Internet Engineering Task Force
INTERNET DRAFT
Expires in September 2001

Nicolas Prigent
Jerome Marchand
Francis Dupont
ENST Bretagne
Bernard Cousin
IRISA
Maryline Laurent-Maknavicius
Julien Bournelle
INT Evry

DHCPv6 Threats

<draft-prigent-dhcpv6-threats-00.txt>

Status of this Memo

Abstract

This document addresses numerous security problems which DHCPv6 could be a victim of. We believe DHCPv6 will be the configuration method of choice. Despite of this fact, we think we have to describe a number of security issues one must be aware of in case of using DHCPv6 in an open environment.

Contents

1. Introduction

This document addresses numerous security problems which DHCPv6 could be a victim of. We believe DHCPv6 will be the configuration method of choice. Despite of this fact, we think we have to describe a number of security issues one must be aware of in case of using DHCPv6 in an open environment.

2. Aim of attacks

This section presents a list as complete as possible of the motives one could have to attack a DHCP system.

2.1. Gaining access to the service

This is the most obvious attack. The aim of the attacker is to use a service it isn't allowed to. This could consist in obtaining an IP address from a server, or in gaining access to special services on the network.
For example, an attacker could obtain the address of DNS server and use it.

2.2. Gaining information about the network

Sometimes, having some information about network is an interesting knowledge for an attacker. From another point of view, the authority on the domain should like to avoid an attacker to be able to know what are the configuration parameters on the network the authority is responsible of, or what services are currently available.

2.3. Avoiding authorized users to access the service, also known as Denial of Service

In particular cases, the aim of the attacker will be to create a Denial of Service.

This is particularly true if DHCP is used in an open environment, for instance in a public network access service.

2.4. Getting confidential information about the client

Despite of the fact this is not the main concern, this is a problem we have to be aware of. Purposes are the same as in traffic analysis. If an attacker knows that a mobile node is currently attached to a specific network, this probably means that its user isn't far away.

Other aims could be to cause client to send datagrams to a particular faked router in order to get them, etc.

3. Means of attack

To fulfill its goals, attacker can use a large amount of techniques. In this section, we will describe them, sorted by aim.

3.1. Gaining access to service

In most of cases, the attacker will claim identity of a valid client. We can first notice that concerning the Solicit message, it isn't a huge importance to identify user that sent it. In fact, anyone can send it, even an attacker, given that it will not cause the creation of configuration state, nor involve emission of a Reply message that will give important information.

To gain access to service, the attacker can use Request, Renew and Rebind messages, because only these messages modify configuration.

First of all, attacker can create a Request message from scratch, then send it to server. Without authentication, server can't know that an attacker isn't a valid user, and will send a Reply message containing configuration parameters.

Use of Renew and Rebind messages is a bit more clever. It requires that attacker waits for a valid client to get out of local network without sending a Release message. This point is, by itself, an error from a security point of view: during the remaining lifetime of the IP address, the attacker can use it while its real owner is out. Moreover, if the attacker sends Renew or Rebind messages, it can keep them for a very long time.

With standard DHCPv6 protocol, anyone can create this kind of message and gain access to network, because no authentication is required. In [4], a special option is proposed for DHCPv4 which enables authentication for messages. But this method require that a pre-established shared secret exists between client and server, which is a quite strong pre-requisite, in particular in case of roaming between wireless IP networks.

Another mean to gain access to network would be to wait for a client to ask for parameters, then to get the Reply message sent by server at the same time as client. Then, when the client does his Duplicate Address Detection, attacker replies it is using this address (which attacker knows, given that it accessed Reply message). Client looses its legitimate address, while attacker gains one.

Given that no confidentiality service is proposed with DHCPv6,

there is, up to now, no mean to protect from this. We should consider to establish a mean of proving ownership of an address given by a DHCP server. But this consideration are up to now out of the scope of this draft.

In the last section, we will explain how using AAA could ensure confidentiality. However, cypher is CPU expensive and should be used with care.

3.2. Gaining information about the network

The first mean of attack to gain information is by getting them using standard protocol. The attacker sends a Solicit message, with an option-request option containing identification of required informations. Note that if authentication is ensured,

this technique will not work any longer.

