

Using OpenBSD relayd(8) as an Application Layer Gateway

EuroBSDCon 2023 • Coimbra, Portugal

Joel Carnat <joel@carnat.net>

Who's that guy?



User of a Terminal since late 90s.

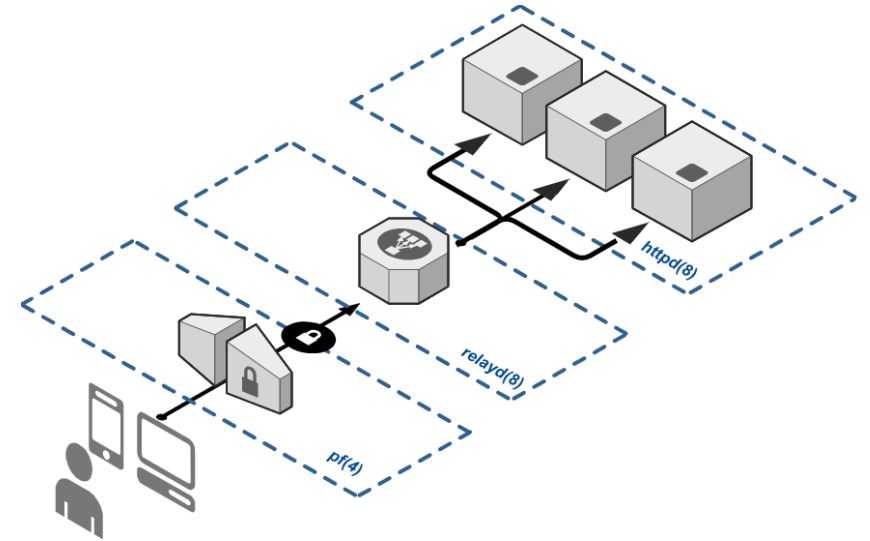
Freelance Technical Architect since 2015.

Self-hosting compulsive.

Blog about FOSS and OpenBSD at <https://www.tumfatig.net>.

What is relayd(8)?

- Multi-purpose daemon available on OpenBSD since 4.3*:
 - load-balancer.
 - application layer gateway.
 - transparent proxy.
- Capable of monitoring groups of hosts for high-availability.
- Operates as:
 - Layer 3 redirection via communication with pf(4).
 - Layer 7 relaying with application level filtering via itself.



* relayd was known as hoststated in OpenBSD 4.1

How to manage relayd(8)?

- *Read the man pages*

```
# man relayd
# man relayd.conf
# man relayctl
```

- *Configure the software*

```
# more /etc/examples/relayd.conf
# vi /etc/relayd.conf
```

- *Control the daemon*

```
# relayd -dvn
# rcctl enable relayd
# rcctl start relayd
# rcctl stop relayd
# relayctl command [argument ...]
```

Terminology

- *Macros: user-defined variables that can be used later on.*
- *Tables: host or a group of hosts defining traffic targets.*
- *Protocols: settings and filter rules for relays.*
- *Relays: layer 7 proxying instances.*

Simplest HTTP relay

A simple HTTP Reverse proxy.

- Define an HTTP PROTOCOL section.
- Define a RELAY section.
 - Listen on address and port.
 - Use the defined HTTP protocol.
 - Forward HTTP traffic to the servers.

```
http protocol www {  
    pass  
}  
  
relay www {  
    listen on 203.0.113.1 port 80  
    protocol www  
    forward to 192.0.2.10 port 80  
}
```

Better simple HTTP relay

A simple HTTP Reverse proxy providing reusable names in configuration and logging state changes and remote connections.

- Define and use MACROS.
- Configure GLOBAL CONFIGURATION.
 - Enable logging.
- Define and use TABLES.
- Update the PROTOCOL and RELAY sections.

```
# Macros -----
ext_addr="203.0.113.1"
webhost1="192.0.2.10"

# Global configuration -----
log state changes
log connection

# Tables -----
table <webhosts> { $webhost1 }

# Protocols & Relays -----
http protocol www {
    pass
}

relay www {
    listen on $ext_addr port 80
    protocol www

    forward to <webhosts> port 80
}
```

Encrypt HTTP relay using Transport Layer Security (TLS)

Secure communication and data transfer between the client and the website using HTTPS.

- Acquire a TLS certificate*.
- Install the TLS certificate
`/etc/ssl/private/relayd.example.key`
`/etc/ssl/relayd.example.crt`
- Define TLS PROTOCOL and RELAY sections.

```
# Macros -----
ext_addr="203.0.113.1"
webhost1="192.0.2.10"

# Global configuration -----
log state changes
log connection

# Tables -----
table <webhosts> { $webhost1 }

# Protocols & Relays -----
http protocol wwwtls {
    tls keypair relayd.example
}

relay wwwtls {
    listen on $ext_addr port 443 tls
    protocol wwwtls

    forward to <webhosts> port 80
}
```

*Out of relayd(8) scope. Use acme-client(1) and httpd(8).

Load balancing & Failover

Distribute incoming requests to several servers.

