

CESNET Technical Report 23/2009

High Available *eduroam* RADIUS server

JAN TOMÁŠEK

Received 16.12.2009

Abstract

This document describes the national RADIUS proxy server of the Czech *eduroam* federation implemented as a high available cluster, consisting of two nodes housed in two geographically separated localities. The cluster acts as a single IP address to ease setup of the RADIUS servers at the side of the connected organisations. Switchover between active and passive node is done by Gratuitous ARP packet. The control and the monitoring of the cluster is done by the heartbeat daemon from the project Linux-HA¹.

Keywords: eduroam, RADIUS server, High Availability, HA-Linux, heartbeat

1 Introduction

A usual method how to achieve high availability of the RADIUS server is to deploy several independent RADIUS servers. This method works very well in simple scenarios where RADIUS servers are authoritative. If a more complex infrastructure is hidden after a local RADIUS server, it might become unresponsive because of this hidden infrastructure. If the RADIUS server becomes unresponsive, it is marked as dead by a “client” (by a “client” an AP or a RADIUS server is meant) and another RADIUS server from a pool is chosen. If there are no more remaining RADIUS servers in the pool, authentication fails.

The hierarchical *eduroam* federation is a good example of such a complex infrastructure. Marking RADIUS server as dead is not a reliable method how to deal with failing RADIUS servers in *eduroam*. Better method is marking only as dead a realm instead of a complete RADIUS server. This dead realm marking² method is implemented for Radiator, but in *eduroam* infrastructure other RADIUS server implementations are also used. Another promising approach is RadSec, which uses TCP as a transport mechanism. With RadSec, the status of the TCP connection can be used as indication if a peer RADIUS server is alive. Sadly, RadSec is available only for UNIX based platforms and we also need a solution for Windows platform.

Czech *eduroam* federation was using two national RADIUS servers since autumn 2004 until summer 2006, when a serious malfunction of the first server happened. During the period the first server was down, only some institutions managed to use the second server correctly. Problems were very likely caused by an incorrect setup at the side of the institutions, but this type of flaws in the setup is very hard to detect before a real failure occurs.

¹ <http://www.linux-ha.org/>

² <http://wiki.eduroam.cz/dead-realm/docs/dead-realm.html>

Absence of dead realm marking for all used RADIUS server implementations and bad experience from summer 2006 led us to remove the second national RADIUS server from the Czech *eduroam* infrastructure. With a good service contract assuring us of a quick replacement of failed components we were able to assure that the service would be restored in a few hours. But 4 or more hours is still quite a long time and also the impossibility to do regular maintenance in working hours made us think about a high availability solution.

2 Terms explanation

2.1 High availability – HA

High availability is a system design that assures an availability degree of a service which is not possible to be achieved with a single component. For more info see the article High Availability³ at Wikipedia.

2.2 Gratuitous ARP packet

Gratuitous ARP packet is a method how to announce MAC address in a local network. HA cluster uses gratuitous packet for redirecting traffic from one node to another. For more info see the article Address Resolution Protocol⁴ at Wikipedia.

2.3 RadSec

RadSec is a protocol for transporting RADIUS datagrams over TCP and TLS. For more info see the article RadSec⁵ at Wikipedia

3 Analysis

The HA RADIUS server is implemented as a cluster consisting of two nodes. Both nodes are using the same hardware – DELL PowerEdge R410. The first node is installed on the primary CESNET server room (POP1) and the second is in a backup locality (POP2).

For a historical reason the Czech national RADIUS proxy server uses an IP address from a server segment which depends on the router r1 located in POP1. It means that in case of failure of the router r1 in POP1, the second node located in POP2 will not be able to reach the Internet. It is possible to resolve this problem by readdressing national RADIUS proxy server, but it would be a very complicated task, because its IP address is registered in many systems in connected organisations.

