32603	Deliverable D 6.2.2v2 **Operational procedures for secured management with transition mechanisms (version 2)**	6net

Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/Partner/DS/No./A1**
Contractual Date of Delivery to the CEC:	February 2004
Actual Date of Delivery to the CEC:	May 2004
Title of Deliverable:	Operational procedures for secured management with transition mechanisms (version 2)
Work package contributing to Deliverable:	WP6
Type of Deliverable*:	R - Report
Deliverable Security Class**:	PU - Public
Editors:	Tina Strauf
Contributors:	Giuseppe Di Battista, Bernard Tuy, Lorenzo Colitti, Jerome Durand, Rob Evans, Dimitrios Kalogeras, Georgios Koutepas, Maurizio Patrignani, Pekka Savola, Stig Venaas, Christian Strauf, Tina Strauf, Torsten Kersting, Simon Leinen

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

This document examines the operational security and management issues behind various methods employed when migrating to native IPv6 network interconnection. A detailed description of all mechanisms that can be used to gradually migrate to IPv6, is well beyond the scope of this document. The user is instead referred to the WP2 cookbooks D2.3.3 and D2.2.3 and their respective updates. These documents not only explain every transition mechanism in great detail but also specify how to install and configure them and where specific mechanisms are applicable. In contrast to WP2 this document will exclusively focus on security issues and management of the most important transition tools. Currently known issues, threats and problems are explained and where possible solutions are presented.

Table of content

1 Introduction

IPv6 is the new version of the Internet protocol, promising to overcome the lack of IPv4 addresses as well as quite a few more features, which either where not present in the previous IP version or only usable through complicated means. It is expected that networks in the future will be designed with this new protocol, and that existing networks will be migrated in due course. The transition from IPv4 to IPv6 was never meant to be performed on a flag day. It was rather planned to be introduced smoothly over a long period of time where both protocols would be run in parallel on most networks but most likely not all. Some parts of the network might already be IPv6-only while others might still be IPv4-only for different reasons. This causes additional problems because IPv6-only nodes cannot "talk" to IPv4-only nodes on their own nor vice versa. The protocols run in parallel just like IP and IPX, for example.

To facilitate the migration, quite a few so called "transition mechanisms" where designed along with the new Internet protocol, which make it possible to build different kinds of "bridges" between IPv6 and IPv4 in the network. These mechanisms however are sometimes very complex and often cannot be managed by conventional IPv4 tools. Also, quite a few security issues arise when deploying IPv6 transition mechanisms, which need to be addressed and solved.

During the course of 6NET, different partners have gained a lot of experience on running and operating different transition mechanisms. Different tools and procedures are or were used for management and security. This document builds on this experience and presents solutions for secure management and operation of the most commonly used transition mechanisms. It can however not guarantee to cover all security, management or operational issues nor all mechanisms as not all have yet been extensively used and some are even still in development.

Transition from IPv4 to IPv6 can be divided into three separate scenarios, which make use of different kinds of transition mechanisms and therefore also differ greatly in terms of management and security issues. The scenarios are "Tunnelling", "Dual-stack" and "Translation".

By tunnelling we mean the employment of either IPv6-in-IPv4/IPv4-in-IPv6 tunnels, layer 2 tunnels over which IPv6 can be transported, or any other kind of tunnel where IPv6 (or possibly IPv4) isn't natively transmitted/routed through a network but rather crosses an intermediate network that does not understand IPv6/IPv4.

Dual-stack is most likely the next step in the evolution of all networks in general and means the deployment of both IPv4 and IPv6 in parallel on the whole network (all hosts and routers). The deployment of a dual-stack networks also creates management and security problems that need to be addressed. As far as these problems apply exclusively to dual-stack networks and the migration from IPv4 to a dual-stack network, we will cover them in this document. Some problems however are not exclusively related to transition but rather to the management and security of IPv6 networks in general. Those issues, problems and solutions are described in WP3 or other work packages.

"Translation" refers to a special kind of mechanisms most commonly employed when IPv6-only hosts or networks need to still be provided with IPv4 connectivity (and vice-versa) or some kiind. It involves the actual translation of IPv4 to IPv6 (and vice versa) either at the protocol level (NAT-PT), at the application level (ALGs) or the transport protocol level (TRT). 6Net partners still have little experience with this last scenario and its transition mechanisms as IPv6-only hosts and networks are very rare and if present mostly experimental lab setups. Problems and solutions

presented in this document result mostly from early tests or pilot deployment and are most likely not complete. It is expected that any further issues that come up during the course of 6NET will be included in updates of WP2 deliverables.

The document is intended to both readers who prefer to read it as a whole as well as for readers looking only for information on specific transition mechanisms. Therefore aside from some pointers to other documents and a hierarchy in terms of specialization of certain mechanisms individual sections are mostly independent and because of that might include some information that was mentioned previously in paragraphs on related mechanisms.

2 Tunnelling

Generally any form of tunnelling poses a security threat to a network. If set up properly tunnels can effectively circumvent and undermine any security features present to guard the network like access control lists and firewalls. In a way they drill a hole through them since these security measures only "see" the outer layer of the packets, which might be well within the permitted parameters but have nothing at all to do with the contents/protocol/traffic inside. So if this traffic reaches a tunnel end-point inside the guarded network it is decapsulated and from there can potentially be very harmful since within a network itself, defence levels are usually much lower. Tunnels used for IPv6 deployment are no exception.

During the migration from IPv4 to IPv6 three different kinds of tunnels maybe be used: IPv6-in-IPv4, IP(v4)-in-IPv6 or other layer tunnels.

In terms of general management of tunnels there is to date no special MIB (Management Information Base) defined for use with for example SNMP for tunnelling and the specifics of tunnelled traffic or tunnel interfaces. However an Internet Draft [TunnelMIB] was released in January 2004 by D. Thaler covering this issue in the context of the IPv6 working group of the IETF.

2.1 General mangement issue with tunnels

Often the tunnel MTU is not specifically configured on the end-points when a tunnel is set up. The system then chooses a default MTU. In Cisco's IOS, for example, the default MTU will be derived from the interface MTZ of the interface towards the other end-point (by substracting the encapsulation overhead). Even if this happens to result in the same MTU at both end-points at the time the tunnel is set up, the MTUs may diverge later (e.g. if one of the end-points starts to support "jumbo frames on the egress interface, or routing changes cause the egress interface to move to one with a different MTU.

The resulting situation is a logical IP(v6) subnet where not all interfaces have the same MTU. This violates a fundamental assumption in IP networking and causes connectivity problems.

However, the symptoms are such that this situation is often hard to diagnose. In the direction from the end-point with the smaller MTU towards the receiver with a larger MTU interface, no problems will show but the other way around packets with larger size than the receiver's MTU will usually be ignored and counted as errors at the receiver. Typical tests with ping or traceroute will not show any problems because these tools use small packets. Protocols like BGP may work over the tunnel most of the time but may come to the point where the side with larger MTU must send a large amount of data and uses larger packets than the other side can take.

