SixXS Heartbeat Protocol
draft-massar-v6ops-heartbeat-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on December 6, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document proposes a heartbeat protocol for signalling availability of hosts with a specific emphasis on providing a signalling protocol for allowing dynamic non-24/7 endnodes to use tunnel’s of the various IPv6 Tunnel Brokers.
1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

When operating a IPv6 Tunnel Broker [RFC3053] with one or multiple Tunnel Servers [RFC3053] or any other form of tunneling across the Internet one doesn’t want to send packets out to unsuspecting endpoints. Unsuspecting endpoints are hosts which don’t expect traffic from the Tunnel Server. This is a common scenario when a non-static but configured endpoint is used. The user enables the tunnel to the current endpoint, uses it and then disconnects from the Internet or gets another IP, due to dhcp policies or various other reasons. Though the endpoint of the tunnel changes but there currently is no way of automatically changing that endpoint or notifying the Tunnel Server that it has changed. Current solutions include using a script to update the endpoint on the Tunnel Broker’s website and various other non-standardized methods.

Then another host get to use the IP which terminates the tunnel and suddenly it receives our tunneled packets. This could be seen as a security risk as the new user of the IP could sniff the packets and look at them or could even setup the tunnel and take over connections which where active on that tunnel impersonating the former host using the tunnel. Many hosts employ so called firewall or logging facilities that will start complaining about packets being sent from the Tunnel Server which they perceive as malicious.

The user of this firewall will in turn start complaining about abuse and hack attempts which takes up time from the staff running the Tunnel Broker and not even mentioning all the entities that will be notified by these type of messages.

This protocol resolves this problem by specifying a oneway heartbeat protocol. As long as the tunnel endpoint is able to send an authenticated heartbeat packet the tunnel is kept configured on the side of the Tunnel Server allowing two-way traffic to be sent. This heartbeat protocol also allows for automatic tunnel reconfiguration so that in the event of an IP change at the client side the tunnel will be reconfigured allowing continued service without a (notable) interruption. If the Tunnel Server doesn’t receive a heartbeat for a certain amount of time it can deem the endpoint as down and deconfigure the tunnel. Taking these two into consideration will stop the sending of tunneled packets and will thus avoid the complaints which could have been brought up before.
This protocol covers all tunneling protocols which use at least one set of inner and one set of outer addresses. Note that some of these protocols already have a heartbeat mechanism in the protocol. In those cases it is of course advisable to use that mechanism.

3. SixXS Heartbeat Protocol

The heartbeat packet is a UDP [RFC0768] packet to IANA assigned port 3740 of the server to which the signalling is to take place. Schematically this will look like the following diagram.

```
+--------+                    +--------+
| Client | <--- Internet ---| Server |
+--------+                    +--------+
```

The heartbeat packet consists of one single line separated by spaces and ending in a NULL. The first part is the command thus allowing extension of this protocol if wanted in the future. Extensions are allowed to put additional commands on separate lines, currently those extensions are not defined but these could be used for transmitting other information like load of the client which could be useful in failover scenario’s.

The basic heartbeat packet looks like the following in ABNF [RFC2234]. The IPv6address and IPv4address ABNF are defined in "APPENDIX B: ABNF Description of Text Representations" of the "IP Version 6 Addressing Architecture" [RFC3513].

```plaintext
packet = command SP epochtime SP md5-signature NULL

command = *%x41-5A [*(SP option)]

option = *VCHAR

epochtime = *DIGIT

md5-signature = *DIGIT

NULL = %x00
```

Epochtime is the time in seconds since "00:00:00 1970-01-01 UTC" which corresponds to the output of the “date +%s” command. Both the client and the server are advised to be synchronized using NTP [RFC2030] to make sure that the system clocks of the hosts don’t differ to much even after travelling the intermediate networks between the client and the server.

The packet must be constructed first with the signature containing
the password for the tunnel. After construction the md5-signature should be calculated and then replace the cleartext-password. The packet can then be sent to the server. This method thus allows verification of the password without sending the password over the network. The Server does the same thing, taking the command part of the packet, adding the password and calculating the md5 signature which can then be compared with the md5-signature which was sent by the client. If these match the packet can be processed further. When the md5-signatures don’t match the server MUST silently ignore the packet.

Calculation of the MD5 signature [RFC1321] MUST be done over the complete string excluding the NULL which is the string terminator in C and most other programming languages. The scenarios section demonstrates how this is performed.

The currently defined commands HEARTBEAT and DISABLE both use the following common options to identify the endpoint for which this heartbeat is meant:

common-options = ("HOST" endpoint | "TUNNEL" outer-endpoint SP inner-endpoint)

outer endpoint = endpoint | "sender"
inner endpoint = endpoint
endpoint = IPv4address | IPv6address

An endpoint MAY also be a MAC address or other defined ‘address’.

3.1 Heartbeat Command

This command is the main reason for this protocol to exist. It tells the Tunnel Server that the current endpoint specified by both the UDP’s source address and the common-option’s outer endpoint is still accepting tunneled packets.

Specification:

command = "HEARTBEAT" SP common-options

Example’s:

Heartbeat from client host 192.0.2.2 :

HEARTBEAT HOST 192.0.2.2 257209200
Heartbeat from client host 192.0.2.2 for IPv6 tunnel endpoint 2001:db8::2:

HEARTBEAT TUNNEL 2001:db8::2 192.0.2.2 409100400

In this example the server determines the inner-endpoint based on the sender IP of the packet. This could be used in all cases but is especially useful when the client is behind a NAT and doesn’t know it’s external address. An application MUST not use the ‘sender’ replacement when it knows it’s public IP address. This because it adds an extra verification step to the heartbeat.

