

BGP MPLS VPN 安全性能分析与改进

李海华^{1,2}

(1 华北水利水电学院, 河南 郑州 450011; 2 河南工业贸易职业学院, 河南 郑州 450012)

摘 要: 在分析 BGP/MPLS VPN 安全隐患的基础上, 针对 BGP/MPLS VPN 域中可能出现的核心攻击, 提出了改进其安全性能的思路, 并给出了一个实施方案, 使得只有得到认证的核心路由器才能加入 BGP/MPLS VPN, 提高了网络的安全性能。

关键词: 多协议标签交换虚拟专用网; 密钥分配模型; 数据加密算法; 加密; 解密

中图分类号: TN911.22

文献标识码: A

文章编号: 1000-7180(2011)11-0141-07

Analysis and Improvement of Security for BGP/MPLS VPN

LI Hai-hua^{1,2}

(1 North China University of Water Resources and Electric Power, Zhengzhou 450011, China;

2 Henan Industry and Trade Vocational College, Zhengzhou 450012, China)

Abstract: On the base of analysing about the poential safety concerns in BGP/MPLS VPN region, and contraposing the potential core attalcks in the BGP/MPLS VPN, we state a thought line to improve its secure ability and draft an embodiment, so it can only permit the Core Router which get the authorization to enter the BGP/MPLS VPN, and improve the safety performance of the network.

Key words: BGP/MPLS VPN; key distribution model; DES; encryption; decryption

1 BGP/MPLS VPN 网络结构

服务商路由器 P(Provider Router)、服务商边缘路由器 PE(Provider Edge Routers)、用户边缘路由器 CE(Customer Edge Routers) [1] 构成 BGP MPLS VPN 网络框架, 它们在网络中扮演着不同的角色. 如图 1 所示.

① 服务商路由器(P). 图 1 中 P1 到 P4 是服务商路由器, PE 同它对端的 PE 构成的隧道, 隧道路径的过程中, 所涉及的 P 路由器构成标签交换路径 LSP(Label Switching Path), VPN 数据报通过交换外层标签在 P 构成的隧道中穿行, P 路由器不跟 CE 直接连接, 因此, P 路由器不要求维护每一个用户站点的特定的 VPN 的路由信息, 只要求维护 PE 路由器的路由信息.

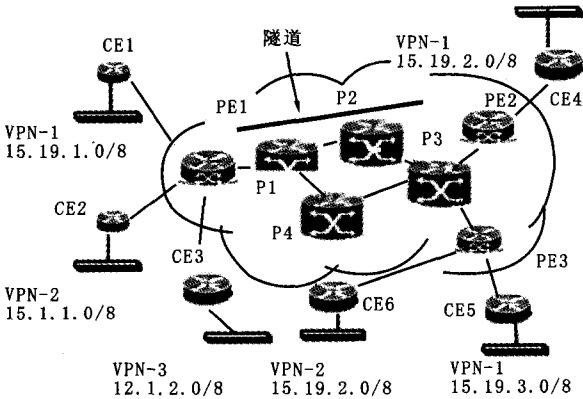


图 1 BGP/MPLS VPN 网络结构

② 服务商边缘路由器(PE). 服务商边缘路由器与用户边缘路由器通过 RIP、OSPF、BGP 等路由协议交换路由信息, PE 路由器保存与它直接相连的 CE 路由器的 VPN 路由信息, 其它路由器的

VPN 路由信息则被过滤,每个用户通过与它相连的 PE 路由器中的 VPN 转发表中的转发表条目 VRF (VPN Routing Forwarding)给 VPN 数据报打上内层标签,从而建立 VPN 隧道,PE1、P1、P4、P3、PE2 构成一条隧道,图 1 中 PE1 到 PE3 是服务商边缘路由器。