Unfortunately some devices (seen on Cisco routers under IOS) ignore attempts to configure a 1480-byte MTU on a tunnel towards a 1500-byte MTU interface, because this is already the default. In these cases we recommend to fix the devices so that they still accept the manually configures MTU. This will guard against problems arising when the "default MTU" changes.

MTU incompatibilities are detected by some routing protocols such as OSPFv3, which is very useful for debugging. However, such protocols usually aren't used over inter-domain tunnels, where problems are most likely to occur. From this point of view it would seem useful if BGP-4 had an option to advertise link MTU in the single-hop case. Alternatively, MTU validation could be made part of a link liveliness detection protocol such as BFD (bidirectional forwarding detection).

Note that path MTU discovery ([PMTUD1], [PMTUD2]) would make it possible for the end-points to discover the largest MTU that can be supported by the underlying network without fragmentation, but this doesn't solve the inconsistent MTU problem, because there is no guarantee that the path MTUs in both directions end up being the same. Also it isn't always implemented for tunnel interfaces (see IPv6-in-IPv4 tunnels).

2.2 IPv6-in-IPv4 tunnels

This category of tunnels covers the most basic and well-known transition mechanisms *Manually Configured Tunnels*, *6to4* and *ISATAP* (please refer to D2.3.3 section 3.2 for a detailed description of these mechanisms). We will cover any specific operational, management and security issues of these mechanisms below. All of these mechanisms however have quite a few issues in common since they all encapsulate IPv6 packets in IPv4 packets.

2.2.1 General security issues with IPv6-in-IPv4 tunnels

Security Issues with IPv6-in-IPv4 tunnels in general lie mainly in the above-mentioned problem of these tunnels circumventing and subverting security measures present for IPv4, specifically normal (IPv4-based/IPv6 unaware) firewalls on which IPv4-encapsulated IPv6 traffic only registers as IP protocol type 41 (IPv6). If one wants to use IPv6-in-IPv4 tunnels, this protocol type has to be permitted in the firewall's rules. This will effectively open a hole in the carefully configured and maintained security of the site as the traffic is let through without further inspection. This IPv6 traffic can be anything and, if the tunnel end-point also acts as an IPv6 router and forwards IPv6 traffic inside the site's IPv6 network, upon reaching the tunnel-end-point inside the site could go anywhere undetected after decapsulation (though the potential damage is in most cases limited to the broadcast domains that the tunnel end-point resides in).

An example:

A site filters all incoming and outgoing (IPv4) http traffic unless it originates from or goes to a specific host (proxy). This protects otherwise unprotected web servers inside the network (i.e web interfaces for configuration of network components) against attacks from the outside. The other way around this could (if the proxy were properly configured) prevent access to certain websites from inside the site. If the site now uses any IPv6-in-IPv4 tunnel mechanism to get (global) IPv6-connectivity, this tunnel most likely needs to pass the firewall to an end-point on the inside, which then becomes the site's IPv6 border router. Any (IPv6) http traffic may then travel anywhere from and to IPv6 nodes in the network, which again leaves IPv6 enabled (and connected) network nodes with IPv6 enabled web interfaces vulnerable to attacks from the outside, if they are executed via IPv6.

The best way to remedy this problem, is installing an IPv6 capable firewall on the tunnel end-point that examines and properly filters the incoming IPv6 traffic after it has been decapsulated from the IPv4 wrapping. This firewall could mirror any rules present for IPv4 at the site's border for IPv6. This is the only way to let only specific IPv6 traffic in and out of the site through the tunnel.

If one only wants to filter specific traffic, one could theoretically employ bitwise filtering and look for specific bit patterns in the payload of the packets. This might make it possible to filter on at least

IPv6 source and destination addresses; but this is very tiresome, prone to mistakes and not at all scalable. We do not assume that anyone would try to do this but want to state specifically that we recommended not to use this method nor any other kind of packet filtering that doesn't work on the decapsulated IPv6 packets.

Even when using a proper IPv6 firewall on the decapsulated packets, one must be careful when setting up a tunnel because any host could potentially spoof the other end-point's IPv4 address and send IPv6-in-IPv4 encapsulated packets. The local tunnel end-point will not know that the source of these packets is not the real remote tunnel end-point and decapsulate the IPv6 traffic which can then (if not further inspected/filtered) freely enter the local IPv6 network. The attacker does not even need to know the IPv6 addresses being used on this network as it can send an ICMPv6 packet to all hosts on the tunnel link using its own IPv6 global address and retrieve the IPv6 addresses used in the network. This problem is not easily solved and in its essence is not really specific to IPv6-in-IPv4 tunnelling because it is based on IPv4 address spoofing. The best way to ward against this is doing strict RPF checking at the site's edge but even then, one cannot be completely sure. One more or less relies on other sites filtering packets with IPv4 source addresses that are not from their site at their edge preventing them from being sent out to the Internet. Concerning IPv6 the local tunnel end-point can add some additional security by dropping packets that turn out to be link-local after decapsulation. This is not always possible though, since some services or protocols rely on the use of link-local (unicast or multicast) addresses. These protocols (e.g. PIM, RIPng) can potentially be attacked by anyone. If encryption or authentication facilities are available for these services they should be used. For the other services, no real filtering can be done. We have already seen the example of a broken IPv6 multicast network because an attacker was sending bad PIM announcements, causing a bad PIM topology on the tunnel end-point. Even if a protocol run over the tunnel is not using link-local addresses (like BGP) the implementation of authentication/encryption is advised.

2.2.2 General management issues with IPv6-in-IPv4 tunnels

Management of IPv6-in-IPv4 tunnels depends more on the specific mechanism used to set up the tunnel(s). However, concerning monitoring, these tunnels have in common, that after configuration they behave like a point-to-point link and will only appear as one hop concerning pings and traceroutes. Unfortunately this "link-like" behaviour does not extend to features like notifications when a link goes down or up, as one can see them with real physical links. One will only recognize a tunnel going down by the fact that packets can no longer be transmitted but debugging and finding the reason for the failure is much harder, and needs to be performed by hand on the "IPv4 way" the tunnel takes, which of course may vary without anybody noticing. Therefore both for maintenance as well as of course performance reasons IPv6-in-IPv4 tunnels should only be set up over topologically short IPv4 distances

Since IPv6-in-IPv4 tunnels are purely IP they cannot be used for routing protocols like IS-IS. They can however be used for BGP without problems.

The MTU for a tunnel link is less than the path MTU for the tunnel since an IPv4 header must be added to all packets going through the tunnel. In general this might lead to undesired fragmentation effects. For IPv6 the use of path MTU discovery makes this a much smaller problem, but it is not always implemented (for tunnels). Some IPv6 implementations instead just always use the minimal IPv6 MTU without checking before. The minimal MTU is 1280 for IPv6-in-IPv4 encapsulated packets. This will avoid the problem of fragmentation in nearly all cases, since the IPv4 path MTU

is often at least 1500 at the cost of adding unnecessary overhead when a larger MTU would be possible.