HEARTBEAT TUNNEL 2001:db8::2 sender 257209200

3.2 Disable Command

The disable command explicitly sets a tunnel ‘down’ on the Tunnel Server allowing a gracious and prompt shutdown of the tunnel on the Tunnel Server. One could even employ this command to stop huge traffic flows to the local endpoint which maybe could not cope with the traffic passed over it through the tunnel.

Specification:

command = "DISABLE" SP common-options

Example:

Disable the tunnel endpoint 2001:db8::2 to 192.0.2.2 :

DISABLE TUNNEL 192.0.2.2 2001:db8::2 409100400

4. Acknowledgements

The protocol presented has formed during the existence of IPng [IPNG] and SixXS [SIXXS] to allow the users of these tunnel broker systems to have a dynamic non-static IPv4 endpoint. This protocol is already in use by SixXS since October 2003 and has made a lot of users very happy and I would like to thank the numerous users requesting the support for dynamic non 24/7 reachable tunnel’s. This was before this protocol was implemented one of the few requirements of obtaining a tunnel even when the project only consisted of one Tunnel Server and was called IPng.
5. Security Considerations

The password used for creating the MD5 [RFC1321] signature should never be made publicly available to 3rd parties otherwise that 3rd party could sign a packet and automatically reconfigure the tunnel endpoint. This could lead into the 3rd party sending traffic in both directions and thus posing as the actual user.

The inclusion of the timestamp along with the verification on the Tunnel Server side should guard against any replay attacks. The Tunnel Server MUST limit that the local clock compared to the timestamp from the packet MUST never differ for more than 60 seconds, this allows for at least some latency and time-desync.

Any packet that is not well formed or contains a invalid signature MUST be silently dropped.

A side effect of this protocol is that whenever the local host cannot send a heartbeat in time to the Tunnel Server that it will deconfigure the tunnel. Envision a Denial of Service over the tunnel to the local endpoint. The local host’s heartbeat cannot reach the Tunnel Server which in turn will automatically disable the tunnel, allowing the host to catch some breath again and thus effectively implementing a poor mans rate limiting.

6. Scenarios

6.1 Host heartbeat

This scenario handles the case where a host, the client using either IPv6 or IPv4, sends notifications of it’s aliveness to a server. This could be used for server management and fallover situations. In combination with extension commands, used on the additional lines in the packet the client could send along information about cpu load etc allowing the server to gather information about those factors.

+--------+                    +--------+
| Client  | <--- Internet ---> | Server |
+--------+                    +--------+

Server’s IPv6 : 2001:db8::1/64
Client’s IPv6 : 2001:db8::2/64
Password      : point
The command, including the password and epochtime would be:

HEARTBEAT HOST 2001:db8::2 409100400 point

The packet to be transmitted, after md5 translation is:

HEARTBEAT HOST 2001:db8::2 409100400
bd72fb8d98b8698fa70cdfeb33bb7342

6.2 IPv6 in IPv4 tunnel heartbeat

This scenario handles the case where a IPv6 in IPv4 tunnel [RFC3056] is setup between two hosts. This is the normal case for a Tunnel Broker system with which a IPv4 only host is given access to a IPv6 network. A tunnel like this can easily be setup using the Tunnel Setup Protocol [I-D.vg-ngtrans-tsp], SixXS Configuration Service [TIC] or similar protocols. A diagram of such a setup:

```
+--------+                               +--------+
| Client | <--- IPv6 in IPv4 Tunnel ---> | Server |
+--------+                               +--------+
```

Server’s Outer IPv4 : 192.0.2.1
Server’s Inner IPv6 : 2001:db8::1/64
Client’s Outer IPv4 : 192.0.2.2
Client’s Inner IPv6 : 2001:db8::2/64
Password            : hartslag

The command would be:

HEARTBEAT TUNNEL 2001:db8::2 192.0.2.2 1051480800 hartslag

The data to be transmitted is:

HEARTBEAT TUNNEL 2001:db8::2 192.0.2.2 1051480800
3f0a026ed8b15e7c1a7a2d92b3c446a

Disabling the same tunnel can be accomplished using:

DISABLE TUNNEL 2001:db8::2 192.0.2.2 1055628000 hartslag

The data to be transmitted is:

DISABLE TUNNEL 2001:db8::2 192.0.2.2 1055628000
53d5bb7bfe4a3a80da01227da02cda24
The usage for which this protocol was defined in the first place is signaling the server of the current IPv4 address of the client. When the server receives a heartbeat packet it looks up the configuration information based on the supplied Inner IPv6 address, verifies that the password is correct by constructing the command and generating the md5 signature. When the md5 signature is incorrect the server MUST silently discard and ignore the packet. A check MUST also be made comparing the Inner IPv4 address with the IPv4 address from which the packet was sent, when incorrect the server MUST discard and ignore the packet. Logs noting these event MAY be made depending on the wishes of the administrator. When the password is correct and thus the md5 verifies the server will configure the IPv6 in IPv4 tunnel to point at the IPv4 address specified by the Inner IPv4 parameter. When the Inner IPv4 parameter is ‘sender’ the server MUST use the IPv4 address from which the packet originates.

Note that TEST-NET [RFC3300] addresses could never reach a Tunnel Server over the public Internet due to filtering of this documentation prefix.

7. IANA Considerations

IANA already allocated port 3740 for the Heartbeat Protocol in April 2003. Thus no further actions from IANA are required.

8. References

8.1 References


8.2 Informative References


Author’s Address

Jeroen Massar
Unfix/SixXS
Hofpoldersingel 45
Gouda 2807 LW
NL

Email: jeroen@unfix.org
URI: http://unfix.org/~jeroen/
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.