                   Socket API Extension for MIF PvD Architecture
                        draft-liu-mif-socket-api-00

Abstract

   IETF MIF working group defines the multiple provisioning domain
   architecture. This document proposes API extension for the PvD-aware
   node to support the MIF PvD architecture.

Requirements Language

   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 RFC 2119 [RFC2119].

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   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."

   This Internet-Draft will expire on January 6, 2016.

to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   IETF MIF working group defines the multiple provisioning domain
   architecture in draft-ietf-mif-mpvd-arch-10 [mpvd-architecture] . It
   defines three levels of PvD support in API: basic, intermediate and
   advanced. This document discusses the advanced PvD API for the PvD-
   aware node.

2. Current PvD-related API implementation

   This section summarize the PvD related API implementations. The
   purpose of this section is to help analyzing the extension of current
   API implementation to support PvD architecture.

2.1. PvD-related API Implementation in Socket API

   The basic socket API includes the following:

   Socket API for a typical server:

   o socket()

   o bind()

   o listen()

   o recvmsg()

o sendmsg()

o close()

Socket API for a typical client:

o socket()

o connect()

o sendmsg()

o recvmsg()

o close()

[RFC3493] extends the basic socket API to support IPv6. It defines
the IPv6 Address Family and Protocol Family and also the socket
address structure, socket options etc.

[RFC3542] defines the advanced sockets API for IPv6. It defines the
socket API to access IPv6 specific parameters. For example, the IPv6
raw socket, the API to access IPv6 and extension headers etc.

[RFC5014] defines the IPv6 socket API extension for source address
selection. It can be used to override the default source address
selection method as defined in [RFC3484] . It defines an address
preference flags that used for the source address selection.
Developers can use this API to explicitly specify the source address
to be used in the communication. Example of use cases of this source
address selection API includes applications that supporting Mobile
IPv6, IPv6 Privacy Extensions, Cryptographically Generated Addresses
etc. It uses per-socket and per-packet flags to implement the source
address selection. It adds a new socket option at the IPPROTO_IPV6
level. The new option is called IPV6_ADDR_PREFERENCES. It can be
used with setsockopt() and getsockopt() calls to set and get the
address selection preferences affecting all packets sent via a given
socket.

3. Extension for PvD advanced API

This section defines the extension of socket API to support PvD
architecture as defined in [mpvd-architecture]

It belongs to the advanced PvD API discussed in section 6.3 of
[mpvd-architecture]. The extension proposed in this document has the
following types of API extension:

o API to get current PvDs that been provided to the node

o API to explicitly select a PvD

o API for DNS resolution

There are different design alternatives for the PvD API. Including:

o Get PvDs and select PvD per-socket.

o Get PvDs and select PvD per-application.

o Get PvDs and select PvD per-node.

This document propose the per-socket approach since it can provide
the maximal flexibility for the application developers to meet all
the kinds of use cases.

3.1. Get PvD Configuration API

The following API is used to get the current PvD configuration of the
node:

o getpvdinfo()

The definition of this API is:

int getpvdinfo(const char *nodename, const char *servname, struct
pvdinfo **res);

The structure of struct pvdinfo is:

struct pvdinfo {

int sockaddr * ai_addr;

int sockaddr * gateway_addr;

int sockaddr * dns_addr;

struct addrinfo * ai_next;

}

The definition of parameters is as follows:

o nodename and servname: The nodename and servname parameter are
   pointers to null-terminated strings or NULL. One or both of these

parameter must be a non-null pinter. A non-null nodename string
can be a node name or a numeric host address string.

o res: The pvdinfo structure. The result is pointed to res
   structure.

3.2. Set PvD API

The following API is used to select the specific PvD.

o setsockopt()

setsockopt(int s, struct * pvdinfo pvd)

The struct * pvdinfo pvd is a new parameter that used to specify the
preferred PvD. The socket can be set to use the PvD that specified
by pvdinfo parameter.

All the socket related operation will be bind to this PvD. For
example, The connect() API call should use the set of configuration
parameters that contained in the pvdinfo (source address, gateway and
DNS etc).

3.3. DNS Resolution

getaddrinfo() is the socket API used to resolve the IPv4 and IPv6
address. This document proposes to extend getaddrinfo() socket API
to allow it use PvD information as a parameter for DNS resolution.

int getaddrinfo(const char * hostname, const char * service, const
struct addrinfo * hints,struct * pvdinfo pvd, struct addrinfo **
result);

The DNS resolution should use the DNS server that contained in the
PvD parameter.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

TBD.

6. Acknowledgements

The author would like to thank the PvD API design team.

7. Normative References

[mpvd-architecture]
          Anipko, D., "Multiple Provisioning Domain Architecture",
          February 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3484] Draves, R., "Default Address Selection for Internet
          Protocol version 6 (IPv6)", February 2003.

[RFC3493] Gilligan, R., "Basic Socket Interface Extensions for
          IPv6", February 2003.

[RFC3542] Stevens, W., "Advanced Sockets Application Program
          Interface (API) for IPv6", May 2003.

[RFC5014] Nordmark, E., "IPv6 Socket API for Source Address
          Selection", September 2007.

Author's Address

   Dapeng Liu

   Email: maxpassion@gmail.com