

IP Security Protocol Working Group (IPSEC)
INTERNET-DRAFT
[draft-ietf-ipsec-dhcp-over-ike-00.txt](#)
Expires: 2 October 2003

T. Kivinen
2 April 2003

DHCP over IKE

Status of This Memo

This document is a submission to the IETF IP Security Protocol (IPSEC) Working Group. Comments are solicited and should be addressed to the working group mailing list (ipsec@lists.tislabs.com) or to the editor.

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Abstract

This document describes method of getting the IP security (IPsec) remote access client configuration information from the security gateway by tunneling dynamic host configuration protocol (DHCP) packets inside the internet key exchange (IKE) version 2 protocol.

INTERNET-DRAFT

2 April 2003

Table of Contents

1.	Requirements	2
2.	Introduction	2
3.	DHCP-over-IKE protocol	3
4.	DHCP packet formatting	6
5.	DHCP payload	7
6.	Format of DHCPv4 packet	7
7.	Typical DHCPv4 packet exchange	10
8.	Normative References	12
9.	Non-Normative References	12
10.	Authors' Addresses	12

[1.](#) Requirements

The DHCP-over-IKE protocol described here fullfills the following requirements:

- o Get the IP address for the client "in time" for use in the Traffic Selectors.
- o Protocol must be able to be use any method of getting the actual configuration data in the security gateway end. This is including but not limiting to: RADIUS, DHCPv4, LDAP, DHCPv6, RS/RA-stateless-IPv6, Diameter, PIB, or local configuration information database in the security gateway.
- o The configuration protocol might be tied to the EAP authentication progress, i.e for some cases the authentication of the client must be done first before the security gateway can get the configuration information for that client.
- o It should be possible for the client to get the same address each time it requests it. This implies some kind of persistent "hardware"/node identifier (note "hardware"/node ID != IKE ID).

2. Introduction

The basic protocol is to put the DHCP [[RFC2131](#)] packets inside the IKEv2 packets as separate payload(s). This does not imply that either end actually has to use DHCP for configuration, but we are simply reusing the existing packet format, which allows wide variety of configuration options. This also allows us to support all current and future configuration needs as defined in the DHCP protocol.

The basic flow of the DHCP exchange is:

```
Server 1                Client                Server 2
-----                -----                -----
<----- DHCPDISCOVER (broadcast) ----->
```

[I.](#) Kivinen

[page 2]

INTERNET-DRAFT

2 April 2003

```
DHCPOFFER ----->

<----- DHCPOFFER

<----- DHCPREQUEST (broadcast) ----->

<----- DHCPACK
```

The DHCPDISCOVER and DHCPOFFER packets might be missing if the client already have configuration and only wants to request that it can actually use it. The actual binding for the address happens during the DHCPACK packet.

3. DHCP-over-IKE protocol

The protocol flow of the DHCP-over-IKE combined with EAP is as follows:

```
Client                Security Gateway
-----                -----
HDR, SAi1, KEi, Ni    -->

<-- HDR, SAr1, KEr, Nr,
    [CERTREQ]

HDR, SK {IDi, [CERTREQ,]
    [IDr,] SAi2,
    TSi, TSr, DHCP}   -->
```

```

                                <-- HDR, SK {IDr, [CERT,]
                                    AUTH, [EAP,] DHCP}

(following packets can be repeated as many times as needed)
HDR, SK {[EAP,] [DHCP]}      -->

                                <-- HDR, SK {[EAP,] [DHCP]}

HDR, SK {[EAP,] [AUTH,]
          [DHCP] }          -->

                                <-- HDR, SK {[EAP,] [AUTH,],
                                    [DHCP,] SAr2,
                                    TSr, TSr }

```

The first two packets are the normal IKE_SA_INIT exchange. The IKE_AUTH phase following that can take as many steps as needed to finish both the EAP and DHCP protocols. The EAP protocol is finished when the security gateway sends EAP(success) packet. The DHCP protocol is finished when the security gateway sends DHCP(ACK) packet.

In the first IKE_AUTH packet the client will include the DHCP payload indicating it wants to have configuration from the security gateway. This DHCP payload may either be DHCPDISCOVER (the client does not already have address, i.e it is in DHCP init state) or DHCPREQUEST

(client is requesting for previously given address, i.e it is in DHCP init-reboot state). If the first packet is DHCPDISCOVER then the security gateway will eventually reply with DHCPOFFER payload, and then client sends DHCPREQUEST payload. When security gateway replies to the DHCPREQUEST with DHCPACK then the configuration protocol is finished.