③ 用户边缘路由器(CE). 图 1 中 CE1 到 CE5 是用户边缘路由器. 用户的网络服务是由 CE 路由器通过与主机或二层交换机的直接连接来提供的,CE 路由器同与其直接相连的 PE 路由器之间建立邻居关系,建立邻居后,PE 路由器通告本地站点的 VPN 路由到远端的 PE 路由器并从远端的 PE 路由器学习 VPN 路由用来构建 VPN 路由转发表 VRF. 用户二层交换机或直接与 PE 路由器建立链接是建立在 PE 路由器的端口上,一个 VRF 可以绑定一个端口,也可以绑定多个端口。

2 BGP/MPLS VPN 安全性能分析与改进思路

2.1 安全性能分析

在 BGP/MPLS VPN 域中,首先,攻击可能发生在网络外部,也就是用户边缘路由器与服务商边缘路由器之间,但是,如果正确配置的 CE 和 PE,它们之间只能通过 IP 数据报流量,标签数据报则不能通过,保证 IP 数据报的安全转递;其次,攻击也可能发生在网络内部,也就是服务商路由器之间,在 PE 路由器上,路由区分符 RD(Route-Distinguisher)与 IP 数据报的 IP 地址构成 VPN-IPv4 地址,以解决不同 VPN 中地址重叠问题,每一个 VPN 有一个唯一的路由目标 RT(Route Target),它可以保证 VPN 路由信息准确导入相应的 VRF 中,通过 VRF,使每一个 VPN 有一个独立的 VPN 隧道,一个 PE 路由器上有多个 VRF,从而为每个 VPN 提供一个独立的隧道,保证 VPN 路由信息的隔离,但是,P 路由器之间是通过交换外层标签来转递 VPN 数据报,这种数据报仍有可能被截获,篡改信息后发向对端的 PE 路由器,使 VPN 遭到攻击. 如果在服务商路由器之间增加一个认证机制,阻止其它路由器冒充服务商路由器进入 BGP/MPLS VPN 域,就能进一步保证 P 路由器之间 VPN 数据报的安全,从而避免潜在入侵者进入系统^[2]。

2.2 改进思路

在图 1 中,路由器 P2 要进入 BGP/MPLS VPN 网络,就必须用标签路由信息同 P3 之间建立标签

交换通道 LSP,标签路由信息是 P2、P3 建立会话实体后进行分发的,P3、P2 通过标签分发协议 LDP (Label Distribution Protocol)中的 Hello 消息^[3]相互发现对方,建立会话实体,实体之间交换“通知消息”、“Hello 消息”、“初始化消息”、“会话保持消息”、“地址消息”、“标记映射消息”、“标记请求消息”等,从而建立标签路由信息. 在这里,对 P3 路由器发布的消息进行加密,那么,只有能对加密过的消息进行解密的路由器,也就是必须得到已在 MPLS 域中的 P3 路由器的认证的标签交换路由器 LSR (Label Switching Router)才能与 P3 建立邻居关系,假若 P2 路由器要入侵 VPN 网络,因为它不知道密码,也没有加解密机制,不能和 P3 相互认证,P2 与 P3 之间不能建立邻居关系,更不能建立标签交换路径 LSP (Label Switching Path),因而,P2 不能进入 BGP/MPLS VPN 网络. 问题是,如果对服务商路由器之间的所有消息都要加密解密,网络的效率就会被大大降低,因而,这里只对足多消息中的关键消息,也就是 Hello 消息进行加密,就能在不降低网络转发效率的条件下提高网络的安全性,因为 Hello 消息不能被破解,P3 就无法与 P2 建立邻居关系,就不能建立会话关系,之后消息的交换就不可能发生。

3 实施方案

一个密钥分配模块用来分发密码,一个加密、脱密模块用来对 LDP 协议中的 Hello 消息进行加密、脱密,进而提高 VPN 数据转发的安全性。

3.1 密钥分配模型

密钥分配模型如图 2 所示,工作原理如下:^[4]

