一种共享 IP 流记录分布式平台①

张国良，丁岳伟

(上海理工大学 光电信息与计算机工程学院，上海 200090)

摘 要：开发了一种基于多个机构之间流量记录共享的分布式平台，平台内的机构在其允许范围内把采集的流记录存储到单独的主机中，然后提供给他人访问服务器的网关。由于保密问题，此平台使用一个预保留、加密和匿名算法，以便流量轨迹数据交换遵从不同规范。

关键词：分布式平台；流记录；保密问题；匿名算法；流量轨迹

Distributed Platform for Sharing IP Flow Records

ZHANG Guo-Liang, DING Yue-Wei

(University of Shanghai for Science and Technology, Shanghai 200090, China)

Abstract: This paper develops a Distributed Platform for Sharing IP Flows (DipSIF) based on NetFlow records between multiple institutions. It is assumed that NetFlow traces collected by each participant are archived on separate storage hosts within their premises and then made available to others using a server that acts as a gateway to the storage. Due to secrecy issues the platform presented here uses a prefix-preserving, cryptography-based, and consistent anonymization algorithm in order to comply to different regulations determining the exchange of traffic traces data.

Keywords: distributed platform; NetFlow records; secrecy issues; anonymization algorithm; traffic traces

1 引言

IP 流量记录广泛应用于不同领域。一方面，轨迹用来观察流量特性以便改进流量分析工具；另一方面，基于实时流量轨迹评估新的入侵或异常探测算法。设计 IP 流量的新演算法时，一个重要缺陷是这种算法因缺少大量的实时流量轨迹不能被精确地评估。以常的流量轨迹只是本地使用（例如流量数据来自实验室网络或者校园网），很少为他人使用，因此研究人员远程访问流量轨迹所面临的是技术和法律问题。技术问题，一方面，缺少访问不同存储器的统一方法；另一方面，在访问流量轨迹之前，缺少对流量轨迹的预处理，往往会出现用于分析的流量轨迹只是所下载流量轨迹数据的很小一部分的现象，造成数据资源的浪费。在法律问题上，主要涉及到流量轨迹数据的保密和安全。因此，为了解决这些问题，需要成的部署一个流量轨迹共享平台。

本文介绍了共享 IP 流记录的分布式平台开发(DipSIF)。 DipSIF 对从不同机构所采集的流记录实现共享，以便使研究人员在不同地方使用。为了实现流量轨迹数据共享，平台提供了定义好的接口，允许研究人员能够以访问存储在本地的 NetFlow 记录的方式远程访问存储在网络主机中的流记录。同时，为了履行保护数据的保密性和安全性、并遵守网络法规的协议，该平台使用了预保留、基础加密、一致性匿名算法。该平台已实现，并且进行了相关测试。

2 流量采集与记录匿名

设计 DipSIF 最重要的两个方面是：（一）采集和存储流量轨迹，（二）匿名分享。

2.1 流记录的采集和储存

一条流记录包含流的特定属性（例如，包和字节在流传输的总数）和流的普遍特性（例如，源 IP 地址和目标 IP 地址）。通常情况下确定一条 IP 流记录至少有以下特定的值：源 IP 地址，目的 IP 地址，源端口，

① 收稿时间:2010-09-23;收到修改稿时间:2010-11-06

目的端口，协议类型。采集和储存 IP 流记录用到不同的解决方法、工具和协议。

NetFlow 属于思科协议，主要提供网络中经过路由设备的数据包流量统计。NetFlow 包含于 Cisco IOS 中，对大多数思科交换机和路由器，NetFlow 能够被用于网络监控、计费和安全技术。NetFlow 能够识别输入、输出 IP 数据包的数据包流，它不包含任何连接设置协议（在路由器间或到任何其他网络设备或终端站）。NetFlow 对现有网络是完全透明的，其中包括终端站和应用软件、网络设备（诸如 LAN 交换机）。此外，用户在每个内网设备上可独立完成 NetFlow 捕获和输出，NetFlow 并不需要在网络中的每个路由器上进行操作。