If the first DHCP payload is DHCPREQUEST, the reply might either be DHCPACK (finishing the configuration) or DHCPNAK, meaning that the client was denied to use of requested address. After DHCPNAK the client MUST start the configuration protocol from the beginning by sending DHCPDISCOVER packet (i.e move to the DHCP init state). If the paramters are not accetable the security gateway SHOULD reply with DHCPNAK instead of waiting for the client to timeout the DHCP rebooting phase and fall back to DHCP init state (i.e back to sending DHCPDISCOVER).

The EAP process might be going on before, after or simultaneously to the configuration progress. I.e in case of RADIUS [[RFC2865](#)] backend is used then the configuration data might be only known after the authentication

process is finished, in which case the security gateway will postpone replying to the DHCPDISCOVER or DHCPREQUEST until EAP process is finished.

The AUTH payload in the reply to the first IKE_AUTH packet is normal authentication packet of the security gateway, i.e either the signature or the MAC depending which authentication method is used. If the EAP method is such that it creates a shared key between the client and security gateway then the packets having the final EAP payloads MUST also have AUTH payloads created using that shared key.

When the security gateway detects that both EAP and DHCP configuration is finished it will send reply packet containing the final EAP or DHCP payload (depending which of those ended last), and payloads needed to complete the CREATE_CHILD_SA (i.e SAr2, and Traffic Selectors).

When using DHCP to request configuration data from the security gateway the client MUST use wildcard address range in traffic selectors when starting to create the SA, and the security gateway MUST then narrow the traffic selectors down. I.e the client uses address range from 0.0.0.0 to 255.255.255.255 in TSi and TSr when sending the SAI2. The protocol and port range can be specified, but address normally cannot be, as it is not yet known (the client is requesting it from the security gateway).

The security gateway MUST narrow the traffic selectors so that the TSi includes only the configured IP address of the client, and the TSr SHOULD include the subnet(s) accessible through the security gateway.

Note, that since the security gateway might be using external configuration backend, the client needs to use normal DHCP retransmission policies when sending the requests. I.e if the security gateway does not reply with a packet containing DHCP payload, the client should send new IKE packet having the same DHCP payload (but new message id), so that security gateway has possibility to sending back the reply

(security gateway cannot send the DHCP packet before client sends next request to it), and if client is trying to use DHCPREQUEST and does not get any reply back (or receives DHCPNAK) then client starts again with DHCPDISCOVER.

Security gateway might also send multiple DHCPOFFER packets, and client can select the one that suits it best to be used.

The security gateway SHOULD wait for configurable time (few seconds) for the replies from external configuration backends and collect all replies to the reply packet (as separate DHCP payloads) going back to the client. It can also send reply back immediately when it has first reply, to fasten up the process (for example in case where it knows there is only one DHCP server in the network). If no replies are received during that time the security gateway MUST send reply to the IKE packet, without the DHCP payload.

Example data flow with new client doing DHCP and one-time-password EAP:

Client -----	Security Gateway -----
HDR, SAi1, KEi, Ni	-->
	<-- HDR, SAr1, KEr, Nr, [CERTREQ]
HDR, SK {IDi, [CERTREQ, [IDr,] SAi2, TSi(0.0.0.0-255.255.255.255), TSr(0.0.0.0-255.255.255.255), DHCP(DISCOVER)} -->	
	<-- HDR, SK {IDr, [CERT,] AUTH, EAP(OTP-Challenge), DHCP(OFFER, ip=1.2.3.4)}
HDR, SK {EAP(OTP-Reply), DHCP(REQUEST, ip=1.2.3.4) } -->	
	<-- HDR, SK {EAP(success), DHCP(ACK, ip=1.2.3.4), SAr2, TSi(1.2.3.4-1.2.3.4), TSr(0.0.0.0- 255.255.255.255)}

Another example doing signature authentication for both ends, and client requesting reuse of the previous address:

Client -----	Security Gateway -----
-----------------	---------------------------

```

HDR, SAi1, KEi, Ni          -->

                                <-- HDR, SAr1, KEr, Nr,
                                    [CERTREQ]

HDR, SK {IDi, [CERT],
        [CERTREQ,] [IDr,]
        AUTH, SAi2,
        TSr(0.0.0.0-255.255.255.255),
        TSr(0.0.0.0-255.255.255.255),
        DHCP(REQUEST,
              ip=1.2.3.4)}    -->

                                <-- HDR, SK {IDr, [CERT,]
                                    AUTH,
                                    DHCP(ACK, ip=1.2.3.4),
                                    SAr2,
                                    TSr(1.2.3.4-1.2.3.4),
                                    TSr(0.0.0.0-
                                       255.255.255.255)}

```

4. DHCP packet formatting

The DHCP packet is filled as normally, except the client will use hardware type (htype) of 31 signifying an IPsec tunnel mode virtual interface. The chaddr field MUST include an identifier unique to the virtual subnet. The client MUST use the same chaddr field in all subsequent messages within the same exchange. In addition, the chaddr SHOULD be persistent between reboots so that the DHCP server will be able to re-assign the same address if desired.

