第 32 卷 第 16 期 计 算 机 工 程 2006 年 8 月
Vol.32 №16 Computer Engineering August 2006
·安全技术· 文章编号：1000—3428(2006)16—0130—03 文献标识码：A 中图分类号：TP309

IKE 协议的简化及安全性分析

陈华兴[1]，鲁士文[2]
（1. 中国科学院研究生院，北京 100040；2. 中国科学院计算技术研究所，北京 100080）

摘 要：IKE 协议的复杂性和安全性一直备受关注，文献[1]对 IKE 协议交换过程进行了简化，该文对简化后的 IKE 协议进行了安全性分析，并针对拒绝服务攻击 DOS 和中间人攻击的身份泄露，提出了改进建议。测试表明，该文提出的方法是可行的、有效的，大大降低了攻击的影响。

关键词：IPSec；IKE；密钥交换；安全性

Simplification and Security Analysis of IKE Protocol

CHEN Huaxing[1], LU Shiwen[2]
(1. Graduate School, Chinese Academy of Sciences, Beijing 100040;
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 The complexity and security of IKE protocol is always paid close attention by many researchers. Reference[1] simplifies the exchange of IKE. This paper discusses the security properties of the simplified IKE and presents some suggestions to solve the problems of denial of service attack and identification leak for man-in-the-middle attack. It is proved that the design is available and feasible. It reduces the influence of attack.

【Key words】 IPSec; Internet Key Exchange (IKE); Key exchange; Security

1 概述

IPSec[2]协议是IETF开发的一套互联网安全标准，为IP数据报提供数据完整性、机密性、数据源认证等安全服务。其中主要包括认证头(AH)、封装安全载荷(ESP)、因特网密钥交换(Internet Key Exchange, IKE)和ISAKMP/Oakley协议。

IKE[3]协议是IPSec中的自动密钥交换协议，用于动态地建立安全关联(SA)，为通信双方协商IPSec通信所需的相关信息如加密算法、密钥信息、通信方身份等。IKE建立在ISAKMP定义的框架上，并实现了 2 种密钥管理协议：OAKLEY和SKEME，它们是建立在多个协议基础上的混合型协议。

IKE 交换由以下 2 个阶段来完成。

(1)通信双方建立 1 个安全的通道的阶段，称为 ISAKMP SA，用于保护第 2 阶段中消息的安全；

(2)用 IPSec 协议建立起具体的 IPSec SA 阶段，确保通信双方的数据传输安全。

IKE 的阶段提供了 2 种模式的交换：主模式(Main Mode)和积极模式(Aggressive Mode)。

(1)主模式需要 6 条消息来完成

1)前 2 条消息进行 Cookie 交换和协商安全策略 SA，包括加密算法、散列算法及认证方法等；

2)中间 2 条消息交换 Diffie-Hellman 公开值和伪随机数 nonce；

3)最后 2 条消息认证 Diffie-Hellman 交换和身份信息。

(2)积极模式仅用 3 个消息来完成

1)第 1 条消息协商策略，交换 Diffie-Hellman 值、机数和身份消息；

2)第 2 条消息还要对响应者进行认证；

3)第 3 条消息对发起者进行认证，并提供交换参与方的身份证明。

主模式主要提供身份保护，积极模式的优势在于其速度。每种模式中都提供了 4 种不同的认证方法：预共享密钥认证，数字签名认证，公共密钥加密认证，修订过的公共密钥加密认证。4 种验证方式的详细过程见文献[3]。

2 对 IKE 协议交换过程的简化

简化后的交换过程的基本类型有如下 4 种：

(1)IKE_SA_INIT 交换用于 IKE SA 协商安全参数、传送 nonce 值和 Diffie-Hellman 值；

(2)IKE_AUTH 交换用于身份认证，并可产生第 1 个 CHILD_SA(用于其他协议的 SA，IPSec SA 等)。

(3)CREATE_CHILD_SA 交换对应于阶段 2 的交换，主要用于产生 CHILD_SA；

(4)INFORMATIONAL 交换的作用是对 IKE 进行事务管理，比如删除 1 个 SA、报告错误条件等。

IKE_SA_INIT 交换必须在其他的交换之前完成，接着是 IKE_AUTH 交换，其过程如图 1 所示。随后是多个可以任何顺序进行的 CREAT_CHILD_SA 交换和 INFORMATIONAL 交换。