Nfdump 是一个提供采集、处理和存储网络流数据的工具包。该工具包还提供了一个转换器，用于把 Flow-Tools[1]的 NetFlow 记录转换成 Nfdump 格式文件。Nfdump 流量记录存储在时间片文件中并把时间戳附在该文件名后面。之后，特殊的文件或者合并，或者再次分解，以便进一步分析。

2.2 流记录匿名化

匿名化[2]是以检验发送的 IP 数据包的方式隐藏网络中特定客户端身份的过程。流量轨迹匿名化并不能适用于所有应用程序。例如某些应用程序，计费及流量监控，需要使用原流量分配给特定用户的带宽或检测网络中拥塞出现的地方和原因。一些匿名算法众所周知，但没有用于流量轨迹匿名化标准的方法。以下是在不同工具中所遇到的匿名化技术：

截断：定义了 IP 地址匿名化的基本类型。用户通过对 IP 地址截断获的字段中选取 n 位。例如，截断每组 8 位将取代相应的 C 类网络地址。

随机排列：使用此方法，一组 IP 地址的随机排列应用于 IP 地址置换。一个 32 位分组密码代表了置换的 IP 地址空间的子集，使用随机哈希表真正的实现了随机排列。因此，在匿名化过程中可能产生任何的排列，不只是可能产生排列的子集。

前缀保留匿名：这种排列的特殊性在于拥有一个独特的结构保留特性：两个匿名化的 IP 地址的 n 位前缀匹配，当且仅当非匿名地址 n 位前缀也匹配。这是通过用户提供的密码完成，这种方法产生一个加密密钥，以确定排列的应用。这种方法允许基于多个匿名者之间共享一个公共值的一致性映射匿名。

基于这些机制，在实践中使用了一些匿名工具。TCPdpriv 实现了 IP 地址的基表和前缀保留变换。为了保持匿名的一致性，以<原 IP 地址,匿名化的 IP 地址>的格式绑定 IP 地址进行存储。在任何时间对一个新的 IP 地址匿名化，这将与已存储并且绑定的原 IP 地址比对，以求得最长前缀匹配。对于那些存储在绑定表中无前缀的剩余位，使用一个伪随机函数。TCPdpriv 仅仅通过对包头操作就可以删除敏感信息。当整个 IP 负载因协议丢弃，则 TCP 和 UDP 的负载也随之丢弃。该工具提供了匿名多层化，以便执行更严格的匿名。TCPdpriv 有两个主要的缺点：其一，仅仅提供在线匿名化；其二，在分布式环境中不能以一致性的方式对并发性轨迹匿名化。

CryptoPAn[3]是一种加密网络轨迹的匿名工具，此工具采用前缀保留特性，没有 TCPdpriv 的使用限制，允许基于多个匿名者之间共享一个公共值的一致性映射匿名，其在线对流记录匿名化的速度很快。AAPI（匿名化应用程序接口）是一种基于 C 的匿名库。这种工具提供快速在线匿名化，同时支持合成、分解和对 NetFlow 轨迹在保存到 tcpdump 中前的不同版本进行过滤。CANINE[4]对于不同的 NetFlow 记录采用不同的方法：IP 地址截断，随机排列和前缀保留匿名。AAPI 和 CANINE 提供了比 IP 地址匿名更多的匿名方式，同时也支持隐藏 IP 地址之外的部分。

3 设计

为了使不同机构之间实现共享，访问控制和在线流量匿名是 DipSIF 架构设计的重点。基于上述需求，由此产生的架构完全是分布式的。

3.1 需求

基于分布式跟踪处理的研究，DipSIF 的关键需求分析如下：

NetFlow 数据的远程存储检索：关键点是为研究人员提供一个应用程序编程接口（API），用于查询远程 NetFlow 存储体里的流记录。

访问控制：该平台将提供一个认证机制和对 NetFlow 存储数据的访问控制。每个存储提供者应当是在其位置独立地决定用户或用户组将访问所采集的流量轨迹。

在线匿名：任何对 NetFlow 记录的请求将会以匿名的方式发送给请求者。这种匿名方式是在线完成的，

并且经过处理的IP地址至少以一种前缀保留方式存储在 NetFlow 记录中。

加密：所有分布式组件之间的通讯渠道通常是不值得信任，因此对所有的通信应加密确保其安全性。

3.2 框架

通过这些需求设计了三层模式架构：第一层，提供检索服务并且独立运行的存储层；第二层，具有客户认证和转发功能的服务器层；第三层，提供访问 NetFlow 数据库的 API 接口的客户端层。安全套接字层提供的加密技术，确保了客户端与服务器之间的认证、数据传输的完成性以及服务器与客户端之间通信的安全性。来自客户端包含选项过滤以及其他控制消息的查询会产生匿名化的 NetFlow 记录，并把结果返回给请求者。