The hlen and chaddr fields SHOULD be determined as follows:

- o If one or more LAN interfaces are available, the hlen and chaddr fields SHOULD be determined from the active LAN interface with the lowest interface number. If no active LAN interface is available, then the parameters SHOULD be determined from the LAN interface with the lowest interface number. This enables the chaddr to be persistent between reboots, as long as the LAN interface hardware is not removed.
- o If there is no LAN interface, the chaddr field SHOULD be determined by concatenating x'4000', the IPv4 address of the interface supplying network connectivity, and an additional octet. The x'4000' value indicates a locally administered unicast MAC address, thus guaranteeing that the constructed chaddr value will not conflict with a globally assigned value.

packet type.

The payload type for the DHCP Payload is sixteen (16). This document describes the DHCPv4 packet type, and other documents may later describe other packet formats (for example DHCPv6).

If unknown DHCP Payload packet type is received then UNKNOWN-DHCP-PAYLOAD-TYPE notification MUST be sent back. This allows future versions to probe if the newer packet type is supported or not (i.e if server responds with notification, then it is not supported, if it replies or does not reply at all immediately, then it is supported).

6. Format of DHCPv4 packet

The typical packet put inside the DHCP Payload will be (described here

I. Kivinen

[page 7]

INTERNET-DRAFT

2 April 2003

only to make it easier for implementators to understand what goes where, see [[RFC2131](#)] for full details):

```

                                1           2           3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!           op           !      htype      !      hlen      !      hops      !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     xid                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!           secs           !           flags           !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     ciaddr                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     yiaddr                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     siaddr                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     giaddr                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     chaddr                                     !
!                                     (16 bytes)                             !
!                                     !                                     !
!                                     !                                     !
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
!                                     sname                                     !
!                                     (64 bytes)                             !
~                                     ~
```

```

!
+-----+
!
!                               file
!                               (128 bytes)
!
~
!
+-----+
! byte 99 (0x63)!byte 130 (0x82)! byte 83 (0x53)! byte 99 (0x63)!
+-----+
!      opt53      ! option len (1)!  opt53 value !      opt50      !
+-----+
! option len (4)!                requested IP address          !
+-----+
! .....          !      opt255          !
+-----+

```

- o op - BOOTP message option code / message type, 1 = BOOTREQUEST (from client to server), 2 = BOOTREPLY (from server to client)
- o htype - Hardware address type, 31 = IPsec tunnel mode virtual interface [[RFC3456](#)].
- o hlen - Hardware address length = 6.
- o hops - Client sets to zero, optionally used by relay agents when

booting via a relay agent.

- o xid - Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.
- o secs - Filled in by client, seconds elapsed since client began address acquisition or renewal process.
- o flags - Flags (normally zero).
- o ciaddr - Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests, i.e in normal initial exchange always zero, i.e client fill with zeros and the server copies from the request packet.
- o yiaddr - 'your' (client) IP address. Never filled in by the client, sent by the server to client telling the clients new IP address.

- o siaddr - IP address of next server to use in bootstrap. Normally zero.
- o giaddr - Relay agent IP address, used in booting via a relay agent. Packets inside the DHCP payload this is normally set to zero.
- o chaddr - Client hardware address. Filled in by the client as specified in [section 4](#).
- o sname - Optional server host name, null terminated string. Normally zero.
- o file - Boot file name, null terminated string; "generic" name or null in DHCPDISCOVER, fully qualified directory-path name in DHCPOFFER. Normally zero.
- o byte 99, 130, 83, 99 - Magic cookie.
- o opt53 - DHCP Message Type.
- o option len - Length of the option in bytes not including the option tag and this length byte.
- o opt53 value - Type of the DHCP message. Client normally first send DHCPDISCOVER, and the server replies with DHCPOFFER. Then client finishes the exchange by sending DHCPREQUEST and server verifies that exchange is done by DHCPACK.

Type	Value
====	=====
DHCPDISCOVER	1
DHCPOFFER	2
DHCPREQUEST	3

DHCPDECLINE	4
DHCPACK	5
DHCPNAK	6
DHCPRELEASE	7
DHCPINFORM	8

- o opt50 - Requested IP Address. Only present in the DHCPREQUEST, when client requests to be allowed to use this IP address.

