基于 Miredo 的 Teredo Relay 性能与安全性分析优化

崔 宇,张宏莉,田志宏,方滨兴
(哈尔滨工业大学网络与信息安全技术研究中心,黑龙江哈尔滨 150001)

摘 要: 隧道是 IPv4 向 IPv6 过渡的一种方式,为处于 IPv4 网络中的 IPv6 孤岛提供 IPv6 互联网接入.本文对 Teredo 隧道的功能、流量和安全方面进行研究,着重分析了 Teredo 隧道中中继设备(relay)的工作原理,指出其节点状态管理机制存在的性能和安全性问题.以 Miredo 中继设备为基础,提出了基于双层查找优化、状态集与时间链扩展、时间链更新优化的性能与安全性增强方法.实验结果表明,正常条件下,插入、更新和回收流程的平均时间缩短了 50% - 60%,提高了中继设备的转发和抗攻击能力.

关键词: IPv6; 隧道; Teredo; Miredo; relay
中图分类号: TP393.08 **文献标识码:** A **文章编号:** 0372-2112 (2014)04-0815-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.04.030

Performance and Security Improvement on Miredo Relay

CUI Yu, ZHANG Hong-li, TIAN Zhi-hong, FANG Bin-xing
(*Research Center of Network and Information Security, Harbin Institute of Technology, Harbin, Heilongjiang* 150001, *China*)

Abstract: Tunneling is one of the main mechanisms for transitioning from IPv4 to IPv6. It facilitates interconnection of IPv6 islands at the edge of IPv4 internet and solves the communication problems between IPv6 islands and IPv6 internet. In this paper the function, traffic and the security of Teredo Tunnel were studied. Additionally, the operating principle of Teredo relay was specifically analyzed. Performance and security problems on "peer status management" in the relay were pointed out. Based on the relay of widespread Teredo software for Linux, "Miredo", methods of two-level-trie, status-timelist-expand and optimize-timelist-update are presented to improve the performance and security of Teredo relay. Experimental results show that under normal conditions, time consumption for processes of inserting, updating and garbage-collecting reduces by 50%-60%, which enhances relay and anti-attack capability.

Key words: IPv6; tunnel; Teredo; Miredo; relay

1 引言

随着 IPv6 技术的发展,IPv4 向 IPv6 的过渡逐步加快,共存现象也愈加明显.为使无 IPv6 接入的用户提前使用 IPv6,研究者提出了众多隧道技术.其中 Teredo 可穿透多级 NAT、不需 ISP 支持,对终端用户是一种方便快捷的 IPv6 接入方式,从用户层面推动 IPv6 的发展.

目前针对 Teredo 隧道的研究主要有以下几个方面:1.Teredo 实现及功能扩展;2.Teredo 流量及性能分析;3.Teredo 安全性研究.在 Teredo 实现及功能扩展方面,研究者在不同平台上实现了 Teredo 功能,并通过更改地址结构、连接过程、数据交换方式等方法进行扩展.其中比较知名的包括 Miredo、SYMTeredo[1]、Silkroad[2]等,以及文献[3]扩展了 Teredo 支持的 NAT 种类.Teredo 流量及性能分析主要包含两个方面,其一是监控网络中 Teredo 流量比例、上层应用等信息.其中文献[4,5]分别测试了 HTTP、FTP、DNS 和 BitTorrent 中 Teredo 地址的使用情况;其二是针对 Teredo 服务的相关性能进行研究,主要对 Teredo 延迟进行测试,如文献[6]等指出 Teredo 相对于 IPv6、6to4、ISATAP 等延迟较大.