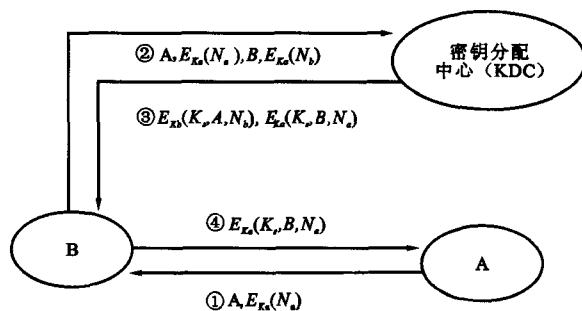


图 2 密钥分配模型

(1) A 首先告诉 B 自己的身份,再加上用自己与 KDC 共享的密钥加密的一个时间延迟 N_a ,说明自己是谁,要与 B 进行会话;

(2) B 收到 A 发来的消息后,知道 A 要与自己会话,但不知道与 A 进行会话的会话密钥,于是便向 KDC 请求:把 A 的身份与时间延迟 N_a ,还有自

己的身份与用自己与 KDC 共享的密钥加密的一个时间延迟 Nb, 都告诉了 KDC, 让 KDC 知道自己要会话的对方是 A;

(3) KDC 收到 B 的请求后, 知道 B 要与 A 进行会话, 要自己给出 B 与 A 进行会话的会话密钥, 于是 KDC 就作出相应的回答: 把会话密钥 K_s , A 的身份, 时间延迟 Nb 用自己与 B 的共享密钥 K_b 进行加密, 再用自己与 A 的共享密钥 K_a 加密会话密钥 K_s , B 的身份和时间延迟 Na, 然后再把它们发给 B.

(4) B 收到 KDC 发回的消息后, 经一系列运算, 发现 Nb, 知道是自己关于 A 的会话密钥请求的一个应答, 并向 A 转发它们的会话密钥 K_s , 只有 A 才能解开.

3.2 LDP 加密/脱密模型

图 3 中, LDP 模块安装后, 密码交换模块通过调用标签分发协议模块中的进程机制执行密码交换模块进程. 从而, 使 BGP/MPLS VPN 网络中两个 P 路由器之间进行密钥交换. 下面以图 1 中 P1 和 P4 之间建立会话实体的过程进行说明, 在会话实体建立之前, P1 与 P4 之间要交换密钥, 以便对对方的 Hello 消息进行解密. 接着, P1 和 P4 之间要互发消息, 消息采用类型—长度—值 TLV (Type—Length—Value) 编码体系进行编码. P1 路由编码函数对 Hello 消息中 TLV 进行编码后, 加密函数接着对它进行加密, 然后由发送函数将它发送到 P4, P4 的脱密函数用它从 P1 得到的密钥对加密过的 Hello 消息进行解密. 它的解码函数将 Hello 消息中的值加以恢复, P4 理解了 P1 的参数. 同理, P1 也通过同样的方式理解 P4 的参数, 这样, P1、P4 之间接着就可以建立对话实体, 为双方交换消息打开基础.

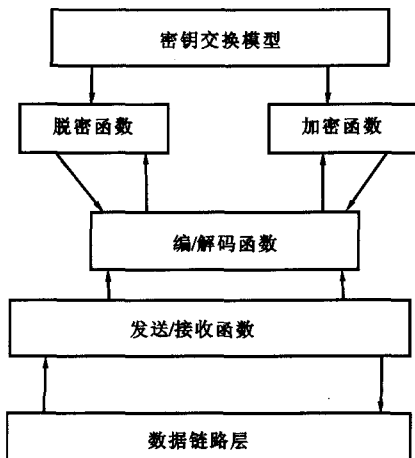


图 3 LDP 加密模型

3.3 加密算法

加、解密函数采用 DES 算法^[5], 加密函数将消息按 8 比特分成八组进行加密, 解密函数解密时将八个加密过的分组解密后组装, 然后通过解码函数解码, 恢复 Hello 消息原貌.