- o requested IP address - IP address client requests to use. Client copies the ciaddr field from the DHCP OFFER to here when sending DHCPREQUEST.
- o opt255 - End of options.

Some other useful options [[RFC1533](#)] and their formats are:

- o Pad - opt0, no data. Used for padding.
- o Subnet mask - opt1, data: 4 subnet mask bytes.
- o Domain name server - opt6, data: 4 * N address bytes for N dns servers.
- o Hostname - opt12, data: N bytes of hostname.
- o NBNS - opt44, data: 4 * N address bytes for N NetBIOS name servers.
- o IP address lease time - opt51, data: 4 bytes, lease time in seconds.
- o Server identifier - opt54, data: 4 bytes, address of the DHCP server, send by server, and copied back in client if known.
- o Parameter request list - opt55, data: N bytes, each byte specifying one option which is requested from the server (client sends this, server tries to put all these options in its reply).
- o Option overload - opt52, data: 1 byte. Value 1 means that the file field contains options instead of its normal contents. Value 2 means that sname fields contains options, and value 3 means that both of those fields contain options.

[7.](#) Typical DHCPv4 packet exchange

The client will first create DHCPDISCOVER payload, and fill in the following data:

- o op - 1 = BOOTPREQUEST
- o htype - 31 = IPsec tunnel mode virtual interface
- o hlen - 6

- o xid - random transaction id
- o chaddr - hardware address, as specified in [section 4](#).
- o magic - bytes 99, 130, 83, 99
- o DHCP Message Type option - bytes 53, 1, 1
- o Parameter request list option - bytes 55, n, n bytes of option numbers, of which the client wants to request.
- o End of options - bytes 255

All other fields are filled with zeros.

The server will reply with DHCP OFFER payload, with following data:

- o op - 2 = BOOTREPLY
- o yiaddr - IP address offered for client
- o magic - bytes 99, 130, 83, 99
- o DHCP Message Type option - bytes 53, 1, 2
- o Server identifier - bytes 51, 4, IP address identifying the server giving out information (DHCP server, security gateway etc).
- o Other configuration options.
- o End of options - bytes 255

All other fields are copied from the DHCPDISCOVER packet. DHCP options are not copied from the DHCPDISCOVER packet.

Those first two packets can be ignored if client already has IP address and wants to try to use that one.

Client will request to use given address by sending DHCPREQUEST payload, with following data:

- o op - 1 BOOTREQUEST
- o htype, hlen, xid, chaddr - same as DHCPDISCOVER (== copied from DHCP OFFER).
- o yiaddr - filled with zero
- o magic - bytes 99, 130, 83, 99

- o DHCP Message Type option - bytes 53, 1, 3
- o Requested IP Address - bytes 50, 4, IP address to be requested (i.e the yiaddr from the DHCP OFFER or the old IP address).

- o Server identifier - bytes 51, 4, IP address - copied from the DHCP OFFER (if trying to use previous IP address this should be filled with the value from the previous run).
- o Parameter request list option - bytes 55, n, n bytes of option numbers, of which the client wants to request.
- o Other options can be copied from the DHCP REQUEST payload.

The server will reply with DHCPACK payload, and finish the configuration protocol. The DHCPACK is filled with following data:

- o op - 2 = BOOTREPLY
- o yiaddr - IP address offered for client
- o magic - bytes 99, 130, 83, 99
- o DHCP Message Type option - bytes 53, 1, 5
- o Server identifier - bytes 51, 4, IP address identifying the server giving out information (DHCP server, security gateway etc).
- o Other configuration options.
- o End of options - bytes 255

The option should be same that what was in the DHCP OFFER payload.

8. Normative References

[RFC2131]

Droms R., "Dynamic Host Configuration Protocol", March 1997

9. Non-Normative References

[RFC2865]

Rigney, C., S. Willens, A. Rubens, and Simpson W., "Remote

Authentication Dial In User Service (RADIUS)", June 2000.

[RFC1533]

Alexander S., and Droms R., "DHCP Options and BOOTP Vendor Extensions", October 1993.

[RFC3456]

Patel B., B. Adoba, S. Kelly, and Gupta V., "Dynamic Host Configuration Protocol (DHCPv4) Configuration of IPsec Tunnel Mode", January 2003.

[10.](#) Authors' Addresses

Tero Kivinen
SSH Communications Security Corp

[I.](#) Kivinen

[page 12]

INTERNET-DRAFT

2 April 2003

Fredrikinkatu 42
FIN-00100 HELSINKI
Finland
E-mail: kivinen@ssh.fi

--

kivinen@ssh.fi

SSH Communications Security

SSH IPSEC Toolkit

<http://www.ssh.fi/>

<http://www.ssh.fi/ipsec/>