However, if the attacker listens to information passing through medium, it can get a lot of them. In fact, it is able to know everything a valid client can ask. It simply has to wait for someone to ask it.

This second threat exists because messages in DHCPv6 aren't cyphered. So anyone can access to information going through medium.

3.3. Creating a Denial of Service

There are a lot of means to involve a Denial of Service against DHCPv6 service. We can classify them according to the target of the attack.

3.3.1. Attacks against server

The main attack against server will try to preclude servers from doing their work. There are different ways to do it.

A simple attack will be to prevent servers from assigning an IP address to a legitimate station due to the attacker reserving all the available addresses. This implies the attacker to make a lot of Request messages, until the servers are not able to assign an address any longer.

Another attack consists in overloading servers of useless work. This can be done in several ways. For example, if a lot of faked clients try to obtain services from a server at the same time, this one will have much work to do, and will not be able to serve legitimate clients. If we suppose the use of cryptography, it's possible to overload server by giving it a lot of faked cyphered information that force it to do useless computation. That's why we have to avoid, as much as possible, the use of asymmetric cryptography, which is well-known to be CPU cycles expensive.

Advertise messages can be used against server, because they aren't authenticated. To make the choice of which server it will use, a client looks at the preference option sent in each Advertise message it receives. The higher the value is, the more the server wants to be used, and more likely the client will use it. Suppose now that a server is a bit overloaded, and doesn't want to be used. In this case, it will reply to Solicit message from client with an Advertise having a low preference value. Given that the server is already overloaded, a small additional overload may be sufficient to create a Denial of Service. An attacker would have a huge interest in "helping" clients choosing the most loaded DHCP server. This can be done sending a faked Advertise message with server address field set to the most loaded DHCP server address, and preference field set to 255.

This will cause the client receiving it to send its Request message to the weakest server, weakening it even more. Due to authentication hasn't been realized when the server needs to send an Advertise message, this attack could possibly not be avoided [we are requesting for feed back about this]. If we suppose we can bring little modification to standard DHCP protocol, it's possible to propose a solution to this problem. What a server needs is to be sure that the client is replying to the Advertise message it sent, and not to another forged one. In the current protocol, client's Request messages have their preference values equal to 0. This field could be used as a proof of liveness: If the client sets it to the received preference value, the server knows that the client is sending a reply to its Advertise message, with preference value it gave in. From server point of view, two cases are possible :

1 - Preference value of the Request message received by the server is different from the one sent in the Advertise. In this case, the server is informed that something abnormal happened.

2 - Preference value of the Request message sent by the client is equal to the one the server sent in its Advertise message. In this case, if we suppose that messages aren't modified onto the wire, or if integrity and/or authentication services are provided on the Request message, the server knows that the client chose it the right way. The fact the client sent it the Request message means that the chosen server was the one that made the best offer, because the preference value is correct.

An attacker can make the server believe wrong information to prevent it from doing good job. This could be done by forging Decline or Release messages, and sending one of them to server. This will cause the server to have a wrong configuration of assigned addresses, causing it probably not being able to fulfill its task.

3.3.2. Attacks against clients

Against clients, one can use the same kind of attacks that were used against servers, i.e. we can try to overload them by giving them too much work.

But there are other methods to attack clients.

An attacker can send an Advertise message with preference value equal to 255. This will cause the client to choose the attacker's machine as server without even waiting for other messages, and to send to it its Request messages. If the server doesn't answer, client will try again, and will never be able to obtain desired parameters.

An attacker can send Reconfigure-init messages too, provoking clients reconfiguration, and causing them to loose their current configuration parameters.

After obtaining a new address, the client has to check whether this one isn't already in use. It does a Duplicate Address Detection. An attacker could respond it is currently assigned, preventing the client from getting it.

An attacker can create faked Reply for client, with wrong parameters, which prevent it from correctly using network. It can for example be an already assigned address, or a wrong router address.

3.4. Getting confidential information about the client

The first manner would be to listen to the medium. While there's no cyphering, everything that goes through the wire (or air) can be seen in clear. This means that using a password as a shared secret between client and server isn't an efficient method. But this is not the only information a hacker can get listening to the medium. It can know who's trying to connect: When a client tries to obtain the same address it obtained the last time, it uses a special option, which gives anybody listening a hint to identify it.