图 4 中, 密码生成器在左边, 提供密码; 加密解密结构在右边, 用来对消息的加密与解密, 输入明文 (m), 它会用密码处理后以密文 (C) 的形式输出, 若输入密文 (C), 它会恢复原来的明文 (m).

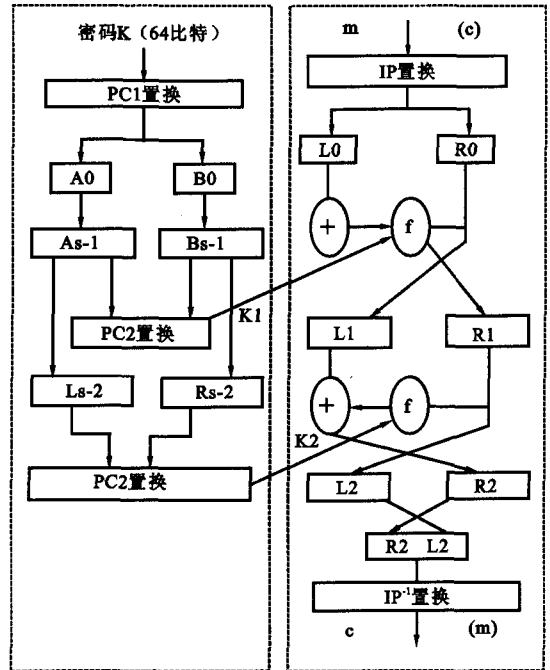


图 4 DES 算法结构

64 比特的密码 K 经 PC1 置换 ($PC1[64] = \{57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36, 63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4\}$) 后分成两部分, 分别放入左右寄存器 A0、B0 中, As-1、Bs-1 分别将 A0、B0 中的数据循环左移 1 位后经 PC2 置换 ($PC2[48] = \{14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10, 23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2, 41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48, 44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32\}$) 后生成 48 比特的密码 K1; As-2、Bs-2 分别将 As-1、Bs-1 中的数据循环左移 2 位后经 PC2 置换生成 48 比特的密码 K2.

如图 5 中 f 函数实现如下功能: 通过 $E(E[48] = \{32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10,$

11, 12, 13, 12, 13, 14, 15, 16, 17, 16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25, 24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1 }扩展,将寄存器 R 中 32 比特的数据变换为 48 比特数据,与 48 比特密钥进行求和运算,得到的 48 比特数据按 6 比一个分组,共分 8 个分组分别送入 S1 到 S8 八个送 S 盒中,每个 S 盒将输入的 6 比特数据变为 4 比特数据,八个 S 盒的输出数据合到一起成为 32 比特数据,经 P 置换($P[32]=\{16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,22,11,4,25\}$),输出的 32 位比特串就是函数 f 的结果 $f(R,K)$.