Teredo 安全主要针对协议组件及交互过程的安全问题进行研究.其中文献[7]首先对 Teredo 中存在的 NAT 级别、源路由、地址欺骗等安全隐患进行了研究.之后分析了针对 Client 和 Relay 中 Peer 队列的 DoS 攻击、针对支持加密认证 Server 的 DoS 攻击、Relay 失效后 IPv6 Host 连接过程,以及不同情况下(比如攻击者知道 NAT

收稿日期:2013-06-17;修回日期:2013-11-24;责任编辑:李勇锋
基金项目:国家 973 重点基础研究发展计划(No.2012BAH37B01);国家自然科学基金(No.61202457);国家 863 高技术研究发展计划(No.2011AA010705,2012AA012506,2012AA012502)

对外 IP)攻击者定位 Teredo 地址概率等安全问题.文献[8]主要关注 Teredo 的 ACL 控制策略,为路由器、防火墙配置提供参考.文献[9]从部署角度列举了 Teredo 的诸多安全问题,包括防火墙检测规避、增加网络内部暴露概率等.文章同时对利用 Teredo 进行的针对 Server 和 Relay 的 DoS、中间人以及反射攻击方式进行了研究,部分内容与文献[7]有重叠.文献[10]对 Teredo 的安全性进行了升级,一是优化 Flag 字段,采用随机方式,减少 Teredo 地址被探测的概率;二是废弃 Cone 位,可降低针对 Peer 队列的 DoS 攻击概率.文献[11]提出了针对 NAT 设备和 Teredo Server 的两种不同方式的利用路由回路的攻击,并通过地址过滤基本解决上述问题.文献[12]提出了利用地址格式、DNS、内层包头检查等方式过滤 Teredo 流量的基本方法,是对文献[8]的补充,但文献也指出现有过滤方法存在一定漏洞,对特定情况无效.

由于 Relay 是数据包转发核心组件,且对其性能和安全性展开针对性研究的论文较少,因此本文展开针对性研究,并重点分析 Miredo Relay 中核心功能节点维护机制存在的问题.在不影响 Server 和 Client 的情况下,通过双层查找优化、状态集与时间链扩展、时间链更新算法优化提高 Relay 的响应速度和安全性.

2 Peer 维护机制性能与安全性分析

Teredo Relay 负责 IPv6 主机、NAT 设备和 Teredo 客户端间的通信.由于 Miredo 使用较多,提供在线服务,且采用查找树和时间链的方式有较好的可扩展性,因此本节以 Miredo Relay 为基础,分析其性能和安全问题.

2.1 Miredo Relay 中 Peer 维护机制

Miredo Relay 的 Peer 维护机制包括实体查找树和虚拟时间链两部分,包括包处理(包含 IPv6(插入)和 IPv4(更新)两个线程,因处理流程类似,均用包处理线程表示)和回收三个独立线程,采用锁机制并行,不能同时对查找树和时间链进行操作,基本框架如图 1 所示.

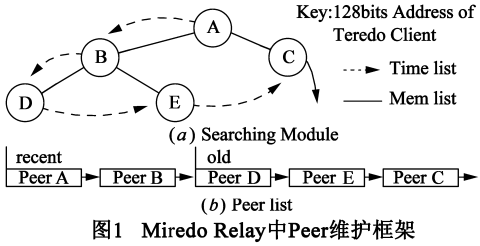


图1 Miredo Relay中Peer维护框架

包处理线程执行逻辑如下:(1)收到 IPv6 包,若目的为 TCA(Teredo Client Address)则在查找树中查找.存在则获取相应数据、不存在则插入新节点,挪动自身到时间链首部并更新 recent 指针,最后进行转发、发送 Bubble 或丢弃等操作;(2)收到封装在 IPv4 UDP 中的 IPv6 数据包,如果源 IP 是 TCA 地址,则在查找树中查找,存在则返回相应结构指针,并挪动到时间链头部,更新 recent 指针;否则丢弃(Relay 从 IPv6 网络端建立 Peer 结构).

回收线程每 30s 为一个周期(默认情况),周期结束时 old 指针右侧所有 Peer 将从查找树中释放并移出时间链,最后 old 指针指向 recent 位置,进入下一周期.

2.2 主要攻击方式

文献[7,9]分析了针对 Relay 中 Peer 队列的 DoS 攻击方式,通过不断向 Relay 发送目的地址为 TCA 的 IPv6 数据包,占满 Peer 队列,使其在超时前无法被正常请求使用.主要包括两种方式:(1)攻击者从 IPv6 端向 Relay 发送大量目的为 TCA 的包,占用大量 Peer 列表,但不维护 Server 端或 Client 端连接;(2)在条件 1 下,攻击者同时维护 Server 或 Client 端,保持连接.