Another manner to gain access to information about client is to masquerade server and to reply pernicious configuration parameters. For example, it's possible to send to client a router address which is in fact the attacker's one. This way, all messages supposedly sent out of the network through this router will in fact be redirected to the attacker.

Confirm messages enable an attacker to obtain confidential information too. An attacker sending to the server a Confirm message with client's Identity Association parameters will be replied with all configuration parameters of this client.

4. Concerns about relays

In previous sections, we haven't dealt with relays. In fact, we think relays are an aspect of DHCP that should be removed. We will explain our motivations in this section.

4.1. Why do relays exist in DHCPv6?

Relays allow clients which are not on the same link as the server to send to them their DHCP messages. At start, clients may have a link-local address, i.e. an address which scope doesn't enable them to communicate directly with DHCP servers located on other links. Addresses of this type will cause messages to be filtered while getting through a gateway or a router. So clients use the link-local multicast address "All DHCP agents" FF02::1:2 as a destination, and their own link-local address as origin. Relays get messages directed to this address, and forward them to servers.

4.2. Why should we suppress relays?

Despite of the fact that relays were useful in DHCPv4, we think they are no longer useful in DHCPv6. IPv6 have features that make us feeling we don't need relays any longer.

4.2.1. Relays cause overload

For clarity purpose, let's use a network example:
Suppose this network is well configured. C1, C2 and C3 use relay R1 to convey with server S1, C5 uses R2 to convey with the server S1, C7 uses R3 to convey with the server S1 and C4 directly convey with S1.

```
           A         B         C         D
           |         |         |         |
 +---+     |         |  +---+  |  +---+  |  +---+
 | C1|---|           |--| S1|  |--| C5|  |--| C7|
 +---+     |  +---+  |  +---+  |  +---+  |  +---+
           |--| R1|  |         |         |
 +---+     |  +---+  |  +---+  |  +---+  |
 | C2|---|           |--| X |--|--| X |--|
 +---+     |  +---+  |  +---+  |  +---+  |
           |--| X |--|         |         |
 +---+     |  +---+  |  +---+  |  +---+  |  +---+
 | C3|---|  +---+    |--| C4|  |--| R2|  |--| R3|
 +---+     |--|R1'|  |  +---+  |  +---+  |  +---+
           |  +---+  |         |         |
```

In the case the DHCP domain administrator wants redundancy to secure his network, a second relay R1' may be installed in parallel to R1. This way, if R1 fails, R1' will be able to replace it. First of all, let's note that if redundancy is not in use, a single relay could be victim of Denial of Service attacks.

Redundancy is one of the main interests of relays. But there is an important drawback to it. Each time a client (in this case, C1, C2 or C3) sends a message directed to "All DHCP Agents", R1 and R1' will both forward it. This means that for each message emitted by a client, there will be two messages forwarded by relays (if we consider two relays). Thus S1 will receive twice the message from clients on network A. If there are only a few clients, it's no big deal... However, an attacker should use this to overload servers. Each message sent generates 'n' messages that server will have to deal with, where 'n' is the number of relays located onto the link. This huge quantity of messages could cause a Denial of Service.

Some might say that we shouldn't then duplicate relays, and use no redundancy. In this case, we don't see the purpose of relays.

4.2.2. Relays cause problems for authentication

Another problem with relays appears if we want to introduce authentication. With a relay as the middleman of the communication, we must use three secured relationships; one between the client and the relay, one between the relay and the server and one between the client and the server. This is an overload for the client which may be a handset with low computation power.

In best cases (i.e. if we consider that network is secured once a message has been received by a relay), we still need security associations between clients and relays, which don't enable clients to do less work that if there were no relays.

What we would have to find is a solution that would replace relays. The main problem (and probably only problem) is that we need clients to be able to communicate with a server which is on another link, while the message using the multicast server address as a destination and/or the client's link-local address as a source is not allowed to pass through relays.