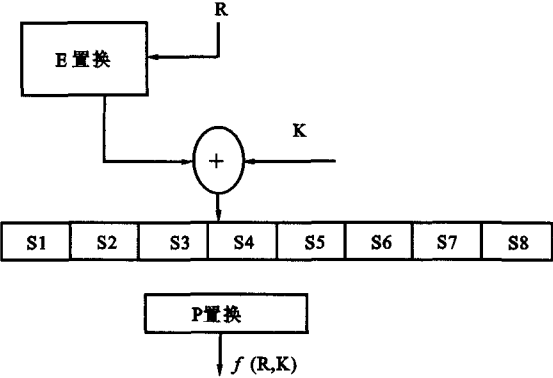


图 5 f 函数的实现

64 比特的消息 m 经 IP 置换($IP[64]=\{58, 50,42,34,26,18,10,2,60,52,44,36,28,20,12,4, 62,54,46,38,30,22,14,6,64,56,48,40,32,24,16, 8,57,49,41,33,25,17,9,1,59,51,43,35,27,19, 11,3,61,53,45,37,29,21,13,5,63,55,47,39,31, 23,15,7\}$),分为两组,每组 32 比特,分别放在左右寄存器 L0 及 R0 中,R0 中的数据一路通过 f 函数后与 L0 中的数据进行与运算后放在寄存器 R1 中,另一路送寄存器 L1 中,寄存器 R2 中数据,一路通过 f 函数后与 L1 中的数据进行与运算后放在寄存器 R2 中,另一路送寄存器 L2, L2 与 R2 中数据交换后经 IP-1 置换($ip-1[64]=\{40,8,48,16,56,24,64, 32,39,7,47,15,55,23,63,31,38,6,46,14,54,22, 62,30,37,5,45,13,53,21,61,29,36,4,44,12,52, 20,60,28,35,3,43,11,51,19,59,27,34,2,42,10, 50,18,58,26,33,1,41,9, 49,17,57,25\}$)后生成密文 C.

3.4 Hello 消息的结构

Hello 消息的结构如图 6 所示. 它由 IP 头、UDP 头及 LDP PDU 组成,其中 LDP PDU 由 IP 头及 HELLO 消息构成.

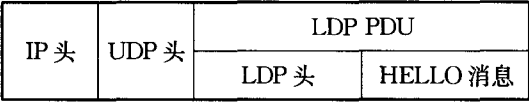


图 6 Hello 消息的结构

版本号、PDU 长度及 LDP 标识符组成 LDP 头,填充版本号字段用了 2 个字节无符号整型数据,指示 LDP 版本号;PDU 长度是一以字节为单位来表示的,用 2 个字节整型数据来填充 PDU 长度字段,PDU 默认长度为 4096 字节,当然,这个长度两个会话实体之间可以协商;用 6 个字节来表示 LDP 标识符,4 个字节用来表示标签交换路由器 LSR 的 IP 地址,2 个字节用来表示 LSR 内的标记空间. 如图 7 所示.

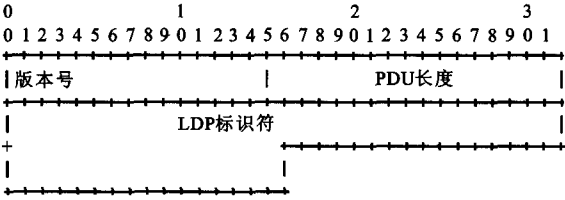


图 7 Hello 消息的结构

在 Hello 消息的编码中,32 比特的数值用于消息标识,如图 8 所示.



图 8 Hello 消息的编码

通用 Hello 参数 TLV 编码如图 9 所示.

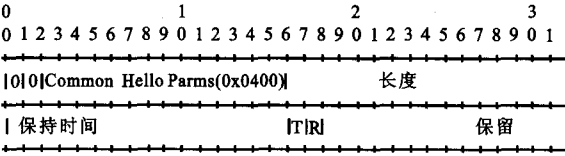


图 9 通用 Hello 参数 TLV 编码

前 32 比特的含义:

```
struct ldp_tlv_header
{
    union
    {
        struct ldp_tlv_header_bits
        {
            /* u_bit: 表示未知 TLV 比特, f_bit: 表示转发未知 TLV 比特, type: 表示类型 */
            BITFIELDS3 (u_int16_t u_bit : 1, u_int16_t f_bit : 1, u_int16_t type : 14);
```

```

}bits;
u_int16_t s_packet;
}flags;
u_int16_t length; /* 长度字段用 16 比特表示 */
};
后续 32 个比特的含义:
struct ldp_tlv_common_hello
{
    struct ldp_tlv_header tlv_h;
    u_int16_t holdtime; /* 用 16 比特表示保持时间 */
    union
    {
        struct ldp_common_hello_bits
        {
            /* t_bit: T 字段, r_bit: R 字段 reserve: 保留字段 */
            BITFIELDS3 (u_int16_t t_bit : 1,
                u_int16_t r_bit : 1,
                u_int16_t reserve : 14);
        }bits;
        u_int16_t s_packet;
    }flags;
};