There might be IPv6 implementations that do not allow the same management operations for tunnel interfaces as for physical interfaces. We have seen at least one implementation that did not allow tcpdump on tunnel interfaces. There are probably other examples.

Purely hardware based routers will need specialised hardware to be able to encapsulate and decapsulate packets so that they can be used as tunnel end-points. Routers that do some operations in hardware and some in software will probably be able to handle this. One should be aware, though, that the software processing power might be limited, and the CPU used for the processing is probably also used for other tasks.

2.2.3 Manually configured IPv6-in-IPv4 tunnels

Other than the above-mentioned general security and management issues for IPv6-in-IPv4 tunnels there are no specific problems with manually configured tunnels. Out of all transition mechanisms building upon these kinds of tunnels, manually configured tunnels however are considered to be the most stable and operationally secure due to the high level of control the administrator has over them. On the other hand, they do require the most work upon setup and and both IPv4 addresses are hardcoded into the configuration which makes it impossible to use these kinds of tunnels between end-points with dynamic IPv4 addresses (i.e. over dial-in lines), at least without some kind of extra automatic setup procedure which we cover in a separate section on Tunnel Brokers.

We have already seen one implementation of tunnels that did not check if the source address of the IPv4 packet was the one configured by the administrator. Any host could potentially send IPv6 packets through the tunnel. It is always recommended to at least check, if the IPv4 source address of an IPv6-in-IPv4 packet is the IPv4 address of the other end-point, even if this doesn't guard against spoofed packets.

2.2.4 6to4

Special issues with 6to4 mainly relate to the way IPv6-in-IPv4 tunnels are set up automatically and the security issues arising when somebody operates a (public) 6to4 relay. For the following section a 6to4 host or router is a host with just a 6to4 pseudo interface. This host might or might not have native IPv6 connectivity. Similarly it might or might not announce its 6to4 prefix to a subnet and thereby act as IPv6 access/default router for this subnet. A 6to4 relay is a dual-stack host with a 6to4 pseudo interface, which forwards packets between the 6to4-domain (2002::/16) and the rest of the IPv6 Internet. A 6to4 relay is also a 6to4 host.

In terms of management and security a network for which a 6to4 host acts as a border router is not affected by the fact that the border router uses 6to4 to provide outside connectivity (either globally or just within the 6to4 domain) aside from the fact that this network's IPv6 connectivity of course depends on the 6to4-connectivity of the 6to4 host.

2.2.4.1 Security issues with 6to4

6to4 hosts or routers accept and decapsulate IPv4 traffic from anywhere. Constraints on the embedded IPv6 packets or where IPv4 traffic is automatically tunnelled to are minimal. Two kinds of attacks are therefore possible:

1. The 6to4 pseudo-interface can be attacked remotely with tunnelled link-local packets. If the interface is not insulated from the host's other interfaces (which is rarely the case in practice) attacks like this could result in a corrupted neighbour cache for the whole system.

   This threat can be averted by adding an access-list to the pseudo-interface to filter out bad tunnelled packets:

   ```
   a. allow from 2002::/16 to 2000::/3
   b. deny everything else
   ```

2. As stated above 6to4 hosts decapsulate and possibly forward any traffic coming in to the pseudo interface. They cannot distinguish between malicious IPv4-encapsulated IPv6 traffic and valid traffic coming from 6to4 relays. This "functionality" can be used both for unidirectional source address spoofing and the reflection of Denial-of-Service attacks against native IPv6 nodes. The latter is not a very big problem since the traffic cannot be multiplied and might even be adversely affected by going through bottlenecks like 6to4 relays, decapsulation and encapsulation. The only problem here is, that an attacker can more easily cover his tracks. The unidirectional source address spoofing of course also exists without 6to4 but becomes harder because the attacker needs to know a valid (existing) IPv6 address. This is a lot easier with 6to4 present because here the attacker can just take any non-6to4-address.

   Attacks like these two can also only be remedied by employing sufficient filters. For example all IPv6 nodes inside the site can be guarded from attacks, if the 6to4 pseudo interface does not accept traffic from the IPv6 prefix(es) used inside the site. This also means that the site's own 6to4-prefix should be filtered on input.

Additional security issues with 6to4 relays are due to the fact that 6to4 relays by nature have a native IPv6 connection in addition to IPv4 and relay rather freely between the two. Native IPv6 nodes anywhere can use the relay as a means to obscure their identity when attacking (possibly even IPv4 nodes). Attackers from IPv6 can attack IPv4 hosts with tunnelled packets sending spoofed 6to4-packets via a relay to the IPv4 hosts. The relay can obscure identity, if it relays any packeta not checking, if the 6to4 address actually matches the IPv4 host, the packet comes from. Note that for relays it is assumed that it is at least configured in a way as to not relay between different 6to4 addresses (except of course from or to other known 6to4 relays), thereby facilitating IPv4 to IPv4 attacks.

1. A 6to4 relay can be used for locally directed (IPv4) broadcast attacks. For example if the relay has an interface with address w.x.y.z/24 an attacker could send packets with a 6to4-address that translates into the address w.x.y.255. This is even possible to remote locations if "no ip directed broadcast" is not configured.

This problem however is easily remedied by another entry in the access list, which prevent's packets with destination similar to the above 6to4-address from getting in.

2. The issue mentioned above is actually only a special case of the general problem of 6to4 relays becoming a part of DoS attacks against IPv4 nodes which might be totally unaware of 6to4 but get hit by encapsulated packets nevertheless. If the attack further is executed with a spoofed source address (which is easily possible as stated above) the source of the attacks cannot be traced. A 6to4 relay can also be used for address spoofing and therefore anonymization of attacks coming from native IPv6 hosts

Generally a 6to4 relay can be reasonably well protected if the validity of source or destination 6to4 addresses is always checked. That is, it should be checked if the enclosed IPv4 address is a valid global unicast IPv4 address. It could even be restricted to only accepting and forwarding 6to4 encapsulated traffic where the 6to4 destination or source address matches the actual IPv4 address the packets come in from or go to. As with the general rule about no forwarding between 6to4 addresses however, exceptions must be made for traffic coming from or going to known other 6to4 relays.

2.2.4.2 Management issues with 6to4

However well protected a 6to4 relay may be, the traffic going through should always be monitored, especially if the relay is configured with the well-known IPv4 anycast address for public 6to4 relays.

Other than monitoring no particular mangement is required for 6to4 since it was specifically designed for ease of use and low maintenance.

2.2.5 ISATAP

ISATAP is another automatic tunnelling mechanism based on the automatic creation of IPv6-in-IPv4 tunnels. As such – from a security point of view – it should not be used, if manually configured IPv6-in-IPv4 tunnels are an option. However, since ISATAP is specifically meant to be used only within a site and if correspondingly protected, it is a reasonably secure and low maintenance mechanism, to provide isolated dual-stack hosts with IPv6 connectivity to the site's main IPv6 network and thereby global IPv6 connectivity.