4.3. How could we replace relays

IPv6 offers a large amount of addresses scope. Up to new, DHCP clients use link-local addresses when they arrive on network to communicate with DHCP agents located on the same link.

Relays are required only because the link-local addresses obtained by clients are filtered by gateways and routers.

If we used site-local addresses instead of link-local addresses, it would be possible to stop using relays.

It is possible for a client to create a site-local address, using router advertisement. But we have to ensure that a faked client will not be able to use this site-local address to get unauthorized services.

This can be done by filtering methods placed on the routers. A message with a site-local address as a source will only be able to cross links if the destination address is addressed to "All DHCP servers".

So a client starts with a local Solicit message with its link-local address as source address and the multicast address FF02::1:2 (renamed "All Local DHCP Servers") as destination address . When the local server receives a local Solicit message, it replies with an Advertise message. If the client receives no response, it retries with a Site Solicit message made with its site-local address as source address and the multicast address FF05::1:3 (renamed "All Site DHCP Servers") as destination address.

Clients prefer link-local servers than site-local servers. But if this site local policy requires a client to be configured by a site server, the local server must not respond to it.

5. Means of protection

Now that we know attacks DHCP has to face to, we can define protective methods.

5.1. What kind of services do we have to offer?

First of all, we need to define services we have to provide.

5.1.1. Authentication

Obviously, authentication is the most important service to provide. Clients and servers must be mutually authenticated, as well as relays must be authenticated, if still required.

Authentication consists in two phases. First of all, candidate to authentication needs to identify itself, i.e. to claim an identity. This means that we have to use an identification system. Then, the client has to prove this identity. This is in most case done using a shared secret.

5.1.2. Authorization

Once the identity has been proved, we have to ensure that the client is authorized to obtain services it asks for. For example, a client allowed to obtain one address is not allowed to take many of them, leaving server with no more addresses available. By the way, some clients can access to particular services that will be refused to others.

5.1.3. Confidentiality

In previous sections, we assumed that there were no confidentiality while using DHCPv6. But we showed that some attacks were using this lack to obtain important information about configuration. Thus, DHCP should use methods ensuring confidentiality.

5.2. Which messages do we have to protect?

Considering that using security services increases the cost in CPU cycles, we need to ensure protection only when required, but each time required. Some messages can't (and needn't to) be secured.

5.2.1. Solicit messages

Solicit messages can't be secured, because we can't suppose that previous security relationships exist between pairs. Moreover, it doesn't seem necessary to protect Solicit messages, given that faked Solicit messages can't harm any DHCP systems.

5.2.2. Advertise messages

Advertise messages may be authenticated, due to risks of Denial of Service they could provoke. Despite of this fact, cyphering them would require to have a pre-established security association between hosts, or to establish dynamic one with the previous Solicit message. This can't be done, given that using a Solicit messages isn't a requirement for clients. So we will consider that Advertise messages aren't secured. To detect faked Solicit messages, a client should (for example) consider that a server which has sent to it an Advertise message, and, when requested three times, never answered, is a faked server and must be ignored and removed form its server list if other servers are present.

But this isn't sufficient. This mean of protection supposes that a faked server always gives the same IPv6 source address.
Consider now that it uses a new address at each Solicit message sent by the client. It will not be possible to identify faked server any longer, and our previous technique will not work.
A heuristic consists for the client, when sending a Solicit message, in making a list of all received Advertise messages.
If it receives an Advertise message with preference equal to 255, it sends a Request message to corresponding server, but the client has to continue making a list of all Advertise messages it gets. If the server who sent the Advertise message with preference value equal to 255 doesn't answer an arbitrary number of times, the client has to choose the server which, in the list, has the next highest preference number. This way, we know that client will eventually send a Request message to a legitimate server.

5.2.3. Request messages

Request messages must be authenticated, or else an attacker will be able to use them to access some services, or to provoke a

Denial of Service. Given that a Request message can contain an IA option, it should be cyphered too. Two cases are possible :

- This is the first time the client uses the server. In this case, the IA option will probably not be used, or will not contain any useful information concerning previous configurations, given the fact that there are no previous relationship between client and server. The security association will be created during this phase, and no IA option is required.