为应对情况 1,必须增加 Peer 容量,使 Relay 可承受一定的带宽攻击.但这会引发两个问题:(1)扩大 Peer 容量会增加查找树的深度,增加查找延时;(2)导致回收线程每个周期结束时释放 Peer 的平均长度增加,可能引发包处理线程阻塞而丢包.因此,需要能提高查找性能并能减少回收线程时间的优化方法.针对第二种情况,需要提高认定 Teredo 连接可信的标准,提前判定连接无效的时间点,增加其维护虚假连接的难度.

3 Relay 性能及安全性优化

针对上一节的问题,本节提出了优化的 Relay 框架,如图 2 所示.

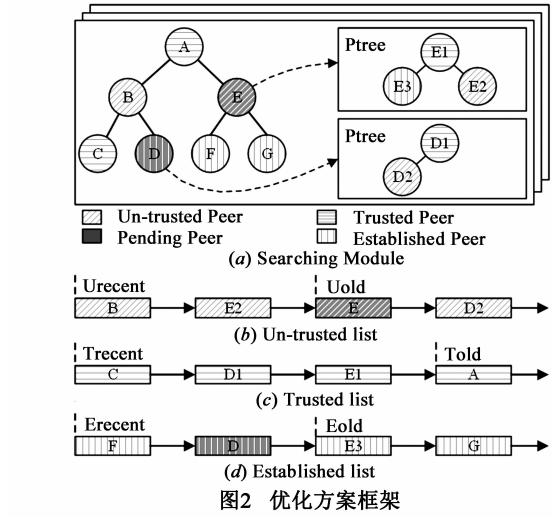


图2 优化方案框架

与原始情况相比,查找树变为多棵且从一层升为两层(每层的查找算法与原始一致),Peer 状态从一种增加为四种,分别是非信任(Un-trusted,简写 U)、信任(Trusted,T)、建立(Established,E)和待定(Pending,P).其

中处于 Pending 状态的第一层节点包含一棵第二层查找树(如图 2(a)中 D、E 节点),且第一层的 Pending 节点和第二层节点均可能处于 U、T 或 E 状态.双层查找树的所有节点状态可用二维变量表示:

$$\{(x,y)\mid x=\{\mathrm{U},\mathrm{T},\mathrm{E}\},y=\{0,1,2\}\},$$

其中"0"表示第一层非 Pending 节点,"1"表示第一层 Pending 节点,"2"表示第二层节点.图 2(b) – (d)分别代表了 U、T、E 三种状态节点形成的时间链.下述 3 节将从双层查找优化、状态集与时间链扩展、时间链更新算法三方面阐述该框架的运行逻辑.

3.1　基于地址结构的双层查找优化

图 2(a)所示为双层查找框架,按查找长度分割,一层查找树加速节点正常情况下的查找速度,二层检测并保证连接的安全性.同时将一层查找树通过 Hash 扩展为多棵,进一步增加查找性能.

3.1.1　一层查找树优化

一层查找树与原始架构相同,通过缩短关键字比较长度提高查找的性能.Miredo 使用 128 位 TCA 作为关键字,其中"2001:0"是 Teredo 标准前缀,用 SerIP 表示 Teredo 服务器地址,Flag 表示客户端 NAT 类型,ExPort、ExIP 表示最外层 NAT 对外端口和地址.

正常情况下 TCA 低 48 位即可区分不同 Peer,主要理由如下:(1)2001:0 为 Teredo 标识,不需比较;(2)SerIP 是 Teredo Server 的地址,当客户连接特定 Server 时,会生成唯一 TCA 地址,ExPort 和 ExIP 随之生成,因此能唯一对应一个 SerIP;(3)Flag 字段用于区分 NAT 类型,一般而言,站点的 NAT 类型是稳定的,ExPort 和 ExIP 可以代表 NAT 的种类,因此可省略.由此,ExPort 和 ExIP 在正常情况下可唯一表示一个 Teredo 客户,将一层查找树的比较长度降为 48 位,减少为原始的 37.5%,理论上可大量提升一层查找树的查询和更新速度.