2.2.5.1 Security issues with ISATAP Clients and Servers

An ISATAP server or router should be protected in such a way as to only permit incoming tunnels from the hosts inside the site. This can be accomplished with simple IPv4 firewall rules. Additionally the site's normal IPv4 border router should permit incoming and outgoing protocol 41 (IPv4 encapsulated IPv6 traffic) only for source and destination addresses belonging to known tunnels. This is not only to protect the ISATAP servers but all ISATAP clients in the site as well, as all clients connected to the same ISATAP server are essentially on the same (IPv6) link and cannot be easily protected from one another.

If the list of ISATAP servers is in any way made automatically available via DNS, DHCP or other means it should be very well maintained.

Since ISATAP clients and servers perform actual neighbour discovery when first starting to communicate with the only difference being that the ISATAP routers do not send unsolicited router advertisements, the same procedures to secure neighbour discovery should be taken as in any native IPv6 network.

2.2.5.2 Management issues with ISATAP

Monitoring traffic between the ISATAP hosts at a site is difficult. All hosts using the same ISATAP router are on the same virtual link, so the packets do not really pass through any other routers (of course the packets might pass through IPv4 routers on the layer below but there they are hardly distinguishable from the normal IPv4 traffic). Monitoring of non-link-local traffic can thus really only be done on the ISATAP router but itself. Note that ISATAP clients within the site can send packets to each other directly using IPv6-in-IPv4 encapsulation and their link-local ISATAP addresses. This traffic does not go through the ISATAP server and can only be monitored at the sending and receiving nodes which is hardly feasible for all hosts of the site.

2.3 GRE Tunnels

The use of IPv4 GRE (Generic Route Encapsulation) tunnels provides another means to transport IPv6 over an IPv4-only network. In most cases they are used because unlike IPv6-in-IPv4 tunnels where IPv6 is directly encapsulated in IPv4 datagrams GRE can be used for the Intermediate System-to-Intermediate System (IS-IS) routing protocol.

2.3.1 Security issues with GRE tunnels

If GRE tunnels are to go through an IPv4 firewall this firewall has to be opened for IP protocol type 47 for IPv4 datagrams coming from or going to the remote tunnel end-point.

GRE tunnel end-points are authenticated by a simple key that is transmitted during the setup of the tunnel. Since the key is transmitted in clear text format this doesn't really add much security and the key is also not used for encryption of any kind.

2.3.2 Management of GRE tunnels

The broader functionality of GRE tunnels comes at the cost of an even shorter MTU, since the GRE header also has to be included in each packet. Other than that, GRE tunnels can be managed like IPv6-in-IPv4 tunnels or point-to-point links respectvely.

2.4 OpenVPN tunnels

OpenVPN is (as the name indicates) a VPN solution. Licensed under the GPL it creates cross-platform (layer 2) point-to-point or ethernet-bridge tunnels over which IPv6 can easily be transported. The software multiplexes IPv6 in IPv4 UDP packets using functionality provided by the OpenSSL library, which may optionally be encrypted. As a VPN solution one has of course to

regard one end of the tunnel as the "server" end and one as the client but both ends use the same software. The tunnel end at the site that provides IPv6 connectivity acts as the server. For more information on how OpenVPN works, please refer to the project's homepage at :

http://openvpn.sourcefourge.net

2.4.1 Security issues with OpenVPN tunnels

In terms of security OpenVPN has the great advantage of providing authenticated and optionally even encrypted tunnels. It is based on OpenSSL for certification and either uses static pre-shared keys or TLS for dynamic key exchange. The use of X.509 certificates can be regarded as very secure. It can only be compromised, if the secret key is not kept safe.

The certificates are not bound to specific hosts. They can be used anywhere between any two hosts. So an owner of a certificate could put both public and private key on his laptop and with that set up an authenticated tunnel from anywhere where he has IPv4 connectivity. This, of course, is the desired functionality for any Virtual Private Network solution but in comparison to the usual IPv6-in-IPv4 tunnels this has quite a few advantages for the deployment of IPv6 on for example dial-in lines where users not usually have static IPv4 addresses. It provides the user with much more flexibility at the cost of security relying solely on the fact that the user keeps his keys safe and *only* uses them for himself.

2.4.2 Management of OpenVPN tunnels

OpenVPN tunnels are very robust and work even on rather unstable/unreliable IPv4 connections between both end-points. They are known to survive even ISDN or DSL reconnects where the client comes back with a different IPv4 address. In this case just a new TLS handshake is performed to authenticate both sides and the tunnel is back online.

In and of itself the mechanism is not automated but it is an ideal basis for setting up a tunnel broker :

- The use of a CA enables a centralized management of acces authorization and trust.
- Failure of the tunnel broker's hardware or the IPv4 link between tunnel broker client and server does not impose administrative work other than fixing hardware or link. The service continues seamlessly after the IPv4 link between client and server is re-established. The FQDN is used to identify a server and hence DNS entries may be changed to redirect tunnel broker clients to a working server in the case of a failure.
- The persistence of the IPv6 link is very good because of mechanisms inherent to the OpenVPN software.
- OpenVPN traverses most NATs without the need of additional configuration. If the NAT does not support this traversal, fowarding of a single UDP port to the OpenVPN client suffices to establish connectivity.

2.5 IPSec tunnels

Like VPN IPsec tunnels are not technically an IPv6 migration mechanism but they can be used to transport IPv6. Due to their authentication mechanism and encryption they provide a more secure way of setting up a tunnel than simple IPv6-in-IPv4 tunnels. Note that using encryption can have a negative effect on performance of the tunnel.

2.6 Tunnel Broker

Tunnel brokers can be a nice way to offer IPv6 connectivity to individuals and perhaps small organisations if there is a large number of potential individuals or organisations. The idea is mainly that those that want connectivity can use a web interface or other means to request and configure a tunnel, doing some form of authentication and supplying their information needed for the setup (which in most cases is their IPv4 address etc). The tunnel at the provider end will be configured immediately and the broker might also supply the customer with a script to configure the user end. Without a broker, each individual would have to contact the provider independently, and an administrator would have to configure each tunnel. So running a tunnel broker is a way of avoiding some of the administrative tasks, and offer instant service at the same time. Additionally a tunnel broker is the only way to connect customers with dynamic IPv4 addresses or otherwise chaning end-points via tunnels that usually have to be configured manually. As with all forms of automatic configuration mechanisms and automatic user/server interaction the use of tunnel brokers does raise some security and management issues, especially since tunnel brokers are often handmade and involve a lot of different techniques. These problems as we present them below however are not specific to the tunnels used. Any form of tunnel that can transport IPv6 can be used to set up a tunnel broker though some tunnels are better than others because they might already offer some features that would otherwise have to be added by hand like authentication, encryption or maintaining connectivity via unstable lines.