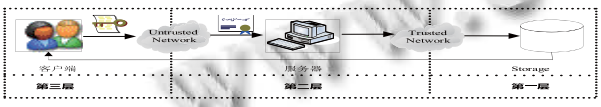

图 1　三层框架

第一层（和第三层）清楚地展现出服务器（和客户端）的功能，第二层表现双重作用：做客户端时，从本地存储器请求 NetFlow 记录；作为服务器端时，处理收到的远程客户端请求。图 2 说明了共享平台架构的设计，在此图中可以知道信息在客户端、服务器端和存储节点之间交换方式，也显示了不同的组件之间部署和相互作用。

终端用户根据需要使用 API 建立对 NetFlow 记录检索的应用程序。API 为用户提供了许多方法，这些方法用来控制所请求的数据集（例如过滤器将用于减少数据集、传输速率等）。若要使用 API 的功能，客户端需要事先通过认证。认证成功后，客户端可以请求流数据。在第一步中，请求消息包含所需数据集的类型，且通过 API 中已预定义的进程把创建的请求数据集传递给服务器。

当服务器收到客户端的请求消息，通过使用访问控制模块检查预授权和匿名策略。如果允许用户访问 NetFlow 库，则请求连同匿名策略一起被转发到存储组件，并把此过程生成日志。存储服务在存储体中搜索相应的数据集，并根据接收到的匿名策略进行匿名化，最后依照请求消息进行过滤。重放模块（nfdump 工具实现的原型）直接将结果数据集以 NetFlow 流的形式返回给客户端。

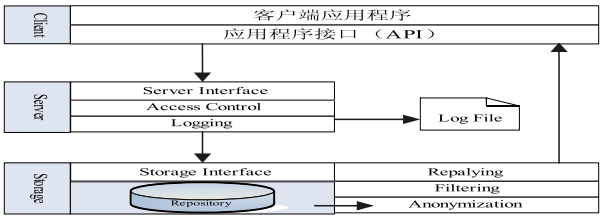

图 2　体系结构

3.3 客户端组件

客户端组件是 DipSIF 的切入点，它包括一个库和能被用于与其他组件通讯的 API。该库支持用户-拥有有效证书 - 请求远程存储体所采集的流轨迹。在此过程中，使用 API 对 NetFlow 记录请求，但是不提供实际检索过程，用户只关心所需要的数据从何处服务器得来，而不是获得方式。

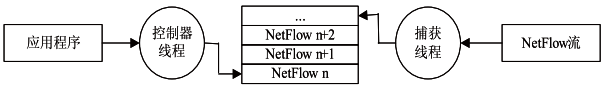

图 3　可控制的捕获单元

客户端库可以缓冲由存储节点重放模块发送的 NetFlow 纪录，同时以可控的速度传递给客户端的应用程序。因此，捕获线程接收流入的 NetFlow 记录并且解码，把这些记录加载到临时的先进先出表中（如图 3 所示）。与之同时，控制器线程根据用户事件从缓冲中取出 NetFlow 记录。除了这个机制之外，请求消息还包含了用以调整由存储组件发送 NetFlow 记录速率的延时属性，这有助于避免接收端缓冲区溢出。在发送一个请求到指定的服务器之前，客户端应用程序把接受者与一个本地未使用端口绑定在一起，与请求信息一起发送。端口号是用来告知存储组件的重放模块将请求数据返回的地址。