3.1.2　二层节点优化

一层查找树仅由 TCA 低 48 位构成,忽略了 SerIP 和 Flag 字段的信息,在以下两种情况下会出现异常:(1)客户可能更换 SerIP,若对应的 ExPort 和 ExIP 更换,则会更新到一层查找树的其它节点,原节点超时释放,不产生影响,而若未变,则会产生冲突,必须更新原有连接;(2)攻击者可发送伪造 SerIP 和 Flag 的数据包,占用 ExIP 和 ExPort,使特定 NAT 内部客户无法与 Relay 建立正常连接.

在上述情况发生时需要动态建立二层查找树以保证连接的安全性,维护 SerIP 和 Flag 相异的 TCA 地址.二层查找树采用 SerIP 和 Flag 作为关键字,查找算法可与一层相同,一旦一级查找树节点存在不同的 SerIP 和 Flag 则动态生成二级查找树,并更新节点状态为 Pending.

3.2　状态集与时间链扩展

原始情况 Peer 保持两种状态:Un-trusted 或 Trusted,Relay 收到 IPv6 数据包时如果是 cone 则直接进入 Trusted 状态,restricted 则发送 bubble 后等待 Client 回包后进入 Trusted 状态.面对大流量 Teredo 数据包时,这种方式可能导致回收线程处理时间增加,包处理线程阻塞丢包.为避免类似情况发生,可扩展 Peer 状态集,通过更严格检查缩短恶意攻击产生的无效或伪造节点的生存时间,更新后状态集与时间链如图 3 所示.

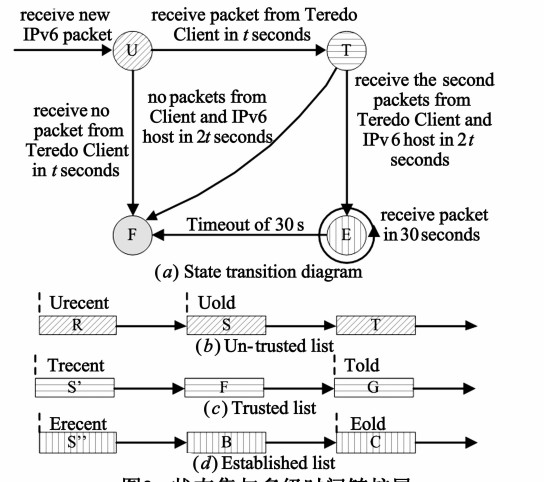


图3　状态集与多级时间链扩展

图 3 中,U 状态为 Peer 建立的初始状态,当 Relay 收到目的为 TCA 的 IPv6 数据包时,在双层查找树中查找,不存在则在特定查找树中(一层或二层)建立新节点并插入 Ulist(Un-trusted list)头部.正常情况下,U 状态节点会在 t 秒内收到客户端回复,进入 T 状态,并把自身移入 Tlist 首部(如 S′).T 状态节点在 $2t$ 秒内分别收到 IPv6 端和客户端的数据包则进入 E 状态并移入 Elist 首部(如 S″),否则释放.E 状态 30s 内未更新则释放.

相比于原始方式,增加了 U-T 和 T-E 两个状态迁移过程,其中 U-T 将 30s 的处理时间缩减为 t 秒、数量降为原始的 $3.3*t\%$,可有效应对 2.2 节中的第一种攻击情况,而 T-E 的转换增加了第二种攻击维持恶意连接的难度.

3.2.1　t 的选择

时间间隔 t 用于细化 Peer 状态的检测粒度,降低异常 Peer 的存在时间.根据文献[6]对 Teredo 的测试,相关结论如下:(1)成功连接的客户端获取测试网页的时延不超过 1.5s;(2)Teredo 建立连接的 CDF 曲线表明,4s 时 90% 的 Teredo 连接可以建立完毕,6s 超过 95%.超过 6s,连接失败率超过 85%,且建立时间越长,失败率越高.由此认为,t 为 6s 可维持较高的连接成功率,单周期释放的 Peer 数量降为原始的 20%.