³ http://en.wikipedia.org/wiki/High_availability

⁴ http://en.wikipedia.org/wiki/Address_Resolution_Protocol#ARP_announcements

⁵ <http://en.wikipedia.org/wiki/RadSec>

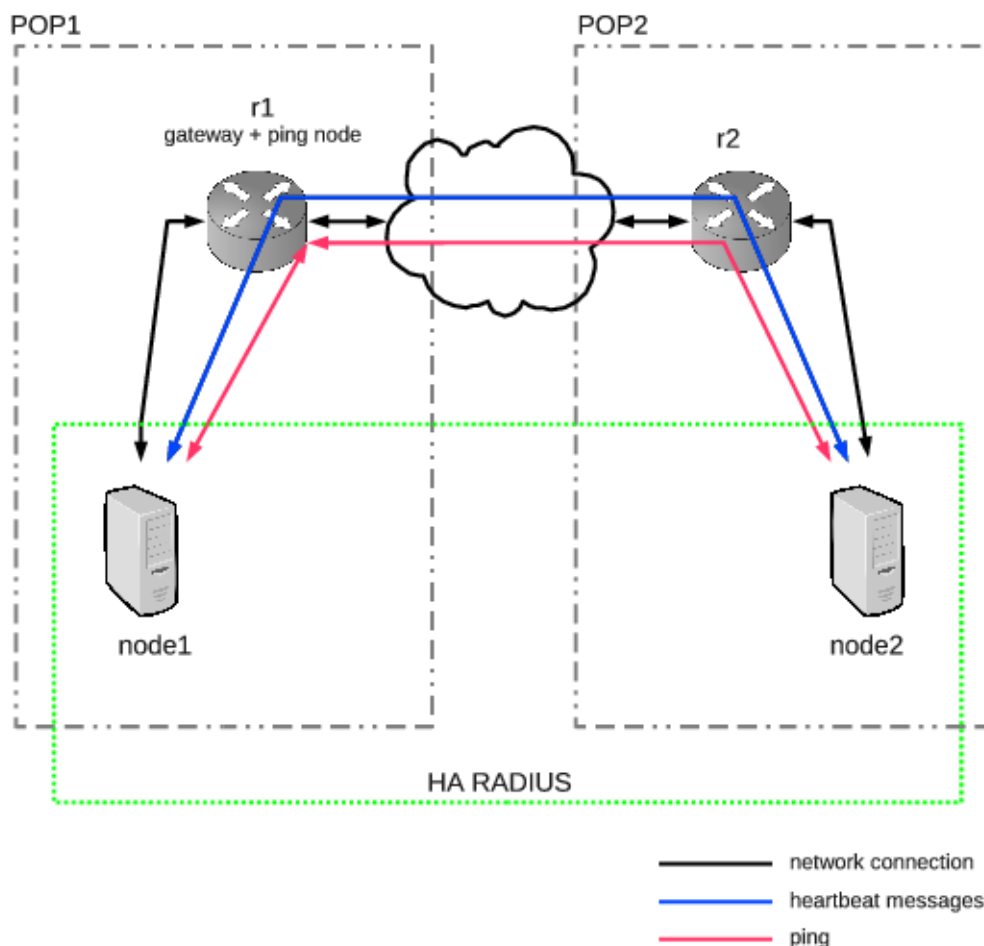


Figure 1. HA RADIUS server schema

3.1 Configuration files of RADIUS components

Configuration files of software components (FW, IPSec, Radiator), together creating national RADIUS proxy server are created by our script. The script is using information stored in CESNET CAAS (a LDAP directory). Generating new configuration files is done when a Czech *eduroam* federation member requires a change in his setup. Or in case a new organisation becomes a member of the federation. Except these moments the configuration files are static. Configuration files are built by a script invoked by an administrator. To simplify the setup, the script for building configuration files is installed only on the first node. The administrator must be aware of this and manually synchronise configuration files to the second node. In case the second node is in the active role, administrator must also restart the components whose configuration files have been changed. From the point of view of the Czech *eduroam* federation both nodes are equivalent.