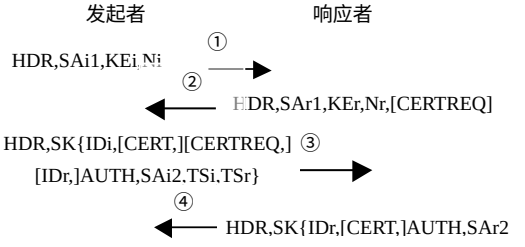


图 1 IKE_SA_INIT 和 IKE_AUTH 交换过程

消息①和消息②为 IKE_SA_INIT 交换，HDR 为 ISAKMP 头部，SAi 载荷说明了发起者所支持的各种加密算法，KEi 载荷发送了发起者的 Diffie-Hellman 值，Ni 是发起者的 nonce

作者简介：陈华兴(1976—)，男，硕士，主研方向：网络信息安全；鲁士文，研究员、博导

收稿日期：2005-08-08 **E-mail：**lu@ict.ac.cn

值。响应者从发起者提供的密码算法中选择 1 个“密码算法套件”并把它加入到 SAr1 载荷中，同时用 KEr 载荷来完成 Diffie-Hellman 值交换，用 Nr 来发送它的 nonce 值。IKE_SA_INIT 交换完成之后，双方都将会产生 1 个 SKEYSEED 值，对每种验证方法要分别计算 SKEYSEED 值。

对于公钥签名：SKEYSEED = PRF (Ni_b | Nr_b, g^xy)；

对于公共密钥加密：SKEYSEED = PRF (hash (Ni_b | Nr_b), CKY-I | CKY-R)；

对于预共享密钥：SKEYSEED = PRF (预共享密钥，Ni_b |Nr_b)。

SKEYSEED 是用来计算其他 7 种密钥的，包括 SK_d、SK_ai、SK_ar、SK_ei、SK_er、SK_pi 和 SK_pr。其中，SK_d 用来为这次 IKE_SA 交换所创建的 CHILD_SA 衍生新的密钥，SK_ai 和 SK_ar 用来为随后的 IKE_AUTH 交换验证其消息完整性的密钥，SK_ei 和 SK_er 是用来为随后的 IKE_AUTH 交换加密的的密钥，SK_pi 和 SK_pr 是产生了 AUTH 载荷时所用的密钥。后 6 种密钥是带方向性的，i 和 r 分别表示密钥的使用者是发起者和响应者。

消息③和消息④为 IKE_AUTH 交换，SK{…}表示其中的载荷是被用相应的密钥加密和完整性保护的。发起者用 IDi 载荷来声明它的身份，用 AUTH 载荷对第①条消息进行完整性保护，用 SAi2 来协商 1 个 CHILD_SA，TSi 和 TSr 分别表示发起者和响应者的通信选择符，也可以选择性地使用 CERT 载荷发送它的证书，用 CERTREQ 载荷来请求响应者发送证书，用 IDr 载荷指明发起者想要对话的响应者的身份。响应者用 IDr 来声称它的身份，可选择性地用 CERT 载荷发送其证书，用 AUTH 载荷来验证其身份，并对第②条消息进行完整性保护，用 SAr2 来选择 1 个 CHILD_SA。

简化后的 IKE 交换中，IKE_SA_INIT 交换和 IKE_AUTH 交换对应于阶段 1 交换，但没有主模式和积极模式之分，而且只用 2 次交换就完成了原来主模式所需的 3 次交换，并能附带产生 1 个 CHILD_SA。

3 安全性分析及改进

下面对简化后的 IKE 易受到的几种攻击进行了安全性分析，并提出了相应的改进建议。

3.1 拒绝服务攻击

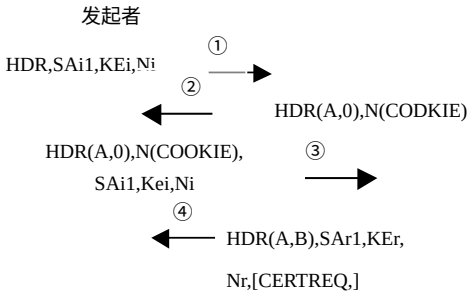
拒绝服务攻击(Denial of Service Attack)[4]是通过大量消耗系统资源(内存资源、CPU计算资源等)的方法，致使系统无法响应正常的处理请求，甚至陷入瘫痪。

在 IKE 的协商过程中，由于响应者在接收到第 1 条消息的时候即创建状态，因此恶意攻击者可以通过发大量的初始消息使得响应者不停地创建状态，耗费内存资源，最终导致内存耗尽，系统崩溃，这就是 IKE 容易受到的损耗内存资源的 DOS 攻击。另外，IKE 需要通过 Diffie-Hellman 交换进行密钥协商，其中的模幂运算会占用较大的计算资源。1 个 DoS 攻击者会通过 IP 欺骗的方法发起大量虚假交换请求，如果系统不能分辨出这些伪造的请求包，则不得不对伪造的请求进行大量模幂运算，这就是损耗 CPU 资源的 DoS 攻击。