3.2.2 Pending 状态管理

第一层节点收到不同的 SerIP 和 Flag 时进入 Pending 状态,表示为{(x,0)→(x,1)|x={U,T,E}}.新增节点形成该第一层节点的二层节点,初始状态为(U,2),并加入 Ulist 头部.原始第一层节点与每个二层节点独立的按照图 3 所示状态转换逻辑维护自身状态和进行时间链操作.比如图 2 中 D 节点实际包含了 3 个点:D、D1 和 D2,分别处于 E、T 和 U 状态.

对 Pending 及其二层节点的管理包括插入、更新和释放三个主要流程,其中插入和更新与原始方式基本一致,只需增加二层节点的处理和时间链更新等操作.释放过程相对复杂,包括 Pending 节点自身和内部二层节点释放两种情况.如图 2 中 D 节点,按时间关系,D2、D1 可能会先于 D 本身释放,则 D 的二层查找树会首先释放;但按照 E 节点的时间链位置,E 先于 E2 被释放,之后 E2 无法通过一层查找树定位,因此不能直接释放 E 节点.

为解决这一问题,在 Pending 节点上增加僵死状态(x,3),表示该节点的 SerIP 和 Flag 已无效,但存在相异的二层节点.同时在一层节点上增加二层节点计数,用于记录其包含的二层节点数目.僵死状态节点若再次收到 IPv6 数据包则重新插入 Ulist,表示一个新连接.

3.3 时间链更新算法优化

Miredo 时间链更新算法如图 4.假设系统处于(a)状态,收到 D 包时移动 D 到队首并更新 recent,进入(b)状态.此时若收到 A 包则进入状态(c).若 D、A 交替出现,则状态将在(b)(c)间不断转换,浪费大量链表操作.

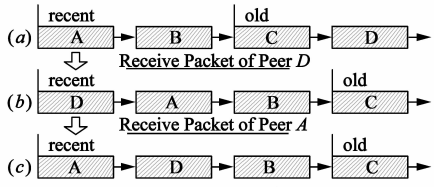


图4 时间链更新机制

事实上,只要节点在 old 指针左侧,本周期结束时就不会被释放.为此可增加时间戳变量进行优化,假设 k 表示某一周期的时间戳,则该周期内 old 指针右侧每个节点时间戳为 k,左侧节点为 $k+1$.更新时,若该节点时间戳为 k 则表明其位于 old 右侧,挪动到链首,时间戳加 1;若该节点时间戳为 $k+1$ 则表明位于 old 左侧,不需要进一步操作.本周期结束时,系统时间戳加 1,与 old 左侧节点一致,进入下一周期.

4 测试

本节对优化前后的 Relay 性能进行了理论分析和实验测试.理论上,每个 Peer 需增加 16B 内存空间,空间占用随 Peer 容量线性增加.时间上,设 $L1$、$L2$ 分别代表两层节点数目,K 为一层查找树数量,则原始情况下插入、更新和释放时间复杂度均为 $\log(L1+L2)$,修改后 $L1$ 节点时间复杂度为 $\log(L1/K)$,$L2$ 节点插入和更新时间复杂度为 $\log(L1/K)+\log(L2_i)$,释放时间复杂度为 $\log(L2_i)$.修改后,$L1$ 节点插入、更新和释放,以及 $L2$ 节点释放流程时间复杂度均有所下降(一般认为 $L2_i \ll L1$),$L2$ 节点插入和更新则相对复杂.下面通过实验评估优化前后的性能.

实验采用统计特定数据包处理时间的方式进行,通过处理时间的变化对优化前后的性能进行对比.实验分为三个部分:插入测试、更新测试和回收测试.

插入测试中,由于 $L1$ 和 $L2$ 节点插入时间复杂度与原始情况差异较大,因此分三种情况:(1)全部节点均为 $L1$ 层;(2)每个 $L1$ 节点包含一个 $L2$ 节点;(3)$L1$、$L2$ 节点各一半,其中最后一个 $L1$ 节点包含了所有的 $L2$ 节点.测试地址数量 0.1M – 1.8M.测试结果如图 5 所示,其中"O/N"分别代表原始和优化后,"1/2/3"表示三种情况.