```

为使 Hello 消息在两个 P 路由器之间可靠传递, 在一个 P 路由器向另一个路由器发送第二个 Hello 消息前, 对前一次发送的 Hello 消息要保留一定的时间, 两个 P 路由器之间可以提出自己的保留时间来协商这个 Hello 保持时间, 时间以秒为单位来表示。Hello 保持时间的默认值为 0, 链路 Hello 消息保持时间为 15 秒, 目标 Hello 消息保持时间为 45 秒^[6]。

T 是 Hello 消息类型指示, 目标 Hello 消息用 1 表示, 链路 Hello 消息用 0 表示; R 表示是否请求发送 Hello 消息, 请求发送时用 1 表示, 没有请求时用 0 表示; 发送 Hello 消息时, 保留字段必须被设为 0, 接收时必须被忽略。

3.5 加密脱密的实现

编码函数首先调用加密函数先从原始 Hello 消息中取出通用 Hello 参数 TLV 的 64 比特作为明文送入加解密模块, 利用密钥交换模块生成的密钥进行加密生成密文, 再由编码函数组装成新的 Hello 消息, 由发送函数发送到邻居 P; 邻居 P 接收函数接收新 Hello 消息, 解码函数调用脱密函数, 从中取出新通用 Hello 参数 TLV 的 64 比特作为密文送入加解密模块, 利用密钥交换模块生成的密钥进行解密生成明文, 将明文回解码函数解码^[7]。

加密/解密程序代码如下:

```

/* 指定用 ElemType 代替 char 类型 */
typedef char ElemType;
/* 初始置换表 IP 数组 */
int IP_Table[64]
/* 逆初始置换表 IP-1 数组 */
int IP_1_Table[64]
/* 扩充置换表 E 数组 */
int E_Table[48]
/* 置换函数 */
int P_Table[32]
/* S 盒数组 (S1, S2, S3, S4S, S6, S7, S8) */
int S[8][4][16]
/* 置换选择数组 1 */
int PC_1[56]
/* 置换选择数组 2 */
int PC_2[48]
/* 对左移次数的规定数组 */
int MOVE_TIMES[16]
/* 字节转换成二进制 */
int ByteToBit(ElemType ch, ElemType bit[8])
/* 二进制转换成字节 */
int BitToByte(ElemType bit[8], ElemType * ch)
/* 将长度为 8 的字符串转为二进制位串 */
int Char8ToBit64(ElemType ch[8], ElemType bit[64])
/* 将二进制位串转为长度为 8 的字符串 */
int Bit64ToChar8(ElemType bit[64], ElemType ch[8])
/* 在生成子密钥函数 int DES_MakeSubKeys(ElemType key[64], ElemType subKeys[16][48])
ElemType temp[56] 中, 通过 PC1 置换函数 int DES_PC1_Transform(ElemType key[64], ElemType tempbts[56]) 将 64 位密码 56 位密码, 接下来进行 16 轮迭代, 产生 16 个子密钥; 在每一轮迭代中, 将 56 位密码通过循环左移函数 DES_ROL(temp, MOVE_TIMES[cnt]) 右移一位, 通过 PC2 置换函数 DES_PC2_Transform(temp, subKeys[cnt]) 产生 48 位子密钥。 */
/* IP 置换函数 int DES_IP_Transform(ElemType data[64]) 将数据按数组 IP_Table[64] 置换 */
/* IP 逆置换函数 int DES_IP_1_Transform(ElemType data[64]) 将数据按数组 IP_1_Table[64] 置换 */
/* 扩展置换函数 int DES_E_Transform(ElemType data[48]) 按数组 int E_Table[48] 进行扩展 */
P 置换函数 int DES_P_Transform(ElemType data[32]) 按数组 P_Table[32] 进行转换 */
/* 异或函数 int DES_XOR(ElemType R[48], ElemType L[48], int count) 对两个 48 位二进制数进行异或运算 */
/* 交换函数 int DES_Swap(ElemType left[32], ElemType right[32]) 完成左右 32bit 数据交换 */