IKE 提供了 Cookie 的交换，可以提供一定程度的抗 DoS 攻击，只要发起者确定响应者 Cookie 域中的 Cookie 和以前从响应者那里接收到的 Cookie 不同，它将抛弃消息；相似地，如果响应者确定发起者 Cookie 域中的 Cookie 和以前从发起者那里接收到的 Cookie 不同，它将抛弃消息，不再进行处理。

对于简化后的 IKE 交换来说，由于发起者事先不知道响应者的 Cookie，在发往响应者的第①条请求消息中响应者的 Cookie 将被置为 0，因此响应者就不可能知道这个创建 IKE_SA 的请求是否为 1 个虚假交换请求，因此简化后的 IKE 交换更容易受到 DoS 攻击。

为了抵抗 DoS 攻击，对 IKE_SA_INIT 交换过程作如下改进，如图 2 所示。



A：发起者的 Cookie 值　B：响应者的 Cookie 值

图 2 抗 DoS 攻击的 IKE_SA_INIT 交换过程

通常情况下，发起者和响应者只进行 1 次交换，发送 2 条消息：消息①和④。然而，当响应者检测到有大量消息①发送过来时，它不接收这些消息，而是发送 1 个不受保护的通知载荷作为响应(消息②)，并在其中包括它的 Cookie。发起者在接收到这条消息时，重新发送 1 条发起消息给响应者(消息③)，并在发送的消息中包含它接收到的响应者的 Cookie，响应者接收到消息，确认其中包括自己的 Cookie 时，才进行处理，否则丢弃。采用这种方式的交换可以有效地抵御损耗内存资源的 DoS 攻击和损耗内存资源的 DoS 攻击。因为在第 1 次的交换过程中，发起者和响应者都不需要更新状态，这样，如果攻击者发送大量的消息①给攻击目标，但是响应者在接收到这些消息以后，并不会创建新的状态，也就不会为保存状态参数在内存中分配空间，从而使得损耗内存资源的 DoS 攻击失去作用。而且在第 1 次交换过程中，发起者的计算量要大于响应者，发起者必须提供 SAi1、KEi 和 Ni 载荷，而响应者只是简单地计算一个 Cookie 值作为回应，这样即使有攻击者要进行损耗 CPU 资源的 DoS 攻击，在被攻击目标的 CPU 资源耗尽之前，自己的 CPU 资源也早已耗尽，从而能有效地抵御这种攻击。

3.2 中间人攻击

中间人攻击(Man in the middle Attack)[4]是指攻击者不仅能够截获通信双方的消息，而且能够任意删除、修改，甚至插入新的消息。中间人攻击是一种很强的攻击方式，又称为主动攻击。攻击者模仿响应者的身份与发起者交换信息，产生了共享密钥K1；同时模仿发起者与响应者通信，产生了共享密钥K2。这样在接下来的通信中，攻击者作为通信双方间的“中转站”，就能够截获、解密双方的消息，甚至修改消息或发送新的消息。

IKE 协议通过 Diffie-Hellman 交换来生成密钥材料，它通过认证的方法防止中间人攻击。但对于简化后的 IKE 交换使用公钥签名认证方式时，由于第 1 个交换是以明文发送的，因此主动攻击者能够模拟通信双方分别与两方进行 Diffie_Hellman 交换。虽然主动攻击者无法伪造发起者的签名生成伪造的消息③发给真正的响应者，使整个交换最终不能完成，但它仍然可以利用与发起者完成的密钥交换值，根据图 1 中消息①、②协商的加密参数，解密消息③的内容，从而可获取发起者的身份信息。

为了保护发起者和响应者的身份，我们建议在

IKE_SA_INIT 交换中，分别用对方的公钥对伪随机数 nonce 进行加密，由于 nonce 是生成密钥的材料，攻击者不能获取 nonce 值，因此主动攻击者不能得到协商的密钥材料，也无法获取发起者和响应者的身份。Diffie-Hellman 密钥交换容易受到中间人攻击，如图 3 所示。

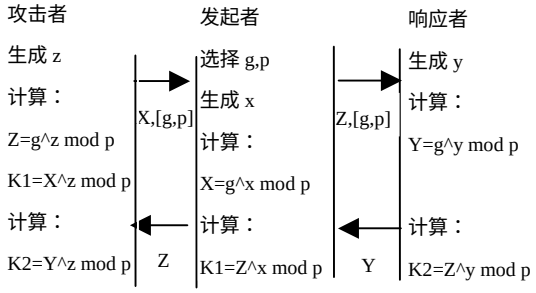


图 3 中间人攻击

在实际应用中，响应者一般作为服务器端，身份保护的意义不大，甚至有时身份信息是公开的，而发起者的身份则应本得到保护。为了保护发起者的身份，还可以使用扩展认证协议(Extensible Authentication Protocol, EAP)的方法，发起者通过在消息③中省略 IDi 载荷来表明它使用 EAP 认证方法的愿望，如果响应者同意使用 EAP 认证方法，它将把 EAP 载荷放在消息④中，并推迟发送 SAr2、TSi 和 TSr，直到在随后的 IKE_AUTH 交换中完成对发起者的认证。使用 EAP 认证方法的 IKE 交换能够保护发起者的身份，窃听者和中间人攻击者都不能获得发起者身份，但响应者身份可能被攻击者获得。在不需要保护响应者身份的情况下，可以采用这种方式，其过程如图 4 所示。