If one wants to implement a tunnel broker, it is a good idea to refer to the Internet Drafts concerning the “Tunnel Setup Protocol” ([TSP1], [TSP2], [TSP3]), which standardizes all of the below mentioned issues.

2.6.1 Authentication

In general one would like to have some idea of who is using the different tunnels, and also make sure that only the owner of the tunnel can change the configuration. If a broker is run by an organisation with directory type structures installed (e.g. a university for their students) one may be able to use usernames and passwords or other means of authentication that the organisation already uses for other services. If there is no prior relationship with the user, one typical approach is to have people register with name, e-mail address etc. and then receive an auto-generated password through e-mail. This way however there is uncertainty about the real identity of the final users. The only certainty with this approach is, that the e-mail address used for registration is confirmed as operating and that the person receiving the password is one of the receivers of e-mails to this address.

An alternative approach could be to abide by the RFC3129 [KIN]. In this context we accept a *Kerberised Internet Negotiation of Keys*. Clients authenticate to a centralized server, the Key Distribution Centre, which in turn issues tickets that servers can decrypt thus making sure that the

client is in fact who it/he claims to be. One of the elements of a Kerberos ticket is a session key generated by the KDC, which may be used by the client and server to share a secret. Kerberos also allows for both symmetric key authentication, as well as certificate based public key authentication (PKinit).

As described above OpenVPN, too, provides a way for authentication but only after an initial trust has been established.

2.6.2 Enabling/disabling tunnels

A user might be provided with the possibility to enable and disable the tunnel by using a web interface or perhaps a script using HTTP or other protocols to communicate with the broker. If the user is always connected with a fixed IP address, all should be well. As long as the user leaves the tunnel enabled, it should forward packets. However, if a customer uses a connection method that does not guarantee a static IPv4 address (e.g. dial-up) and somehow gets disconnected, it would be prudent to take the tunnel down automatically to prevent unnecessary use of resources. In terms of security, this would assure that the session is not (intentionally or unintentionally) "hijacked" by someone else who connects with the same address, thus receiving traffic intended for the original user. One possibility would be to automatically take down the tunnel if no packets are received from the user for some time, especially if packets are flowing in the other direction. This could be combined with some utility at the user's end that sends some sort of keep-alive messages. The type of packets does not really matter as long as something is sent. One could also have some TCP session between user and broker for controlling the tunnel, and have keep-alive messages sent over the TCP session. If no data arrives or the session is reset the tunnel can be taken down. The downside to all this is that a solution is harder to deploy if it would require some specialised software for the user's operating systems.

2.6.3 Guarding against misuse and statistics

When running a tunnel broker, one should also apply filtering of some IPv6 packets. The broker (or more correctly, the router that is the provider's end-point of the brokered tunnels) should drop any packets received on a tunnel interface, that have a source address not belonging to the remote side (no route for that address in that interface). And it should also not send packets into a tunnel that has a source address belonging to the remote side. The first kind of filtering may be problematic if the remote end is multihomed or if the IP address of the customer is not part of the configuration of the tunnel. In these cases the (current) IP address of the customer could nevertheless be registered upon setup of the tunnel and used for the purpose of filtering while the tunnel is running.

When running a tunnel broker service at least some statistics on bandwidth usage per tunnel might be useful. If somebody offers a broker to random Internet users, people should not be very surprised if they encounter DOS attacks. It is good if similar attacks and other problems can be detected, and such traffic blocked if necessary.

2.6.3.1 Tunnel Broker based on IPv6-in-IPv4 tunnels

If IPv6-in-IPv4 tunnels are used to set up a tunnel broker there should be some consideration for monitoring the tunnels' usage. Once set up, the tunnels are no different from manually configured

tunnels and therefore the same management and security issues apply as stated above. As for stability of the connection and the configuration of the tunnel itself: If the IPv4 connection breaks, so does IPv6 but neither user nor provider will make a notice of that unless traffic is sent and doesn't reach the other side. Any management tool or script based on ICMPv6 could be used to monitor the state of IPv6 connectivity.

If on the other hand the IPv4 connection comes back up, no reconfiguration is necessary to establish IPv6 connectivity again, provided of course the IPv4 addresses stayed the same. If that is not the case and the tunnel was not unconfigured on for example the broker's side, the traffic could go to another host being allocated the same address, released previously by the host who asked for the tunnel setup.

2.7 DSTM

DSTM is a mechanism offering IPv4 connectivity/services to dual-stack hosts within an IPv6-only network. For a detailed description of the mechanism please refer to D2.3.3 or a newer version of this cookbook..

2.7.1 Security Issues with DSTM

All of the equipment inside the IPv6 network where DSTM service is available must allow protocol 4 (IPv4) to allow for IPv4-in-IPv6 tunnelling. However some filtering has to be done on the server to prevent unauthorized access as well as providing authorized clients with a secure service. Therfore the following filters should be deployed on the DSTM server:

```
allow ipv6 tunnel_request from allowed_hosts to DSTM_server
allow ipv6 tunnel_request from DSTM_server to TEP
allow ipv6 tunnel_reply from DSTM_server to allowed_hosts
allow ipv6 tunnel_delete from DSTM_server to allowed hosts
allow ipv6 tunnel_delete from DSTM_server to TEP
deny ipv6 from any to any
```

Currently the implementations of DSTM developed and tested within 6NET specifically the one developed at ENST in France does not offer any means of authentication. Future implementations that plan to use the Tunnel Setup Protocol (TSP) [TSP1][TSP2] and DHCPv6 [DCHCPv6] should support authentication.

2.7.1.1 DTI problems with ENST's DSTM implementation

The current implementation of DSTM provided by ENST requires that both DSTM server and TEP are on the same machine in order to have tunnels on the TEP coherent with the allocations made by the server. The implementation is mostly based on RPC [RPC]. At this time, there is no implementation of a communication protocol between the server and the TEP for tunnel setup, even if some ideas are foreseen (TSP and DHCPv6).

If the TEP is not physically located on the machine on which the DSTM server is installed, then it uses Dynamic Tunnel Interfaces (DTI) for tunnel setup. The mechanism is simple: when the TEP receives an IPv6 packet with protocol number 4 (meaning the data carried is an IPv4 packet), then it automatically creates an IPv4 in IPv6 tunnel using the information carried in the packet (IPv6 and IPv4 source and destination address). It is important to notice that there is no way in this case to check that the tunnels setup on the TEP corresponds to the allocations made by the DSTM server. The following case can happen: Host (A) can be allocated an IPv4 address and communicate with the IPv4 world using the IPv4 over IPv6 tunnel created using the DTI of the TEP. If another host (B) sends an IPv4 in IPv6 packet, using the same IPv4 source address as A, then the tunnel between the TEP and A will be destroyed and replaced by a tunnel between the TEP and B. This can be seen as a spoofing attack and is a reason why this approach is now obsolete.