```

```

/* S 盒置换函数 int DES_SBOX(ElemType data[48])
进行 8 次循环,每次循环将 data[48]中的 6bit 数据按数组 S
[8][4][16]中的分量转转换成 4bit 2 进制数据 */
/* 加密单个分组 */
int DES_EncryptBlock(ElemType plainBlock[8], ElemType
subKeys[16][48], ElemType cipherBlock[8]){
ElemType plainBits[64];
ElemType copyRight[48];
int cnt;
Char8ToBit64(plainBlock, plainBits);
/* 初始置换(IP 置换) */
DES_IP_Transform(plainBits);
/* 16 轮迭代 */
for(cnt = 0; cnt < 16; cnt++){
memcpy(copyRight, plainBits+32, 32);
/* 将右半部分进行扩展置换,从 32 位扩展到 48 位 */
DES_E_Transform(copyRight);
/* 将右半部分与子密钥进行异或操作 */
DES_XOR(copyRight, subKeys[cnt], 48);
/* 异或结果进入 S 盒,输出 32 位结果 */
DES_SBOX(copyRight);
/* P 置换 */
DES_P_Transform(copyRight);
/* 将明文左半部分与右半部分进行异或 */
DES_XOR(plainBits, copyRight, 32);
if(cnt != 15){
/* 最终完成左右部的交换 */
DES_Swap(plainBits, plainBits+32);
}
}
/* 逆初始置换(IP-1 置换) */
DES_IP_1_Transform(plainBits);
Bit64ToChar8(plainBits, cipherBlock);
return 0;
}

/* 解密单个分组 */
int DES_DecryptBlock(ElemType cipherBlock[8], Elem-
Type subKeys[16][48], ElemType plainBlock[8]){
ElemType cipherBits[64];
ElemType copyRight[48];
int cnt;
Char8ToBit64(cipherBlock, cipherBits);
/* 初始置换(IP 置换) */
DES_IP_Transform(cipherBits);
/* 16 轮迭代 */
for(cnt = 15; cnt >= 0; cnt--){
memcpy(copyRight, cipherBits+32, 32);
DES_E_Transform(copyRight);
/* 将右半部分与子密钥进行异或操作 */
DES_XOR(copyRight, subKeys[cnt], 48);
/* 异或结果进入 S 盒,输出 32 位结果 */
DES_SBOX(copyRight);
/* P 置换 */
DES_P_Transform(copyRight);
/* 将明文左半部分与右半部分进行异或 */
DES_XOR(cipherBits, copyRight, 32);
if(cnt != 0){
/* 最终完成左右部的交换 */
DES_Swap(cipherBits, cipherBits+32);
}
}
/* 逆初始置换(IP-1 置换) */
DES_IP_1_Transform(cipherBits);
Bit64ToChar8(cipherBits, plainBlock);
return 0;
}

/* 在编码函数中调用加密函数如下 */
ldp_message_hello_encode(u_char ** pnt, u_int16_t *
size, struct ldp_message_hello * hello)
{
int ret;
u_char * spnt = * pnt;
/* 调用加密函数 DES_EncryptBlock */
memcpy(plainBlock[8], ldp_message_hello, 8);
DES_EncryptBlock(plainBlock[8], subKeys[16][48], ci-
pherBlock[8]);
memcpy(ldp_message_hello, cipherBlock[8], 8)
ret = ldp_message_header_encode(pnt, size, &hello->
mes_h);
if (ret < 0)
return ret;
ret = ldp_tlv_common_hello_encode(pnt, size, &hello->
common);
if (ret < 0)
return ret;
}

/* 在解码函数中调用解密函数如下 */
ldp_message_hello_decode(u_char ** pnt, u_int16_t *
size,
struct ldp_message_hello * hello)
{
/* 调用解密函数 DES_DecryptBlock */
memcpy(cipherBlock[8], ldp_message_hello, 8);
DES_DecryptBlock(cipherBlock[8], subKeys[16][48],
plainBlock[8])