- Client and server know each other. In this case, we can suppose they have established a shared secret during a former session. So they can use it to cypher their communication. If the server has forgotten this shared secret, we can guess that the information contained in the IA option was forgotten too.

5.2.4. Confirm, Rebind and Renew messages

Confirm, Rebind and Renew messages must be authenticated, and cyphered because they contain useful information. They usually come after Request/Reply messages have been exchanged, so we can suppose the same security association will be used.

5.2.5. Reply messages

Because they are a very sensible part of the protocol, Reply messages must be authenticated and cyphered. The client receiving a Reply message must be sure it comes from a legitimate server, and that it is the only one able to access it.

5.2.6. Decline and Release messages

Decline and Release must be authenticated, because a server receiving one must be sure it comes from a legitimate client.

5.2.7. Reconfigure-init messages

Given the opportunity to create a Denial of Service Reconfigure-init messages offer, they must be authenticated. This ensures clients receiving one that it's not faked, and that they really need to start a reconfiguration process.

5.3. Why can't we use IPsec ?

The problem with IPsec is that it requires pairs to have IP addresses. In the case of DHCP, we obviously can't be sure of it.

5.4. So what can we do ?

5.4.1. Consideration about computation power

First of all, we have to suppose that clients could be mobile handsets, with weak computation power. This means that we can't consider they are able to do a lot of cyphering. By the way, using asymmetric cryptography every time would be dangerous, creating risks of Denial of Service. That's why clients must use asymmetric

cyphering as few as possible. However, it's probably not necessary (and even not possible) to stop using asymmetric cyphering at all. Moreover, this implies finding a way to create dynamic security associations, i.e. to exchange keys between client and server.

5.4.2. Consideration about security relationship

We have to consider that there are no reason for having a pre-established security relationship between client and server that cooperate for the first time. This implies that we have to use an external mean of authentication. Since the client isn't supposed to be able to access to the network, it can't contact directly a distant server for authentication. However, it can use services from local DHCP server to do it (provided it's a legitimate DHCP server). This seems to be quite dangerous. How can a client be sure it isn't sending its password/credential/whatever to a fake server? To be honest, we think [but here again, we are waiting for feedback] that it can't. But it's no big deal, if the information it sends is useless to a faked server.

Given circumstances, mutual authentication will require help from a third party. The latter will be in charge of the authentication of servers for clients, and vice-versa. We will describe later how using AAA makes this possible.

Identication for each client must employ an unique identifier, to be able to distinguish them. Each client will need to have its own security association with the server. Thus, a client will not be able to masquerade as another one.

5.4.3. Consideration about multicast addresses for clients

Given new applications for DHCPv6 (mobile IP for example), mutual authentication is mandatory. But this requires a greater amount of CPU cycles. In [1], one can find propositions about using a multicast address for DHCP clients when reconfiguration is necessary. With standard methods, this will create security issues. There are three cases:

1 - The Reconfigure-init message is not authenticated. This means that anyone can send a Reconfigure-init message in place of servers, and force clients to abort current configuration, causing a Denial of Service. Obviously, this is unacceptable.

2 - The Reconfigure-init message is signed with the server's private key, providing authentication. This means that each client receiving a Reconfigure-init message must verify it with the server's public key. But this involves a great amount of CPU cycles. Given that DHCP clients could be mobile handsets, with low computation power, it should be easy for an attacker to send faked messages. Clients would have to check every Reconfigure-init message, preventing them from doing anything useful, thus causing a Denial of Service.

3 - The Reconfigure-init message is signed with a secret symmetric key shared by clients and servers, enabling easier computation than asymmetric authentication methods. However, by this way, any client is able to create a faked Reconfigure-init message and to send it to other clients, causing them to re-init their configuration. One should say that it would be possible to use a different key for each client. We totally agree with that, but in this case, using a multicast address would be of small interest.

Considering this, we can suppose that using a multicast address is not a good idea. Usually, using methods that reduce differences between entities is not a good thing when we need Access Control to be done.