3.2 Status data

All status data needed for providing services on the national RADIUS proxy server are quite easy to recover. If a loss of status data happens during an ongoing user authentication, then the user usually must repeat his authentication attempt. It

is usually done by the user's supplicant which does it automatically so the user typically does not notice anything.

Review of status data kept on the national RADIUS proxy server:

- SA of IPsec connections:
 - valid about 18 hours
 - to negotiate new ones requires typically 30 seconds, maximum 5 minutes
- TCP RadSec connections:
 - permanently established until Radiator restarts (maximum 24 hours or a new CRL)
 - re-established typically in less than one second
- status data for dead realm marking feature:
 - kept in the operational memory of a Radiator process
 - being gathered during the normal run-time
 - if lost, and the first RADIUS server of an organisation is down, it might cause the timeout of the first authentication attempt
- status data about proxied packets:
 - kept in the operational memory of a Radiator process
 - without them the response to the Access-Request packet cannot be delivered
 - when lost, ongoing authentications will fail

The status data review shows that there is no need to take any extra precaution on them. From HA point of view the national RADIUS proxy server is stateless. This fact significantly simplifies implementation of the cluster. The only information which must be kept in sync between both nodes are configuration files and software – both is static.

3.3 Failure reaction time

The HA Linux allows to implement clusters with a reaction time less than one second, in case of the national RADIUS proxy server it is not necessary to react so fast. The cluster is geographically separated and because of that short network outages might happen. Also re-negotiation of IPsec SA can take longer time. Reaction time in minutes is perfect for our purpose.

4 HA RADIUS server implementation

The HA RADIUS server is built up on the heartbeat daemon of Linux-HA⁶ project.

Services provided by a cluster are in HA terminology called resources. Even the IP address of the cluster is a resource. All resources are managed by the Cluster Resource Manager (CRM). Classical CRM was replaced by Peacemaker project which allows to maintain quite a complex dependencies between resources. To keep our implementation as simple and robust as possible we decided to use classical CRM from heartbeat.

⁶ <http://www.linux-ha.org/>

Resource control is done by the standard Linux system init scripts. They are not called directly, but via wrapper scripts. They are stored in the directory `/etc/ha.d/resource.d/`, these scripts call other scripts stored in the directory `/usr/lib/ocf/resource.d/heartbeat/`. These scripts finally call the corresponding system init script.

Resources of HA RADIUS are described in the file `/etc/ha.d/haresources`:

```
node-1 195.113.xxx.xxx on-off firewall ipsec-policies racoon radiator
```

The basic resource is the IP address used by the HA RADIUS server to communicate with other RADIUS server. The node, which takes over the active role in the cluster, activates the IP address on its virtual interface `eth0:0` and announces its active role by Gratuitous ARP⁷ packet. Since this moment all traffic is re-routed to newly activated node.

The `arp send` command is used to send Gratuitous ARP packet:

```
arp send -U -i 195.113.187.22 eth0 -c 1
```

4.1 Resource “on-off”

Resource “on-off” indicates whether the node is active = ON or inactive = OFF. The state OFF is indicated by the presence of the file `/etc/OFFLINE`. The presence of the file is tested by components of the national RADIUS proxy server. In case the file is present all components refuse to start. The file is created after a reboot of a node and if the node gains the active role in the cluster, it removes the file and launches other resources.

The file duplicates information (whether the node is or is not active) which is kept by heartbeat daemon. We decided to go this way, to be able to remove heartbeat daemon and manually control both nodes of the cluster.

4.2 Resource “firewall”

This resource controls firewall rules of a node. When the node becomes active, all FW rules are preventively re-set. When the node becomes passive, the FW rules remains intact.

4.3 Resource “ipsec-policies”

This resource controls IPSec policies in Linux kernel on a node. When the node becomes active, all policies are preventively re-set. When the node becomes passive, the IPSec policies remain intact. They never match because they are linked to the IP address which the node no longer controls.