请求消息包含以下属性：

用户身份（即认可的和有效的证书）

所需的服务器

返回地址（即 IP 端口）

请求数据（即时间周期）

可选的过滤器（例如，端口或基础协议）

延时选项（例如，每条记录延迟 10 毫秒）

当收到一个请求消息，服务器组件根据认证的结

果给以响应。

3.4 服务器组件

服务器组件作为网关为客户与存储之间提供通信，用于执行客户端身份认证、授权和控制对储存库的访问。每个服务器组件维护它自己的证书库（信任库），该库管理一个用户访问授权证书的列表。在请求转向存储节点之前，请求者需要通过使用私钥匙进行身份认证。公共密钥体系（PKI）共享一个公钥匙是为了利用 SSL 对通讯信道加密。

所有服务器为接受传入的查询使用一个预定义的端口。该服务器接口允许客户端应用程序与服务器实例通过调用远程方法进行通信，这种远程方法是通过使用 RMI 定义的接口。该服务器接口允许检索服务器的实际状况和发送请求信息到存储组件。利用有效以及授权证书，用户可以通过以下各项去请求排序和过滤结果数据集：

时间（即流记录的开始时间及结束时间）

端口号（例如，网络目标端口大于 80）

协议类型（例如 TCP-传输控制协议，UDP-用户数据报协议，ICMP-Internet 消息控制协议）

为了提供访问控制，客户端与服务器端之间的交互需要以证书的方式进行身份认证。只有通过认证的用户将获得相应的权限去检索远程存储库的流量记录。服务器管理员可以查看客户端的证书，并决定证书是否添加到服务器的信任库（参照图 4），该证书可由本地认证权威颁发。

服务器的原型实现支持两组用户，管理员组和认证客户组。前者，包括管理人员，负责维护服务器和存储组件，通过接受新用户的认证书，把新用户注册请求添加到信任库，管理用户帐号，对个体用户和组用户分配匿名设置。第二组用户包括允许访问 NetFlow 数据的客户端。

认证客户端必须持有有效和能被认证的证书，才可能请求流数据，此过程通过客户端应用程序使用 API 进行访问。客户无权修改任何流数据，以及请求访问非匿名的数据。由于服务器组件在所有的流量请求中充当媒介，因此假设另一种授权机制：对于一个给定的用户组，仅仅给予访问子流量记录的（例如属于一个子网的流量记录）或获得一个特定时间段内流记录的权利。

为了监测平台性能和控制数据交换，任何传入请求和传出的数据流在传递过程中被服务器记录到一个文件中。这些由时间戳请求，请求者身份（用户名，用户组和 IP 地址），匿名选项，一段时间内所请求的流记录，过滤器类型的应用以及处理时间组成的信息包含在生成的日志文件中。

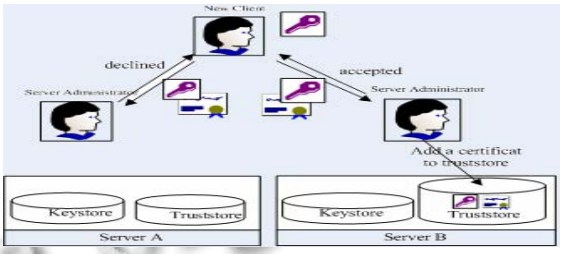

图 4 注册过程

3.5 存储组件

存储组件是负责储存 NetFlow 记录。DispSIF 架构并没定义采集流记录的方式，可以用任何工具将流记录存储到数据库中。NetFlow 记录可以存储到数据库中，或者直接以原始格式存到二进制文件上。因此，DipSIF 的设计与存储类型无关。

存储组件的存储接口提供了用于搜索和检索源自类型独立和存储系统独立（数据库或独立的原始文件）的存储体的流记录集的方法，并能处理异构存储库[5]（如图 5 所示）。基于会产生许多不同的、非兼容的格式的 NetFlow 记录的事实，此存储平台能处理不同类型的流记录。在这种情况下，存储组件把所有的流记录格式转换成 NetFlow 的统一版本格式。