```

```
memcpy(ldp_message_hello, plainBlock[8], 8);
pal_mem_set(hello, 0, sizeof(struct ldp_message_hello));
/* 对消息头部分解码 */
ret = ldp_message_header_decode(pnt, size, &hello->
mes_h);
ret = ldp_tlv_common_hello_decode(pnt, size, &tlv_h,
&hello->common);
if (ret < 0)
return ret;
}
```

4 仿真实验

随机选择 50 个网络拓扑结构, 分别在未采安全策略及采用安全策略条件下用 NS-2 对安全性能进行仿真实验, 并进行归一化处理, 如图 10 所示. 分析可以看出, 采用安全策略后, MPLS 网络的安全性能提高了 11%~19% 之间.

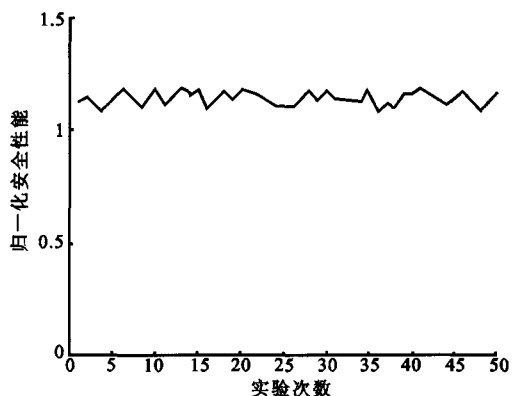


图 10 实施安全策略前后安全性能比较

5 结束语

文中指出了 BGP MPLS VPN 潜在的核心攻击, 给出了 LDP 加密/脱密模型. 采用 DES 算法对路由器之间会话所用的消息数据进行加密/解密, 只有被认证的路由器才能加入 BGP/MPLS VPN, 使得它的安全性能得到了提高.

参考文献:

- [1] IETF RFC-3031. Multiprotocol label switching architecture[S]. 2001.
- [2] RFC 3036. LDP Specification[S]. 2001.
- [3] IETF RFC-3032. MPLS LabelStack Encoding[S]. 2001.
- [4] 阮立志, 许凌云. 一种基于网格计算环境的安全实现研究[J]. 微电子学与计算机, 2009; 265-268.
- [5] Schneier. 应用密码学: 协议、算法、与 C 源程序[M]. 北京: 机械工业出版社, 2000.
- [6] 吴伟. 下一代 IP 网络技术保障—多协议标签交换技术[M]. 北京: 清华大学出版社, 2002.
- [7] 归奕红, 杜瑞颖. 基于 FGC 的无线传感器网络安全结构研究[J]. 微电子学与计算机, 2009(06): 223-226.

作者简介:

李海华 男, (1965—), 副教授. 研究方向为网络协议与信息安全.

(上接 140 页)

- [6] Kalla R. IBM Power5 Chip: A dual-core multithreaded processor[J]. IEEE Micro, 2004, 24(2): 40-47.
- [7] 奚杰, 陈杰, 朱玥. 利用 SystemC 实现多核系统的快速建模[J]. 微电子学与计算机, 2010, 27(7): 214-217.

作者简介:

金玉苹 女, (1981—), 硕士研究生. 研究方向为嵌入式系统结构设计、模式识别.

焦平 男, (1982—), 硕士研究生. 研究方向为嵌入式系统结构设计.