⁷ http://en.wikipedia.org/wiki/Address_Resolution_Protocol#ARP_announcements

4.4 Resource “racoon”

This resource controls racoon daemon. Racoon is an IKE daemon responsible for negotiation of SA for IPSec. When a node becomes active, racoon is launched. When the node becomes passive, racoon is terminated.

4.5 Resource “radiator”

This resource controls radiator daemon. Radiator is an implementation of RADIUS server, it provides primary function of the cluster. When a node becomes active, radiator is launched. When the node becomes passive radiator is terminated.

4.6 Heartbeat daemon configuration

Configuration of the first node:

```
logfacility local0
initdead 180
keepalive 1
deadtime 180
warntime 20
udpport 694
ucast eth0 195.113.xxx.xxx
auto_failback off
ping 195.113.xxx.1
node node-1
node node-2
```

Configuration of the second node is identical except the

```
ucast eth0 195.113.xxx.xxx
```

option. This option defines IP address of the first node. In configuration of the first node there is the IP address of the second node. If multicast is used the configuration of both nodes is identical. We decided to use unicast not to abuse other systems, connected to the same network segment, by service traffic of the cluster.

The `initdead` option defines how long the daemon is inactive after start. This feature provides time for system stabilisation after reboot. The `keepalive` option defines how often the daemon checks status of the cluster nodes. `warntime` defines how long the daemon waits before it logs warning about possibly failed node. Finally `deadtime` defines how long the previously active node must be unreachable before the passive node tries to gain control of the cluster.

The `ping` option defines so called ping node⁸. The ping node is not a member of the cluster (it does not provide any services to the public) but it acts as arbitrator in disputes about the active role in the cluster. If the passive node does not receive any messages from active node, it assumes that the active node has failed but it is

⁸ <http://www.linux-ha.com/PingNode/>

permitted to gain cluster control only if it receives replies from the ping node. The ping node is a default gateway.

The `auto_failback` option determines whether a resource will automatically fail back to its “primary” node, or remain on the node where it is just running, until the node fails, or an administrator intervenes.

4.7 Configuration files synchronisation

We developed our own tool for synchronising the configuration files between nodes. We decided to do our own implementation because commonly used tools like CVS or modern Unison or Dropbox are not able to work on complete FS. They usually require that all configuration files are placed into one or into a limited set of directories and that only soft links are made into final destinations.

Our script needs a list of files with full path names to be synchronised. An administrator who executes the script must have installed his SSH key on destination node.

Backup copies of older revisions of configuration files are kept in files named according to the template `filename-%Y%m%d-%H%M%S`. It would be useful to enhance our script to be able to restart services whose configuration files were changed during the synchronisation.

Our script is available online⁹.

5 Operational experience

Czech national RADIUS proxy server is running in HA cluster mode since the end of summer 2009. So far our experience is very positive. The only problem we had was a fitful packet loss. Network interfaces of both nodes did not report any problems but heartbeat daemon was frequently complaining about lost or delayed heartbeat packets. The cause of the problem was in the RADIUS traffic itself. Heartbeat daemon uses UDP as a transport for its messages. Radiator also uses UDP as transport mechanism (except the hosts who are using RadSec). Fitful RADIUS packet bursts were causing overflow of Linux kernel UDP buffer and resulted into heartbeat packet loss. Default Linux kernel uses UDP buffer of 128 KB. Before deploying HA we were using 8 MB, but to avoid heartbeat packet loss we had to raise the UDP buffer up to 16 MB.

The size of the Linux kernel UDP buffer is controlled by the command `sysctl -w "net.core.rmem_max=16777216"`, to make this setting permanent it must be written into the file `/etc/sysctl.conf`. Lost UDP packets are counted by Linux kernel and the number is printed as the field “packet receive errors” in output of the command `netstat -su`.