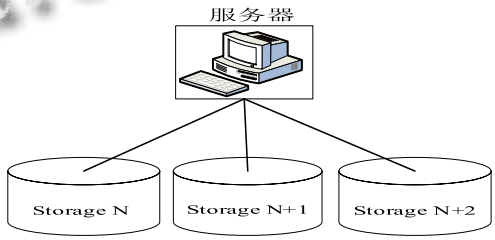

图 5 多个异构存储库连接到一个服务器组件

对于匿名的 IP 数据包，在被转发到请求者之前，采用了 CryptoPan[3]前缀保留的 IP 地址匿名算法。匿名化的关键是建立在身份验证过程由服务器组件创建，一旦认证成功，发送到客户端。流数据是以其原始状态存储，直到离开存储节点时才被匿名化。在匿名化过程中，只有 IP 地址匿名化，而其他的 NetFlow 字段

保持不变。这些字段不被匿名化的原因是为了提供给研究者所需更为真实的网络轨迹。为了增强安全保护，该平台将重建作为匿名流程基础的密钥。对于每一个传入的请求，服务器产生一个会话密钥，在会话过程中对 NetFlow 记录进行匿名化。为了更高的灵活性和在 NetFlow 提供更多的控制，原型的未来版本可能使用诸如端口号匿名的等更多的匿名选项，这种使用方法是依据对流记录使用的应用需求（例如，默认端口号匿名化，一旦应用程序要求原端口号，将使用一种特殊的授权机制）。

4　实现

基于上述设计，原型采用 Java 开发工具包实现。实现的框架如图 6 所示。可以从图中所知，客户端和服务器之间以及服务器和存储组件之间都使用远程方法调用（RMI）进行通信。所有客户端组件和服务器之间采用 SSL 连接进行通信。存储组件和客户端之间的通信反馈使用 NetFlow 协议。原型以 nfdump 格式把流量记录存储到文件中。由于 CryptoPAn[3]是 nfdump 工具链的组成部分，其功能服务于请求，并作为匿名工具使用。

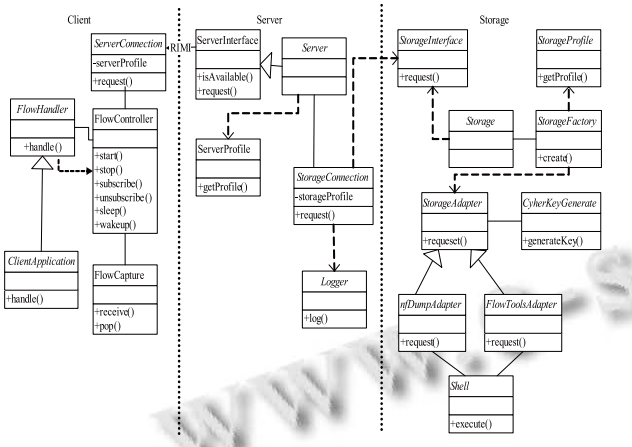

图 6　实现架构

应用程序开发者通过 nfshare 库使用共享平台，扩展 API 中的 FlowHandler 类，并重写 init()和 handle()方法。init()方法是用来向远程存储库发送请求，其实例如下：

```
private void init() throws Exception()
{
    Request request = new Request(
        “Server URL”,//服务器地址
        “your IP”,//auto
        “Port”,//端口号
        “time window”,//系统时间
        “filter options”,//过滤选项
        Delay,//延时(毫秒为单位)
        Limit//限制
    );
    FlowController fc = new FlowController();
    fc.subscribe(this);
    fc.start(request);
}
```

该 FlowController 对象负责处理一个请求并把接收的数据传递给应用程序。当接收到新的流记录时，handle()方法是作为回调函数使用，用来告知应用程序，其函数如下：

```
protected void handle(Flowrecord record)
```

FlowController 以 FlowRecord 对象的方式把流记录传递给应用程序。图 6 所示应用组件看作 ClientApplication 类。

该原型包括 nfDump 和 FlowTools 存储模型。为了支持 NetFlow 的其他存储格式，StorageAdapter 接口需要对特殊格式数据实现并具体化。Shell 类允许执行 Linux 的 shell 命令，不能提供访问流数据的 API 的存储开发包的请求，仅仅处理二进制数据。Shell 命令在存储组件中已被预定义，用户不能直接执行任何 shell 命令。

