Relayd: a load-balancer for OpenBSD

Giovanni Bechis

giovanni@openbsd.org

University of Applied Sciences,
Vienna, Austria
May 5, 2012
what is relayd useful for?

- Reverse proxy
- SSL accelerated reverse proxy
- Transparent proxy with filtering capabilities
- Application redirector
- Load balancer
- Wan link balancer
a short story

- First imported in OpenBSD 4.1
- Initially it was called hoststated(8)
- Renamed to relayd(8) in OpenBSD 4.3
- Written by pyr® and reyk®
some relayd(8) features

- written with security in mind and based on imsg framework
- ipv4 and ipv6 capable
- carp(4) capable
- snmpd(8) integration
Relayd is divided in a main process and 3 different engines

- Parent process
- HCE: Host check engine
- PFE: Pf engine
- Relay engine
The parent process is the only one that runs with elevated privileges, it runs as 'root' to be able to handle:

- configuration files
- setup sockets
- external script execution (privileges will be dropped to _relayed user before ”execvp” function call)
- carp demotion requests
The Host Check Engine uses some methods to verify that the target host service is functional, before routing traffic to the host. It can use:

- icmp
- tcp
- ssl
- http/https
- external scripts
pf engine

The Packet Filter Engine allows integration with the OpenBSD Packet Filter.

- Creates and destroys PF rules
- Updates PF tables based on HCE notifications
This engine is responsible to filter and relay packets

- Creates listening sockets for services
- Filters protocols before relaying
reverse http proxy
reverse http proxy

table <web_hosts> { 10.0.0.1 }

interval 10
timeout 200
prefork 5
log updates

relay httpproxy {
    listen on 192.168.0.1 port 80

    forward to <web_hosts> port 80 check http "/" code 200
}
reverse http proxy

A script can be used to check the web server status

table <web_hosts> { 10.0.0.1 }

relay httpproxy {
  listen on 192.168.0.1 port 80

  forward to <web_hosts> port 80 \\ 
  check script "/scripts/chkweb.pl"
}

A script can be used to check the web server status ... or everything else

```perl
#!/usr/bin/perl -w

use Socket;

my $remote = $ARGV[0];
my $proto = getprotobynumber('tcp');
socket(Socket_Handle, PF_INET, SOCK_STREAM, $proto);
my $hport = 80; # Http port
my $sin = sockaddr_in($hport,inet_aton($remote));
if (connect(Socket_Handle,$sin)) {
    socket(Socket_Handle, PF_INET, SOCK_STREAM, $proto);
    my $mport = 11211; # Memcached port
    $sin = sockaddr_in($mport,inet_aton($remote));
    if (connect(Socket_Handle,$sin)) {
        exit 1;
    } else {
        exit 0;
    }
} else {
    exit 0;
}
```
http filters

Relayd in "reverse proxy" configuration can filter http requests

- Change or append http headers
- Filter http requests by checking http headers
- Filter http requests by checking url
http filters

http protocol "httpfilter" {

  # Return HTML error pages
  return error

  # allow logging of remote client ips to internal web servers
  header append "$REMOTE_ADDR" to "X-Forwarded-For"

  # URL filtering
  request path filter "articleid=*select*" \
    from "/module/article/article/article.asp"

  # close connections upon receipt
  header change "Connection" to "close"
}

http filters
http filters

Forbidden
rejcting request

OpenBSD relayd at www.example.com port 80
ssl accelerated reverse http proxy
ssl accelerated reverse http proxy

table <web_hosts> { 10.0.0.1 }

http protocol "httpfilter" {
    # close connections upon receipt
    header change "Connection" to "close"
    # SSL accelerator ciphers
    ssl { sslv3, tlsv1, ciphers "HIGH:!ADH", no sslv2 }
}

relay httpproxy {
    listen on 192.168.0.1 port 443 ssl
    protocol "httpfilter"
    forward to <web_hosts> port 80 check http "/" code 200
}
ssl accelerated reverse http proxy

Rsa certificate generation

openssl genrsa -out /etc/ssl/private/192.168.0.1:443.key 1024
openssl req -new -key /etc/ssl/private/192.168.0.1:443.key \
   -out /etc/ssl/private/192.168.0.1:443.csr

openssl x509 -req -days 365 \
   -in /etc/ssl/private/192.168.0.1:443.csr \
   -signkey /etc/ssl/private/192.168.0.1:443.key \
   -out /etc/ssl/192.168.0.1:443.crt

With the files 192.168.0.1:443.crt and 192.168.0.1:443.key in the right place relayd will do his job
transparent http proxy

Diagram:
- Lan
- Relayd
- Internet
transparent http proxy, relayd setup

http protocol "httpfilter" {
    # Return HTML error pages
    return error

    header change "Connection" to "close"

    # Block requests to unwanted hosts
    request header filter "*youtube.com*" from "Host"
    request header filter "*facebook.com*" from "Host"
}

relay httpproxy {
    listen on 127.0.0.1 port 8080
    protocol "httpfilter"
    forward to destination
}
application redirector
application redirector, relayd setup

table <srv> { 192.168.0.1, 192.168.0.2 }

redirect mysql {
    listen on 192.168.3.1 port 3306
    tag RELAYD
    sticky-address
    forward to <srv> port 3306 mode roundrobin check tcp
}
load balancer
load balancer

dns protocol "dnsfilter" {
    tcp { nodelay, sack, socket buffer 1024, backlog 1000 }
}

relay dnsproxy {
    listen on 127.0.0.1 port 8053

    protocol "dnsfilter"

    forward to <dns_servers> port 53 \  
    mode loadbalance check tcp
}
relayctl(8)

- relayctl is the software used to control relayd
- It can change many configurations at runtime
- It can be used to show many informations about our current relayd(8) setup
Some info for our "relay" setup

$ sudo relayctl show sessions
session 0:1 192.168.107.205:44159 -> :80       RUNNING
    age 00:00:01, idle 00:00:01, relay 1, pid 5613
$ sudo relayctl show hosts
Id   Type   Name           Avlblty Status
 1    table  web_hosts:80  active (3 hosts)
 1    host   10.0.0.1      100.00% up
       total: 12/12 checks
 2    host   10.10.10.22   100.00% up
       total: 12/12 checks
 3    host   10.10.10.33   100.00% up
       total: 12/12 checks
Some info for our "redirect" setup

$ sudo relayctl show summary

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Name</th>
<th>Avlblty</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>redirect</td>
<td>mysql</td>
<td>active</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>table</td>
<td>srv:3306</td>
<td>active (1 hosts)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>host</td>
<td>192.168.1.3</td>
<td>100.00% up</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>host</td>
<td>192.168.1.4</td>
<td>0.00% down</td>
<td></td>
</tr>
</tbody>
</table>
Pf interaction

$ sudo pfctl -a relayd/mysql -s rules
pass in quick on rdomain 0 inet proto tcp from any \
  to 192.168.1.5 port = 3306 flags S/SA \
  keep state (tcp.established 600) \
  tag RELAYD rdr-to <mysql> port 3306 \
  round-robin sticky-address
Both Munin and Nagios have plugins to check relayd health status.
questions ?