2.7.2 Management Issues with DSTM

Once the DSTM mechanism is up and running, only IPv6 configuration of the network is necessary. So DSTM is really a way of providing some kind of dual-stack functionality without the added overhead of actually running and maintaining the two protocols everywhere in the network. That makes network monitoring much easier because the network administrator does not have to care about IPv4 anymore.

The DSTM service performance does not depend on the size of the DSTM network but rather on the number of people using DSTM at the same time. This means that DSTM can be made available for a large domain with only one server provided the actual amount of IPv4 connections is low compared to IPv6.

For monitoring a DSTM enabled network, the system administrator should monitor the tunnels created on the TEP and the allocations made by the DSTM server. The system should be able to check that the tunnels correspond to the allocations performed by the DSTM server and send alert messages in case an error occurs.

2.8 TSP (Tunnel Setup Protocol)

TSP [TSP1] is not a tunnelling mechanism but rather a means to facilitate the automatic setup of tunnels of any kind. In that way it is a management facility itself, which can be used for example to implement tunnel brokers [TSP3] or in the context of DSTM [TSP2] as a standardized way of exchanging the necessary tunnel information between the two hosts (see Figure 1). TSP messages are exchanged on TCP port 4343. The protocol makes it possible to authenticate the message sender (using MD5 as an example). In terms of security TSP offers the possibility to authenticate the message sender (using for example MD5 checksums). Both actual security and management issues with TSP depend mostly on the implementation used.

2.9 Detecting the presence of IPv6 in IPv4 tunnels

Networks containing IPv6-in-IPv4 tunnels are difficult to manage, because the tunnels are transparent to the rest of the network. In several scenarios, however, the ability to detect tunnels and determine their end-points would be useful.

One example is troubleshooting: if a link in the tunnel's path fails, packets sent through the tunnel are lost, and an IPv6 traceroute will not reveal the source of the problem. In this case, the ability to determine that the failed link is in a tunnel, perhaps also performing an IPv4 traceroute between the tunnel end-points, could help identify the faulty link and take appropriate action.

Troubleshooting is not the only scenario. For example, in some networks tunnels could be used as backup paths in case of problems with the main (native) links. In this case, it would be useful to know if a given destination is reached via a native link or via a tunnel, which may have lower performance. Or, a site or national network could use tunnels as an interim measure until native IPv6 infrastructure is in place; in such a network, a tool capable of detecting tunnels could determine how much still has to be done to complete the migration to native IPv6. Yet another example concerns the management of dynamically generated tunnels, such as tunnels set up by a tunnel broker: It might be useful to determine the IPv4 address of a tunnel end-point without having to query the tunnel broker application.

There are various methods for detecting the presence of a tunnel and/or determining the IPv4 addresses of the tunnel end-points. Some of these suggest the presence of a tunnel, some suggest the possible IPv4 addresses of tunnel end-points, and some do both. Examples are:

- IPv6 address inspection. For tunnels that embed the IPv4 address of the end-point in the IPv6 address, such as 6to4 or ISATAP, inspection of the IPv6 addresses used may help deduce the presence of a tunnel and suggest the IPv4 address of one of the end-points.
- Path MTU discovery. If path MTU decreases by 20 bytes between two consecutive nodes on a path, the link between the two nodes may be a tunnel. Path MTU discovery does not provide information on the end-points, and in the case of multiple tunnels in a path, it detects only the first one.
- DNS lookups. If the DNS is appropriately configured, DNS mappings may provide information about tunnel end-points. For example, if the DNS name of a tunnel interface maps to both an IPv6 address and an IPv4 address, the end-point of the tunnel may be the IPv4 address.
- SNMP queries. The IP MIB and the tunnel MIB [TunnelMIB] provide information on the presence of tunnels and their end-points. The information gained through SNMP is accurate in most cases, but SNMP access is usually only available to network administrators.

Some of these methods yield accurate information, while others provide only hints. Also, some are always available, while others require administrative access to the network. Taking into account the fact that not all the methods provide the same information, a tool that could employ all of them, merging the results from each to form as accurate a hypothesis as possible, would be desirable. Such a tool would also be more efficient than manual intervention of an operator.

3 Dual-stack

Dual-stack is the conceptually easiest and for quite some time to come the best way of deploying IPv6. The drawback to this scenario is the fact that it involves the maintenance of two separate IP infrastructures including management and security.

3.1 Security considerations for dual-stack networks or hosts

The most important paradigm for security in dual-stack networks or on stand-alone dual-stack hosts is that (if this network or host is also provided with global IPv6 connectivity) security for every IPv6 host must mirror exactly the security provisions in place for IPv4. Every firewall rule and every access list that is restricting access to a host needs to be "translated" into corresponding rules and access lists for IPv6. This is not always easy, especially if the network topology is not the same for IPv6 and IPv4. In that case access lists and firewall rule sets cannot be mirrored at all but need to be composed in such a way that they culminate in the exact same level of security for IPv6 for every host as for IPv4.

A special case is, when there's not even global IPv4 connectivity in a network, because that network sits behind a NAT and is addressed with private addresses. For IPv6 on the other hand all hosts could be addressed with globally unique (and reachable/routed) addresses, if connectivity is for example provided through a tunnel. In this case security for IPv6 needs to be designed from scratch although present firewall rules for the NAT itself can provide a basis, if they are translated to corresponding IPv6 rules.

3.2 Management (and performance) issues with dual-stack networks

An important aspect of dual-stack deployment is performance. Dual-stack hosts are configured to always prefer IPv6 when a hostname resolves into both an A (IPv4 address) and an AAAA (IPv6 address) record. The deployment of dual-stack services (e.g. FTP mirror) with different performance for IPv4 and IPv6 must be avoided because the IP layer does not remain transparent. We have seen the deployment of a dual-stack FTP mirror with a poor IPv6 performance, causing all the people used to upgrade their applications on this mirror having deployed IPv6 FTP clients to get a 80kBps bandwidth instead of 4Mbps for their downloads. This issue can of course only be controlled for services one deploys oneself. For remote services the only thing one can do is to keep the reason for these problems in mind (and to educate unknown users accordingly). It is important that people know that this does not happen because IPv6 is slower in and of itself.

One further management issue in deploying an IPv6/IPv4 dual-stack network lies in configuring both internal and external routing for both protocols. If one has for example used OSPFv2 for intra site routing before, adding IPv6 to the Layer 3 network one will either make the transition to OSPFv3 or IS-IS necessary or one will at least be forced to run one of these IGPs in addition to OSPFv2.

4 Transition Mechanisms Using Translation

This section covers management and security issues concerning migration techniques that provide a means of communication between actual IPv4-only and IPv6-only hosts through some sort of

translation. This translation can take place at either the transport, network or application layer of the communication and usually requires a server in between the two network nodes taking part in the communication, that translates/relays the traffic. Based on this functionality both security and performance issues are of great importance when using these mechanisms.