But there are less conventional means to solve this problem. This takes into account that the only thing we have to be sure of is that the server emitted a Reconfigure-init message, which is not exactly the same that being sure that the server emitted the Reconfigure-init message we received. It means that a client just have to know if a Reconfigure-init message it received causally follows one sent by the legitimate server and haven't been modified. Two cases are possible: In one hand, the Reconfigure-init message comes directly from the server. In the other hand, the Reconfigure-init message has been emitted by a third party, after this one received a legitimate one. We simply need to ensure that the client received a Reconfigure-init message identical to the one sent by the legitimate server.

This can be done with this method:

1 - The server generates randomly a value 'V' and computes a hash 'H[V]' from it. This value will be used to prove the identity of the sender of the next Reconfigure-init message.

2 - The server sends 'H[V]' to clients with configuration parameters in Reply messages.

3 - When clients receive 'H[V]', they save it.

4 - When the server sends to clients a Reconfigure-init message, it encloses in this message a special option TBD, which contains 'V'. It also performs a hash of the entire Reconfigure-init message H[RI], and saves it. This hash will be used to check the integrity of the Reconfigure-init message.

5 - When a client receives a Reconfigure-init message, it first computes a hash 'H' of the value contained in the option TBD. Then, the client compares 'H' with H[V] it obtained from server. If values are equal, it means that the Reconfigure-init has been sent by the server, because, due to the properties of hash functions, it's very hard to find a value that returns 'H' when applied the hash function. Then the client computes a hash of the entire Reconfigure-init message, and inserts it in a TBD2 option. It sends this option in the next Request message to server.

6 - When a server receives a Request messages containing a TBD2, it knows this Request message has been sent by the client after this one received a Reconfigure-init message. We have to consider two cases:

  - The content of the option TBD2 is equal to the hash of the Reconfigure-init message the server sent. This means that the client replied to the Reconfigure-init the server sent, given that the value contained in the TBD2 option is the result of the application of the hash function on the Reconfigure-init message the client received. This way, the integrity of the Reconfigure-init message the server sent has been checked.

  - The content of the option TBD2 is different from the hash of the Reconfigure-init message the server sent. This means that the Reconfigure-init the client replied to is not the one the server sent. A new Reconfigure-init message must be sent to this client using its unicast address.

It works only because we don't have to protect against replay. A server sends a Reconfigure-init message when it wants clients to reconfigure. Once reconfiguration has been done, the key is no longer valid for clients that have done it. Thus, if an attacker gets the value and tries to replay a Reconfigure-init, two cases are possible:

1 - The client which is under attack hadn't received the original Reconfigure-init message. In this case, the attacker is in fact useful, enabling the client to initiate reconfiguration, which the client had to do. If the attacker tries to modify the content of the message, the server will know it by checking integrity of the message.

2 - The client which is under attack received the original message. In this case, the client has initiated his reconfiguration, and the key used by the attacker is no longer valid.

5.4.4. The proposed solution

Of course, we won't propose to use password for authentication, given the fact that it should be possible for an attacker to get it, unless the use of encryption.

Use of DNSSEC or IPsec doesn't seem to be a good idea, because the check of server identity can be done only after the client has been configured, i.e. after the transaction.

In fact, all we have to do is to be able to create dynamic security association between clients and servers.
This can be done using a third party that will be able to authenticate client and server, and to distribute session keys.
A AAA architecture would be a tool of choice.

Here is a scheme showing the communication channels between entities :

```
+--------+   +---------+   +---------+
|        |   |         |   |         |
|  DHCP  |___|  DHCP   |___|   AAA   |
| Client |   |  Server |   | Servers |
|        |   |         |   |         |
+--------+   +---------+   +---------+
```

Where AAA Servers is a AAA System that is able to authenticate client and to communicate with the DHCP server in a secure way.

Here is another scheme, showing static security associations :

```
+--------+   +---------+   +---------+
|        |   |         |   |         |
|  DHCP  |___|   AAA   |___|  DHCP   |
| Client |   | Servers |   |  Server |
|        |   |         |   |         |
+--------+   +---------+   +---------+
```

The AAA servers are able to authenticate and authorize the DHCP client as well as to authenticate the DHCP server. They share security associations with each of them.

This mean that we have a chain of security relationships between the client and the DHCP server we can use to make them communicate together.