根据流量记录共享平台的需求分析完成设计，并且进行了实现。其功能分析如下：

检索存储的 NetFlow 数据：DipSIF 提供了用于对远程存储库中流量记录数据检索的 API。该库提供了用于控制请求数据的类型和接收数据流的种类的功能。一个可用的过滤器选项设置根据具体的属性对请求的 NetFlow 记录进行分类，如 IP 协议或端口号。接受流通过延迟 NetFlow 记录、限制数据量和中断一段时间对接收的流进行控制。

访问控制：身份认证使用 PKI 验证，同时根据访问列表进行授权。每个存储供应者，决定用户访问所采集的数据流量的权限。任何客户端需要向管理者提交一个签名的证书，才能够访问相应的 NetFlow 数据。

在线匿名：任何的 NetFlow 记录发送给请求者之

前且离开存储节点时匿名化。原型使用 CryptoPAn 库对 NetFlow 记录匿名，该匿名采用前缀保留的方式。

加密：客户端和服务器之间的通信使用 SSL 连接进行解密。即使存储组件和客户端之间的 NetFlow 传输得不到保护，但 IPSec 可用于保护记录传输。另一个方法是采用 SCTP（流控制传输协议）和传输层安全协议提高 NetFlow 记录传输的安全性。

模块化设计以及存储接口在存储技术上独立，可以在平台中增加用于处理除 nfdump 格式之外其他不同格式的模块。

DipSIF 性能是通过各个组件的性能表现出来。由于服务器和存储组件可能同时服务于多个请求，当有大量的请求到来时，这些组件可能形成瓶颈。服务器组件对每一个到来的请求执行最少处理（验证用户证书，检查授权政策），就不会形成负载；在同一时间处理大量的请求时，存储组件就会形成负载。目前原型独立地处理每一个请求，所以当多个请求到达存储组件时，每个请求会触发 nfdump 文件中可以增加磁盘访问时间的查找。

5 总结

网络轨迹对于研究人员在流量分析领域有着重要的作用。访问这些轨迹存在普遍的问题：其一，管理原因，即网络供应商不能分享其流量数据；其二，缺少用来搜索因特网中已存在流记录的工具。

本文所呈现的 DipSIF 架构可以实现共享 NetFlow 记录。一方面，它允许流量轨迹的所有者与研究人员共享其数据，并提供访问控制机制。同时采用匿名方法，使得实时流量中的 IP 地址不被发现；另一方面，搜索者和协议设计者可以很容易准确的远程访问到存储库位置，并提供用于检索数据的工具。这种控制可以减少数据浪费，研究人员根据具体需要从数据集中下载数据，而不是整体下载。

该平台可以进一步改进，增加其他功能。比如，扩展授权机制，以便让管理员限制外部用户访问流量记录（例如，外部用户仅能访问网站流量）；执行远程存储处理是扩展的又一用途，返回一个累计值，而不是大量的流量记录。使用 SCTP 协议将取代当前用于传输 NetFlow 记录的 UDP 协议，更能增强安全性、可靠性以及拥塞控制意识。

参考文献

1 Plonka D. FlowScan: A Network Traffic Flow Reporting and Visualization Tool. In: 14th USENIX Conference on System Administration, New Orleans, Louisiana, USA, December 2000.

2 Koukis A, Antoniades T, Markatos EP. A Generic Anonymization Framework for Network Traffic. IEEE International Conference on Communications (ICC 2006), Istanbul, Turkey, June 2006.

3 Xu J, Fan J, Ammar M, Moon SB. On the Design and Performance of Prefixpreserving IP Traffic Trace Anonymization. 1st ACM SIGCOMM Workshop on Internet Measurement (IMW 2001), SanFrancisco, California, USA, November 2001.

4 Slagell L, Luo A, Yurcik KW. CANINE: A Combined Conversion and Anonymization Tool for Processing NetFlows for Security. In: International Conference on Telecommunication Systems, Modeling and Analysis, Dallas, Texas, USA, November 2005.

5 Baumgardt N. Design and Setup of a Distributed Storage Repository for NetFlow Records [Thesis]. University of Zürich, Switzerland, March 2008.