4.1 NAT-PT/NAPT-PT

4.1.1 End-to-End security

As noted in RFC2766, NAT-PT and end-to-end security do not work together. When an IPv6-only node (X) initiates communication to IPv4-only node Y, the packets from X have certain IPv6 source and destination addresses which are both used in IPsec (AH or ESP) and TCP/UDP/ICMP checksum computations. Since NAT-PT translates the IPv6 address of X into an IPv4 address that has no relationship to X's IPv6 address, there is no way for recipient Y to determine X's IPv6 address and in that way verify the checksums.

4.1.2 Prefix Aassignment

RFC2766 does not explain how the IPv6 nodes learn about the prefix that is used to route packets to the NAT-PT box. If the prefix is pre-configured in IPv6 nodes, the IPv6 node would prepend the preconfigured prefix to the address of any IPv4-only node with which it wants to initiate communications. However, with a prefix, there might be a reachability problem if the NAT-PT box were to shut down unexpectedly. If an attacker would somehow be able to give the IPv6 node a fake prefix, the attacker would be able to steal all of the node's outbound packets to IPv4 nodes.

Even though this is not specified in RFC2766, DNS servers and DNS-ALGs should be used for outgoing connections to return the prefix information to the IPv6 node as a means to avoid the problem of a statically preconfigured prefix. When an IPv6-only node wishes to initiate communications with an IPv4-only node, its resolver would send an AAAA query. This query can be passed through the DNS-ALG which itself looks for an A record. In this case the DNS-ALG can prepend the appropriate prefix for NAT-PT itself and thus return a full AAAA record to the IPv6-only node. The DNS-ALG can also monitor the state of a number of NAT-PT boxes and use only the prefixes of those that are running. The method by which a DNS-ALG determines the state and validity of a NAT-PT box must of course also be secure. The DNS-ALG and each NAT-PT box should be configured with a pairwise unique key that will be used for integrity-protected communications. Note that messages from DNS-ALGs are not integrity-protected and can therefore be modified. To prevent such a modification, a DNS-ALG can sign its packets. The DNS-ALG's public key can be made available like that of any other DNS server (see RFC2535) or presented in the form of a certificate that has a root CA that is well know to all nodes behind the NAT-PT gateway. A shared-key technique may not be as practical.

4.1.3 Source address spoofing attack

There are two cases in which an attacker will use NAT-PT resources, one where the attacker is in the same stub domain as the NAT-PT box and the second where the attacker is outside the NAT-PT stub domain.

Suppose that an attacker is in the same stub domain as the NAT-PT box and sends a packet destined for an IPv4-only nodeto the other side of the NAT-PT-gateway, forging its source address to be an address that topologically would be located inside the stub domain. If the attacker sends many such packets, each with a different source address, then the pool of IPv4 addresses may quickly get used up, resulting in a DoS attack (or rather Address depletion attack). A possible solution to this attack as well as to similar attacks like resource exhaustion or a multicast attack is to perform ingress filtering on the NAT-PT box (which is the border router). This would prevent an attacking node in its stub domain from forging its source address and thus from performing a reflection attack on other nodes in the same stub domain. The NAT-PT box should also drop packets whose IPv6 source address is a multicast address. Address Depletion attacks can be prevented by employing NAT-PT in a way that it translates the TCP/UDP ports of IPv6 nodes into the corresponding TCP/UDP ports of the IPv4 nodes/addresses. However, sessions initiated by IPv4 nodes are restricted to one service per server. Of course IPsec might be used to further increase security.

Suppose now that an attacker outside the NAT-PT domain sends a packet destined to an IPv6-only node inside the NAT-PT domain and forges its (IPv4) source address to be an address from the IPv4 address pool used for NAT-PT. The same attacks are then possible as in the scenario above. Again filtering can be used to prevent this. The NAT-PT gateway should drop all packets whose IPv4 source address is a broadcast/multicast address. It should also filter out packets from outside that claim to have a source address from inside the NAT-PT domain.

4.2 TRT

While TRT is a nice mechanism to support IPv6-only networks with IPv4 connectivity for more than just one application it has a few drawbacks that also include security and management issues.

- It supports only unicast TCP/IP traffic, however it is theoretically possible to implement multicast support as well.
- TRT is more difficult to scale than the stateless translation methods. The TRT relay server has to keep track of all the "TRT" connections to properly handle all error conditions. The scaling problem can be eased using anycast technology to reach the closest TRT relay server.
- The TRT relay server could potentially generate a major security problem, since it can be used as an intermediate hop to reach IPv4 servers. The served community of a TRT relay server therefore has to be carefully controlled by packet filtering or access control lists. To reduce the problem site local addresses (or their respective replacements) could be used for accepting incoming IPv6 packets
- TRT requires a specially configured DNS server (DNS-ALG) to run.
- Due to the nature of the TCP/UDP relaying service, it is not recommended to use TRT for protocols that use authentication based on source IP address (e.g., rsh/rlogin).
- IPsec cannot be used across a TRT relay.

4.3 ALGs

Application Layer Gateways mostly act as a kind of proxy or relay between IPv6-only clients requesting access to a service or rather server that is only accessible via IPv4. Like with any relay, in terms of security the most important threat is therefore an attacker using the relay to obscure his

identity and source address. As SPAM becomes more and more of a problem on the Internet, this is particularly true for mailservers acting as SMTP relays between IPv6 and IPv4. However, employing some kind of access control can lessen this threat to great extent.

In terms of management ALGs are not really different than any other (dual-stack) server. The only issue might be performance if more and more people start using it and it becomes a kind of bottleneck for the requests and traffic corresponding to an application. Load, traffic and the amount of users should therefore be monitored to alert the administrator well ahead of a server running out of capacity.

4.3.1 Security issues arising when using a DNS-ALG

A DNS-ALG is required whenever IPv6-only nodes should be allowed to initiate communication with IPv4-only nodes for example within a NAT-PT scenario. Since the DNS-ALG will translate simple “A record” requests into “AAAA record” requests and vice versa DNS-SEC will not work in this case. However, as pointed out in draft-durand-v6ops-natpt-dns-alg-issues, if the hosts set the “AD is secure”-bit in the DNS header, it is possible for the local DNS server to verify signatures. Another option to increase security is for the DNS-ALG to verify the received records, translate them and sign the translated records anew. A third option would be if the host had an IPSec security association with the DNS-ALG to protect DNS records.

5 IPv6 multicast

IPv6 multicast is a special kind of IPv6 functionality and while some of the above mentioned transition mechanisms like tunnelling techniques and dual-stack networks can also be used for IPv6 multicast traffic, there are also some special mechanisms designed exclusively for the migration from IPv4 to IPv6 multicast.

5.1 IPv6 multicast translators

There are several issues with such translators, many of which are similar to NAT-PT.