Another change we had to make was the modification of firewall rules to allow quick re-establishment of RadSec connections. We are using stateful firewall implemented with iptables. Our iptables rules are using priority chains to process packets associated with established TCP connections. The rest of packets must pass much

⁹ <http://www.eduroam.cz/config-sync/>

complex rules. If a packet comes from a registered IP address and if the packet has the state NEW, then it serves for establishing new TCP connection. This type of packet is accepted. Other packets are ignored (rule DROP) as unrelated. These rules complicated situation in case the active and the passive node changed. From the point of view of RADIUS servers connected to our cluster nothing changed. From the point of view of the new active node packets related to TCP connections established with the old active node were unrelated so the firewall of the new active node ignored them. Because of this re-establishing of the RadSec connections took too long – until TCP connection timeout. When we changed the rule DROP to REJECT, RadSec connections re-established immediately after switching between active and passive node.

Example of our firewall rules:

```
$IPT=iptables
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
...
$IPT -N KEEP_STATE
$IPT -F KEEP_STATE
$IPT -A KEEP_STATE -m state --state RELATED,ESTABLISHED -j ACCEPT
...
$IPT -N radsec
for HOST in $SRVC_RADSEC
do
    $IPT -A radsec -i $EXT -p tcp -m state --state NEW -s $HOST -j ACCEPT
done
$IPT -A radsec -i $EXT -j LOG --log-prefix "REJECT-radsec "
$IPT -A radsec -i $EXT -j REJECT
...
$IPT -A INPUT -i $LOOPBACK -j ACCEPT
$IPT -A INPUT -i $EXT -j KEEP_STATE
...
$IPT -A INPUT -i $EXT -p tcp -d $MY_RAD_IP --dport 2083 -j radsec
...
$IPT -A INPUT -j LOG -m limit --limit $LOG_LIMIT --log-prefix "D-IN "
$IPT -A INPUT -j DROP
```

Example of warnings of heartbeat packet loss and delays:

```
heartbeat: WARN: 1 lost packet(s) for [node-1] [1497002:1497004]
heartbeat: WARN: 9 lost packet(s) for [node-1] [1520245:1520255]
heartbeat: WARN: Late heartbeat: Node node-1: interval 25250 ms
```

Example of logs since the moment the passive node-1 has gained control of the cluster. We raised this situation by a shutdown of the node-2.

```
heartbeat: info: Received shutdown notice from node-2.
heartbeat: info: Resources being acquired from node-2.
heartbeat: info: acquire all HA resources (standby).
ResourceManager: info: Acquiring resource group:
    node-1 195.113.187.22 on-off firewall ipsec-policies racoon radiator
ResourceManager: info: Running
```

```
/etc/ha.d/resource.d/IPaddr 195.113.xxx.xxx start
IPaddr: INFO: Using calculated nic for 195.113.xxx.xxx: eth0
IPaddr: INFO: Using calc. netmask for 195.113.xxx.xxx: 255.255.255.0
IPaddr: INFO: eval ifconfig eth0:0 195.113.xxx.xxx
        netmask 255.255.255.0 broadcast 195.113.187.255
IPaddr: DEBUG: Gratuitous Arp for 195.113.xxx.xxx on eth0:0 [eth0]
IPaddr: INFO: Success
on-off: INFO: Resource is stopped
ResourceManager: info: Running /etc/ha.d/resource.d/on-off start
on-off: INFO: Success
firewall: INFO: Resource is stopped
ResourceManager: info: Running /etc/ha.d/resource.d/firewall start
firewall: INFO: Success
ipsec-policies: INFO: Resource is stopped
ResourceManager: info: Running /etc/ha.d/resource.d/ipsec-policies start
ipsec-policies: INFO: Success
racoon: INFO: Resource is stopped
ResourceManager: info: Running /etc/ha.d/resource.d/racoon start
racoon: INFO: Success
radiator: INFO: Resource is stopped
ResourceManager: info: Running /etc/ha.d/resource.d/radiator start
radiator: INFO: Success
heartbeat: info: Standby resource acquisition done [all].
mach_down: info: mach_down: nice_failback: foreign resources acquired
mach_down: info: mach_down takeover complete for node node-2.
heartbeat: info: mach_down takeover complete.
```