年。

[22] Squid，“ Squid 缓存功能：HTTPS（HTTP 安全或 SSL / TLS 上的 HTTP）” http：//wiki.squid-cache.org/Features/HTTPS，2015 年。

[23] C. Lan，J。Sherry，RA Popa 和 S. Ratnasamy，“ Mbark：将中间盒安全外包到云中”，*Proc。USENIX NSDI*，2016 年。