One problem is caused by the fact that the translator needs a set of IPv4 and IPv6 addresses that is routed to the device, and which it can use as source addresses in translated packets. That way if the translator accepts traffic from outside the site, packets may appear to come from within the site but in reality they come from the outside. They are just translated within the site. This means one has to consider carefully where to place the device in the network. When doing packet filtering in the network one should be able to have different policies for packets coming from the translator and packets that truly originated within the site. This may be less of a problem for multicast than unicast.

A special case for multicast is the concept of Rendez-Vous points used with the PIM-SM protocol. They may have restrictions on who can send multicast traffic to it and who can't. In this case one must take into consideration the source addresses used by the translator.

The translator itself might also be configured to allow only specific packets to be translated. It can restrict to both specific sources and specific multicast groups.

Translators may make it harder to track the real source of data packets. In many cases one needs access to the translator or logs from the machine to find the real source of certain packets. However, when translating from IPv4 to IPv6 one will often encode the complete IPv4 source address into the IPv6 source address.

One particular problem for multicast translators is how to handle scopes. There are several types of scopes for both IPv4 and IPv6 multicast that should be mapped into one another. If for example IPv4 groups with site-local scope were mapped into IPv6 groups with global scope, there is a danger that packets sent to these IPv4 groups are forwarded to IPv6 receivers outside the site.

5.1.1 Specific issues with the translator by Stig Venaas

The issues described in this section refer specifically to multicast translators implemented according to [v6v4McastGW].

This translator acts as an RP for a /96 IPv6 prefix. Once set up the translator can be used by anyone within the domain where the RP itself is located. If embedded-RP [EmbeddedRP] is used in the configuration, the gateway might be globally reachable. To control who can use the translator, one should control who can use it as an RP. This can be accomplished by controlling where the RP-mapping is defined. If the use should be limited to some administrative domain, the best way is to use the appropriate scope for the /96 prefix. By using e.g. site-local scope, the translator is only reachable within the site. The RP may also have restrictions from who it accepts PIM register messages.

One may wish to restrict which IPv4 groups are reachable through the gateway. In particular, if a global /96 IPv6 prefix is used, one should in most cases not include non-global IPv4 groups. For this it might be useful to have some access lists for defining which IPv4 groups to give access to. So far the only one known implementations of this draft does not yet include access lists or scope checks.

When receiving translated packets, it may be difficult to find out, who the real source is. The translated packets all contain one fixed IPv4 address and one fixed IPv6 address as source address. For RTP packets however, one may identify the source from the RTP header. On the other hand RTP header information could easily be wrong or forged. The draft suggests that one may use an IPv6 source address where the last 32 bits is the IPv4 source address. The current implementation however, does not do this.

6 Recommended tools to manage and secure transition mechanisms

There are (to date) no management tools specifically developed to monitor and mange IPv4/IPv6 transition mechanisms. The user is instead referred to normal IPv4- or IPv6-tools, which in most cases can be adapted to perform the needed tasks for e.g. tunnelled links etc.

With tunnels in particular one often has the problem that SNMP-based tools have no way of differentiating between normal and encapsulated traffic, since appropriate MIBs have not yet been standardized. To remedy that the IPv6 working group at the IETF is currently working on a specification for a special IP tunnelling MIB [TunnelMIB].

Where CPU-performance is of interest (translators, gateways) there is no difference in monitoring the state of a server between IPv4 or IPv6.

7 Conclusion

There are many mechanisms to support the coexistence of IPv4 and IPv6. Most of them are already widely deployed (tunnelling, dual-stack and sometimes even translation techniques) and the experience gained with them lead to the compilation of the operational procedures described in this deliverable.

For every transition mechanism covered, network administrators can find descriptions of known problems as well as recommendations regarding management and solutions to security threats raised by their deployment. The lack of security for some of the transition mechanisms has also been reported (e.g. DSTM dynamic tunnel interfaces) as well as the need of some supervision tools for the new features offered by the transition mechanisms. Together with the cookbooks D2.3.3 and D2.2.3 and their respective updates this document can be used as a basis for securely implementing and deploying such tools.

References

[DSTM] “Dual-stack Transition Mechanism (DSTM)”, J. Bound et al, IETF Internet Draft, December 2002, (draft-ietf-ngtrans-dstm-08.txt)

[RPC] “RPC: Remote Procedure Call Protocol Specification Version 2”, R. Srinivasan, IETF RFC 1831, August 1995

[TSP1] “Tunnel Setup Protocol (TSP): A Control Protocol to Setup IPv6 or IPv4 Tunnels”, M. Blanchet, IETF Internet Draft, July 2002, (draft-vg-ngtrans-tsp-01.txt)

[TSP2] “DSTM IPv4 over IPv6 tunnel profile for Tunnel Setup Protocol(TSP)”, M. Blanchet, O. Medina, F. Parent, IETF Internet Draft, July 2002, (draft-blanchet-ngtrans-tsp-dstm-profile-01.txt)

[TSP3] “IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)”, draft-blanchet-v6ops-tunnelbroker-tsp-00.txt, M.Blanchet, February 2004

[DHCPv6] „Dynamic Host Configuration Protocol for IPv6 (DHCPv6)“, R. Droms et al, IETF Internet Draft, November 2002, (draft-ietf-dhc-dhcpv6-28.txt)

[FreeNet6] “FreeNet6 web site”, FreeNet6, http://www.freenet6.net, as accessed in January 2003

[Sec6to4] “Security Considerations for 6to4”, P. Savola, IETF Internet Draft, January 2003, (draft-savola-v6ops-6to4-security-02.txt)

[KIN] “Requirements for Kerberized Internet Negotiation of Keys”, M. Thomas, IETF RFC 3129, June 2001

[6to4] “Connection of IPv6 Domains via IPv4 Clouds”, B. Carpenter, K. Moore, IETF RFC 3056, February 2001

[ISATAP] “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)”, F. Templin et al, IETF Internet Draft, January 2003, (draft-ietf-ngtrans-isatap-12.txt)

[NATPT] “Network Address Translation – Protocol Translation (NAT-PT)”, G. Tsirtsis, P. Srisuresh, IETF RFC 2766, February 2000

[v6v4McastGW] “An IPv4 – IPv6 multicast gateway”, draft-venaas-mboned-v6v4mcastgw-00.txt, S. Venaas, February 2003

[EmbeddedRP] “Embedding the Rendevous Point (RP) Address in an IPv6 Multicast Address”, draft-ietf-mboned-embeddedrp-04.txt, P.Savola, B.Haberman, April 2004

[TunnelMIB] “IP Tunnel MIB”, draft-ietf-ipv6-inet-tunnel-mib-00.txt, D.Thaler, January 2004

[PMTUD1] “Path MTU Discovery”, J. Mogul, S. Deering, IETF RFC 1191, November 1990

[PMTUD2] “Path MTU Discovery”, draft-ietf-pmtud-method-01.txt, M. Mathis, February 2004