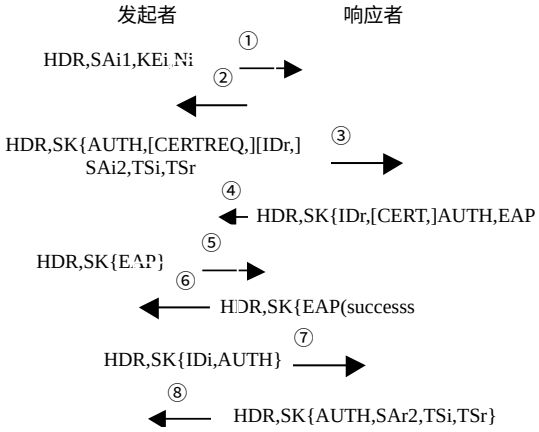


图 4 使用 EAP 的 IKE 交换过程

参考文献

1 Kaufman C. Internet-draft draft-ietf-ipsec-ikev2-17.txt[S]. RFC4306, 2004-09.

2 Kent S, Atkinson R. Security Architecture for the Internet Protocol[S]. RFC2401, 1998-11.

3 Harkins D, Carrel D. The Internet Key Exchan-ge[S]. RFC2409, 1998-11.

4 宋育芳, 张宏科. Internet 密钥交换协议的安全性分析[J]. 计算机工程与应用, 2004, 40(8): 136-139.

(上接第 108 页)

该应用设计方案实现后(路由表有 40 个表项)，经由 Intel 模拟平台(Workbench)中测试，得到如图 1 的结果。

通过以上试验数据可见，该应用设计基本上达到了设计要求。

3 总结展望

网络处理器内部一般都是多核(multi- core)的结构。一般可以分为两种：(1)具有一般运算能力和指令存储能力的处理单元PEs(processing elements)；(2)能够完成特定处理任务的功能模块FUs(function units)，如CRC校验单元等。现有的商业网络处理器中，这两种单元一般采用以下两种组织机制：流水线：每个核被设计成具有特定处理功能的模块，这些模块以流水线方式组织在一起完成分组的处理。这种结构的网络处理器主要有：Cisco的PXF，Motorola的C-5 DCP等。并行处理：每个处理单元PE都可以完成相似的任务，多个处理单元彼此间可并行执行。这种结构的网络处理器主要有：Intel的IXP1200, IBM的PowerNP, Agere的PayloadPlus等[5]。

通过本文的论述可见，如果网络处理器与 Intel IXP 系列处理器特性相同(同为并行处理)，本文给出的几个公式均适用。同时本文提出的几个公式显然不能完全适用于按流水线组织的处理器类型，因此下一步的研究重点将是给出适用于这种类型的网络处理器应用设计的公式。

参考文献

1 Lakshmanamurthy S, Liu Kin-Yip, Yim Pun, et al. Network Processor Performance Analysis Methodology[J]. Intel® Technology Journal, 2002, 6(3): 19-28.

2 Intel Corporation. IXP1200 Network Processor Family Hardware Reference Manual[Z]. http://developer.intel.com/design/network/ixa.html, 2001.

3 Intel IXP2400 Network Processor[Z]. http://www.intel.com/design/network/products/npfamily/ixp2400.htm, 2004.

4 Intel IXP2800 Network Processor[Z]. http://www.intel.com/design/network/products/npfamily/ixp2800.htm, 2004.

5 谭章熹, 林 闯, 任丰源等. 网络处理器的分析与研究[J]. 软件学报, 2003, 14(2): 255-265.

6 Wolf T, Franklin M A. CommBench — A Telecommunications Benchmark for Network Processors[C]. Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, 2000: 154-162.

7 Memik G, Mangione-Smith B, Hu W. NetBench: A Benchmarking Suite for Network Processors[C]. Proceedings of the International Conference on Computer-aided Design, 2001: 39-43.

8 Audenaert S, Chandra P. NPF Benchmarking Working Group Co-chairs, Network Processors Benchmark Framework[R]. NPF Benchmarking Workgroup, http://www.npforum.org/.

9 Johnson E J, Kunze A R. IXP2400/2800 Programming[M]. Intel Press, 2004.