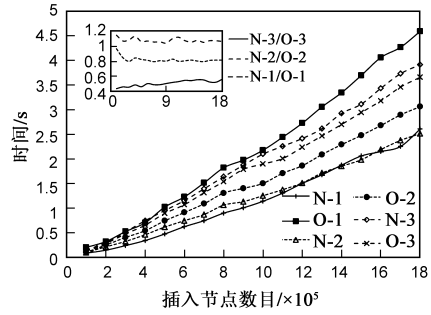


图5 插入性能测试

图 5 中,情况 1 的优化效果明显,平均用时仅为原始的 51.3%(N – 1/O – 1),性能提高一倍.情况 2 的优化效果有所降低,平均比值为 82.7%,表明插入二层节点会加大系统负载.情况 3 的平均比值为 108.3%,主要由于实验中没有限制二层节点数目,导致二层查找树较大、$\log(L2_i)$ 增加所致,这种现象可认定为一种攻击行为.三种情况下优化前后的时间比例随数据包数目增加保持稳定,未出现较大波动.

更新过程包括查找和时间链更新两部分,查找过程相对于插入省去了创建步骤,在上述三种情况下应与插入过程有相似的性能趋势,因此更新过程主要测试 $L1$ 层更新的性能.时间链更新包括 U-T、T-E 和 E-E(E 状态内更新)的移出和移入操作.测试数据采用每 10 个数据包为一组进行,每组数据包序列为 6|4|6|4|6|6|6|6|6|6,可使特定 $L1$ 节点从 U 迁移到 E 状态并在 E 状态内更新,不同组数据包交叉排序.实验同时增加 $L1$ 层在 U 状

态的纯查询测试进行对比,每组序列为 6|6|6|6|6|6|6|6|6|6,不同组数据包交叉排序.测试结果如图 6 所示,其中"E"表示第一组测试,"U"为对比测试.

图 6 中,纵向比较,优化后的算法在两种情况中用时均有较大减少,平均仅为原始的 42.7% 和 44.5%,且波动不大.横向比较,N-E 曲线平均仅比 N-U 曲线增加 1.8%,表明从 U 到 E 的时间链操作在优化后增加的计算量很小.

回收测试主要对优化前后回收线程释放一定数量节点的用时进行对比.与插入测试相同,分三种情况,用"1/2/3"表示.回收流程包括从树中删除节点和释放内存两部分,其中后者与插入和更新线程不冲突,可并行进行,需要测试的时间只为将节点从树中删除的时间,因此所有节点在 Ulist 时间链中即可.实验结果如图 7 所示.

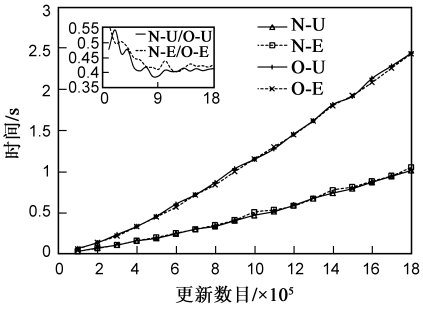


图6　更新性能测试

图 7 中,N-2 曲线用时最少,主要因为 *L*2 释放仅从 *L*1 节点子树释放即可,而子树只包含自身,因此最快.N-1 曲线居中.N-3 较慢,主要由于此时二层子树节点较多.纵向对比,三种情况下优化后平均用时仅为原始的 53.2%,40.1%,86.4%,性能均有所提高.理论上,优化后,每次回收流程用时将有较大缩短,若按 $t = 6$s 计算,回收线程每周期用时降为原始的 8%-17.2%,极大的降低了回收时间过长导致包处理线程丢包的概率.