The detailed method is:

1- In its DHCP Request, the client will insert a special option TBD1 containing :

   - A replay protection indicator.
   - Credentials that prove the client's identity.
   - A randomly created session key to be shared with the DHCP server.

   This special option will be cyphered with the key the client shares with its AAA server, which can be symmetric or asymmetric.
   Due to the fact that TBD1 option is cyphered, the server can't access the information it contains.
   Client will also insert a second option, TBD2, containing it's unique identifier. This one is formed as follows:

   user@realm

   where user is the client unique network access identifier [6] in its administrative domain, and realm the name of the administrative domain it depends of. This option presents two advantages:

   - It helps AAA servers to send TBD1 option to the appropriate realm.
   - It helps the AAA server responsible of the client to know which client it is dealing with.

   These two options will be piggy-backed in the Request message sent to the server.

2- When the DHCP server receives the Request message, it first sends TBD1 and TBD2 options to its AAA server, which is collocated on the same administrative domain. This one will be in charge of forwarding both options to the AAA server which is responsible of the client.

3- The client's AAA server receives information from the DHCP server's AAA server. If information are checked as being correct, i.e. the client is authenticated, client's AAA server sends back the session key to DHCP server's AAA server, in order for it to forward this session key to the DHCP server.

4- The DHCP server's AAA server receives the session key, and uses its security association with DHCP server to forward the session key to it.

At this time, client and server have a shared session key, which will be used for each message between them.

In Reply message, server adds a TBD3 option, containing :

- The hash of the key used to sign the Reconfigure-init message.

A security association between the client and the server have been created using this method without making important modifications to protocol.

6. Security Considerations

This document addresses numerous security problems which DHCPv6 could be a victim of.

7. References

[1] J. Bound, M. Carney, C. Perkins, R. Droms, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", draft-ietf-dhc-dhcpv6-18.txt, work in progress, March 2001.

[2] N. Asokan, P. Flykt, C. Perkins, T. Eklund, "AAA for IPv6 Network Access", draft-perkins-aaav6-03.txt, work in progress, March 2001.

[3] F. Dupont, M. Laurent-Maknavicius, "AAA for mobile IPv6", draft-dupont-mipv6-aaa-00.txt, work in progress, February 2001.

[4] R. Droms, W. Arbaugh, "Authentication for DHCP Messages", draft-ietf-dhc-authentication-16.txt, work in progress, January 2001.

[5] G. Montenegro, C. Castelluccia, "Statistically Unique and Cryptographically Verifiable Identifiers and Addresses", draft-montenegro-sucv-00.txt, work in progress, April 2001.

[6] B. Aboba, M. Beadles, "The Network Access Identifier", RFC 2486, January 1999.

8. Addresses of contributors

Questions about this memo can be directed to the authors:

Nicolas Prigent
ENST Bretagne
2, rue de la Chataigneraie
BP 78
35512 Cesson-Sevigne Cedex
FRANCE
EMail: Nicolas.Prigent@rennes.enst-bretagne.fr

Jerome Marchand
ENST Bretagne
2, rue de la Chataigneraie
BP 78
35512 Cesson-Sevigne Cedex
FRANCE
EMail: Jerome.Marchand@rennes.enst-bretagne.fr

Francis Dupont
ENST Bretagne
Campus de Rennes
2, rue de la Chataigneraie
BP 78
35512 Cesson-Sevigne Cedex
FRANCE
Fax: +33 2 99 12 70 30
EMail: Francis.Dupont@enst-bretagne.fr

Maryline Laurent-Maknavicius
INT Evry
9, rue Charles Fourier
91011 Evry Cedex
FRANCE
Fax: +33 1 60 76 47 11
EMail: Maryline.Maknavicius@int-evry.fr

Julien Bournelle
INT Evry
9, rue Charles Fourier
91011 Evry Cedex
FRANCE
Fax: +33 1 60 76 47 11
EMail: Julien.Bournelle@int-evry.fr

Bernard Cousin
University of Rennes / IRISA
Campus Universitaire de Beaulieu
35042 RENNES Cedex
Fax: +33 2 99 84 71 71
EMail: Bernard.Cousin@irisa.fr