pland: The PLAN Active Router*

for PLAN version 3.1

Michael Hicks

July 25, 2001

Synopsis

```
pland [-router] [-firewall d1,d2,...,dn] [-l log] [-ip port]
      [-rf rout_tab_file] [-hf host_file] [-key key_file]
      [-policy policy_file] [-authlist h1,h2,...,hn] ifc_spec_file
```

Description

`pland` is the daemon program that acts as the PLAN active router for PLAN version 3.2, written in OCaml [1] version 2.02. Currently, pland only works on UNIX-based systems. There also exists a PLAN active router written in Java (heretofore referred to as `ARMain` after its classfile name), for PLAN version 2.2, which has its own documentation. Pland and ARMain are not compatible with one another.

Pland communicates with local applications (referred to as *host applications*) via UNIX domain sockets by default, or via TCP/IP sockets using a special compilation option. See the `-ip` option description for more.

Depending upon the compilation flags used, pland can make use of any combination of two link layers: Ethernet and UDP/IP. If there is interest, we may also enable ATM as a link layer. Currently, only Linux-based versions of pland can use Ethernet, enabled by a patched version of OCaml [2] which reveals a special Linux device socket type. How these link layers are used is decided by the contents of the *interface specification file* (`ifc_spec_file`); details on the format of this file are specified below.

PLAN (network layer) addresses are made up of two components: a 32 bit address and a 16 bit port number. When running on top of Ethernet the choice of these two fields is completely arbitrary, while when running on top of UDP/IP the address portion must match the IP address of the underlying machine and the port portion must be an unused UDP port. These two link layers can be

*This work was supported by DARPA under Contract N66001-96-C-852.

used singly or in combination; essentially this allows separate Ethernet-based PLANet's to be interconnected by 'tunnels' through the Internet. Also, the use of UDP/IP allows multiple routers to run on the same host, which is useful for testing purposes.

To enable service-level security, a PLAN node may be associated with a public and private key pair. These keys are stored in a file as base-64 encodings of DSA public and private keys. A tool, `plan_keygen`, is provided with the distribution to generate these keys. Additionally, you may specify a policy file to indicate the node's service access policy at startup time. This file depends on the policy manager compiled into pland.

Pland can either use static or dynamic routing; the former is set by a table at startup, while the latter (the default) is realized via a RIP-style routing protocol.

The 'standalone' version of pland does not allow dynamic linking of code. Also in the distribution is a version of pland which is integrated with the active loader [3], called `plan_loader`. This version is slightly slower.

For a tutorial on using the router (and the injector) please refer to the PLAN tutorial [6]; for more detailed information, please refer to the PLAN Programmer's Guide [5].

Command-line options

`-router` Specifies *router mode*; this option eliminates the ability of the node to receive local traffic via the implicit port. This should have a performance advantage.

`-firewall` *dev1*,…, *devn* Specifies the devices, as given in the interface specification file, to watch as a firewall. All packets transmitted on these interfaces are encapsulated with a wrapper chunk that limits their capabilities. This is described in more detail in the Security Guide [7].

`-l` *logfile* Specifies the name of the file to write logging messages to. Messages are *not* (unlike the Java version of PLAN) logged in append-only fashion, so the logfile may not be safely deleted at any time (ugh). Specifying a `-` will write messages to stdout. If no logfile is specified, the default is `log` in the current directory.

`-ip` *incoming port* Specifies the number of the node's implicit/injection port. This number is translated into the file `/tmp/sware`*num* to be used by UNIX domain sockets, or to the TCP listen port if TCP/IP sockets are used. In order to work with the `inject` program correctly, this number should also match the port portion of one of the network addresses of the node (as specified by the interface file, see below).

`-rf` *routfile* Specifies that static routing should be used, and that the table may be found in *routfile*. This file has the form of pairs of hosts, the first of which indicates a node in the net, and the second indicates the next "hop" to that node. If the first node of the pair is the word `default`, then the hop serves as the default route.

`-hf` *hostfile* Specifies the hostfile that should be used to resolve hostnames in the `getHostByName` and `thisHost` PLAN routines. This file should have the same format as `/etc/hosts`; the default is set to `EXP_IP_ADDRS` in the current directory.

`-key` *keyfile* Specifies the file that contains a base-64 encoding of the DSA public key and private key for this node. If this option is not specified, then the file in the local directory with the name `key.`*portnum* where *portnum* is node's implicit port is used. If this file doesn't exist, a default key is used.