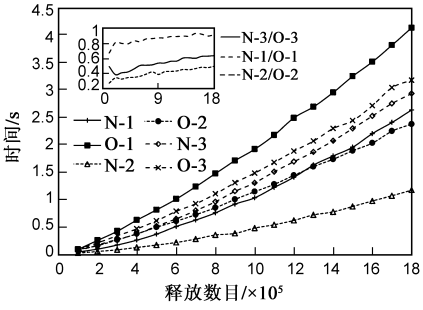


图7　释放性能测试

5　结论

本文对 Teredo Relay 进行研究,提出的双层查找优化、状态集扩展、时间链扩展和时间链更新优化算法在不影响 Server 和 Client 的基础上有效的提高了 Relay 的中继和抗攻击能力.实验表明,正常情况下优化后的包处理和回收流程用时仅为原始的 50% 左右,极端条件下与原始相近.

本文在以下方面仍需进一步研究:(1)Teredo 扩展功能,如 NAT 支持类型的增加等已在部分 RFC 中提出,但 Miredo 及其它开源软件中未实现,相关的安全问题需进一步研究;(2)提出的优化方法与查找树本身算法无关,因此可替换其它查找树算法进一步优化.

参考文献

[1] Huang Shiang-Ming, Wu Quincy, Lin Yi-Bing. Enhancing Teredo IPv6 tunneling to traverse the symmetric NAT[J]. Communication Letters, 2006, 10(5): 408 – 410.

[2] 吴贤国,刘敏,李忠诚.面向 NAT 用户的 IPv6 隧道技术研究[J].计算机学报,2007,30(1):1 – 9.
Wu Xian-Guo, Liu Min, Li Zhong-Cheng. Research on the IPv6 tunneling technology designed for network address translator users[J]. Chinese Journal of Computers, 2007, 30(1): 1 – 9. (in Chinese)

[3] Thaler Dave. RFC6081 (Teredo extensions)[S]. IETF RFC, 2011.

[4] Malone David. Observations of IPv6 addresses[A]. Proceedings of the 9th Passive and Active Measurement Conference (PAM)[C]. Berlin: Springer-Verlag, 2008. 21 – 30.

[5] Defeche Martin, Vyncke Eric. Measuring IPv6 traffic in BitTorrent networks[OL]. http://tools.ietf.org/html/draft-defeche-ipv6-traffic-in-p2p-networks-00, 2012.

[6] Zander Sebastian, Andrew LH Lachlan, Armitage Grenville. Investigating the IPv6 Teredo tunneling capability and performance of internet clients [J]. ACM SIGCOMM Computer Communication Review, 2012, 42(5): 13 – 20.

[7] Hoagland James. The Teredo protocol: tunneling past network security and other security implications[R]. Cupertino: Symantec, 2007.

[8] Hogg Scott, Vyncke Eric. IPv6 security[M]. Indianapolis: Pearson Education, Cisco Press, 2008.

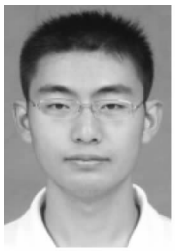
[9] Frankel Shelia, Graveman Richard, Pearce John, Rooks Mark. Guidelines for the secure deployment of IPv6[R]. Gaithersburg: National Institute of Standards and Technology, 2010. 80 – 119.

[10] Thaler Dave, Krishnan Suresh, Hoagland James. RFC5991 (Teredo security updates)[S]. IETF RFC, 2010.

[11] P Shanmugaraja, D Balamurugan, S Chandrasekar. An approach to secure Teredo tunneling technology[J]. International Journal of Engineering Research & Technology (IJERT), 2013, 2(3): 1 – 6.

[12] F Gont, W Liu. Security implications of IPv6 on IPv4 networks [OL]. http://tools. ietf. org/html/draft-ietf-opsec-ipv6-implications-on-ipv4-nets-05, 2013.

作者简介



崔 宇(通信作者) 男,1985 年生,黑龙江哈尔滨人,哈尔滨工业大学网络与信息安全专业博士研究生.主要研究方向:IPv6、网络与信息安全.

E-mail: cuiyudaniu@163.com

张宏莉 女,1973 年生,吉林榆树人,哈尔滨工业大学计算机学院副院长、教授、博士生导师.主要研究方向:计算机网络与信息安全、网络测量、并行处理.