`-policy` *policyfile* Specifies the file that contains a program describing the node's security policy. In the current release, the policy manager compiled by default is QCM, and so this file would contain a QCM program. Otherwise, the monolithic policy manager is used. The form of both files is described in more detail in the Security Guide [7].

`-authlist` *h1,...,hn* Specifies the list of hosts to establish security relationships with. After the node starts up, each host is contacted in turn, and the nodes authenticate each other using the protocol described in the Security Guide [7]. The result is a shared secret which is used in future communications between the nodes.

Interface File Description

The interface file describes the underlying link layer arrangment that pland should use. In general terms, the interface file consists of a list of devices which either are (in the case of Ethernet) or represent (in the case of IP) the network interfaces to be used by pland. Each interface has broadcast, rather than point-to-point, semantics. This file specifies for each interface its name, its type, its network address, and an optional list of neighbors. The list of neighbors is used to simulate broadcast for IP-based linklayers (i.e., when a PLAN program broadcasts out of a particular IP device, all of the hosts in the neighbor list shall receive the message). The grammar for the file is given below.

Lexical definitions

int		an integer
id		an identifier
string		a string of characters
ip-addr	=	`[0-9][0-9][0-9].[0-9][0-9][0-9].[0-9][0-9][0-9].[0-9][0-9][0-9]`
dns-name	=	id (`[.]` id)*
addr	=	(ip-addr | dns-name)
	|	(ip-addr | dns-name) `':'` int
type	=	`'ethernet'` | `'ip'`

Grammar

file	::=	int *entry-list*
entry-list	::=	*entry* | *entry entry-list*
entry	::=	string type addr *addr-list*
addr-list	::=	addr | addr `','` *addr-list*

Consider the following example:

```
3
ip0 ip kamagarini:3324 kamagarini:3323,kamagarini:3327
ip1 ip kamagarini:3325 kamagarini:3326
eth0 ethernet kamagarini:3326
```

This file specifies that pland has three interfaces, `ip0`, `ip1`, and `eth0`, the first two of which are virtual IP interfaces, and the third is an actual Ethernet interface.

The name of the ethernet interface, `eth0`, must match the name of an Ethernet device known by the system[1]. The network address of the interface has no restrictions on it (i.e., it doesn't have to match the IP address of the local machine) other than it should be different than the address of all of the other interfaces. There is no need to specify a list of neighbors; broadcasts will actually make use of Ethernet broadcast to send to all hosts on the same Ethernet LAN as the device.

[1]This option only works on Linux systems. To determine the names of valid devices, use the `ifconfig` command. Pland must be running as the superuser to access these devices.

The network addresses of IP interfaces `ip0` and `ip1` are `kamagarini:3324` and `kamagarini:3325` respectively. Note that if pland uses one or more IP interfaces then the first portion of the address (the `kamagarini` part) must be the IP address of the machine upon which pland is running (and thus must be the same for all interfaces), and the second part (here `3324` or `3325`) must be different for each interface. This is because each IP interface results in the creation of a UDP/IP socket in pland which is bound to the specified port. In this case, pland will bind to UDP ports 3324 and 3325 when it is running on machine with domain name `kamagarini`; UDP packets received on port 3324 will be attributed to device `ip0` and those on 3325 will be attributed to `ip1`. The neighbor list for each IP device specifies the network layer addresses of machines to send to if a broadcast message is sent out of the device. This list is essentially the network addresses of all nodes to be considered on the same LAN as the interface (minus the address of the interface itself). Currently, you must specify at least one neighbor.

The device name `lo` is reserved (it specifies the local pseudo-device). If the `':' int` part of the address is left off, the default port is used, as specified on the command line with `-ip`, or 3324. Note that for the `inject` program to work properly, the port number given to `-ip` when pland is started should match one of the port portions of the network addresses specified in the interface file. In this example, pland could be safely started with `-ip`'s argument being one of 3324, 3325, or 3326.

References

[1] X. Leroy. "The Objective Caml system, documentation and user's guide," `http://pauillac.inria.fr/ocaml/htmlman`.

[2] `http://www.cis.upenn.edu/~switchware/activebridge/software/ocaml-patch`.

[3] `http://www.cis.upenn.edu/~switchware/activebridge`.

[4] D. Scott Alexander et. al, "Active Network Encapsulation Protocol (ANEP)," RFC, ANEP documentation, 1997.

[5] J.T. Moore and M.W. Hicks, "PLAN Programmer's Guide," PLAN documentation.

[6] P. Kakkar with M.W. Hicks and J.T. Moore, "The PLAN Tutorial," PLAN documentation.

[7] M.W. Hicks, "PLAN Security Guide," PLAN documentation.