- Define a TABLE that references all the servers.
- Select a scheduling algorithms (aka MODE):
hash, loadbalance, random, roundrobin
source-hash.
- Select a health-checking method (aka CHECK):
no check, code, icmp, host, path, script, send
data expect pattern, tcp, tls

```
ext_addr="203.0.113.1"
whost1="192.0.2.11"
whost2="192.0.2.12"
whost3="192.0.2.13"
interval 5
table <webhosts> { $whost1, $whost2, $whost3 }

http protocol wwwtls {
    tls keypair relayd.example
}

relay wwwtls {
    listen on $ext_addr port 443 tls
    protocol wwwtls
    # l/b using roundrobin, no check
    # forward to <webhosts> port 80
    # l/b using source-IP, check HTTP return code
    forward to <webhosts> port 80 \
        mode loadbalance \
        check "/health-check" code 200
}
```

Fallback server(s) - automatic switch

Automatic reaction on server(s) outage:

- Switch service to secondary server pool.
- Display an incident status page rather than HTTP/5xx error pages.
- Display a static "be back soon" page while performing maintenance.
- Define a TABLE for primary server(s).
- Define a TABLE for fallback server(s).
- Define a primary FORWARD directive. Use a CHECK method.
- Define a fallback FORWARD directive.

```
ext_addr="203.0.113.1"
whost1="192.0.2.11"
whost2="192.0.2.12"
whost3="192.0.2.13"
interval 5
table <webhosts> { $whost1, $whost2 }
table <fallback> { $whost3 }

http protocol wwwtls {
    tls keypair relayd.example
}

relay wwwtls {
    listen on $ext_addr port 443 tls
    protocol wwwtls

    # l/b using round-robin, check HTTP return code
    forward to <webhosts> port 80 mode roundrobin \
        check http "/" code 200

    # switch service if all previous checks fail
    forward to <fallback> port 80
}
```

Fallback server(s) - manual switch

Managed operations on server(s) outage:

- Same as automatic switch.
- As part of a **Business Continuity Plan**, switch to remote | mutualized | resources limited | staging servers...
- Define a TABLE for primary server(s).
- Define a TABLE for fallback server(s).
Use the DISABLE attribute.
- Define a primary FORWARD directive.
Use a CHECK method.
- Define a fallback FORWARD directive.
Use a CHECK method.

```
ext_addr="203.0.113.1"
whost1="192.0.2.11"
whost2="192.0.2.12"
whost3="192.0.2.13"
whost4="192.0.2.14"
interval 5
table <webhosts>          { $whost1, $whost2 }
table <fallback> disable { $whost3, $whost4 }

http protocol wwwtls {
    tls keypair relayd.example
}

relay wwwtls {
    listen on $ext_addr port 443 tls
    protocol wwwtls

    # l/b using source-IP, check HTTP return code
    forward to <webhosts> port 80 mode loadbalance \
        check http "/" code 200

    # l/b using round-robin, check HTTP return code
    forward to <fallback> port 80 mode roundrobin \
        check http "/" code 200
}
```

Fallback server(s) - manual switch example

```
# relayctl show summary
Id      Type      Name                Avlblty Status
1       relay     wwwtls              active
1       table     webhosts:80        active (2 hosts)
1       host      192.0.2.11         100.00% up
2       host      192.0.2.12         100.00% up
2       table     fallback:80        disabled
```

- Primary hosts are up and running.
- Secondary hosts are disabled.

Service is UP.

Fallback server(s) - manual switch example

```
# relayctl show summary
Id      Type      Name                Avlblty Status
1       relay     wwwtls              active
1       table     webhosts:80        empty
1       host      192.0.2.11         95.56% down
2       host      192.0.2.12         95.56% down
2       table     fallback:80        disabled
```

- Primary hosts are down.
- Secondary hosts are disabled.

 Service is DOWN.

Fallback server(s) - manual switch example

```
# relayctl table enable 2
command succeeded

# relayctl show summary
```

Id	Type	Name	Avlblty	Status
1	relay	wwwtls		active
1	table	webhosts:80		empty
1	host	192.0.2.11	76.79%	down
2	host	192.0.2.12	76.79%	down
2	table	fallback:80		active (2 hosts)
3	host	192.0.2.13	100.00%	up
4	host	192.0.2.14	100.00%	up

- Primary hosts are down.
- Secondary hosts are enabled.

Service is UP.

Failback shall happen as soon as relayd detects a Primary host up.

Use `relayctl table disable 1` to prevent such an automatic failback.

Relaying multiple FQDNs*

Expose multiple hostnames using a single IP.

- Define a TABLE for each server pool.
- Reference every TLS server certificates enabling TLS Server Name Indication (SNI).
- Limit FQDNs using FILTER RULES.
- Define FORWARD directives to map FQDNs with TABLES.

```
(...)  
table <blog> { $whost1, $whost2 }  
table <cloud> { $whost3 }  
  
http protocol wwwtls {  
    tls keypair blog.example  
    tls keypair nextcloud.example  
  
    block  
    pass request header "Host" value "blog.example" \  
        forward to <blog>  
    pass request header "Host" value "cloud.example" \  
        forward to <cloud>  
}  
  
relay wwwtls {  
    listen on $ext_addr port 443 tls  
    protocol wwwtls  
    forward to <blog> port 80 mode roundrobin \  
        check http "/" code 200  
    forward to <cloud> port 80  
}
```

*AKA Apache Virtual Host, AKA nginx server blocks.

Relaying multiple pathnames*

Design reaction rules (allow, deny, forward...) depending on URL path.

- Define a TABLE for each server pool.
- Limit pathnames using FILTER RULES.
- Define FORWARD directives to map pathnames with TABLES.

```
(...)  
table <blog> { $whost1, $whost2 }  
table <cloud> { $whost3 }  
  
http protocol wwwtls {  
    tls keypair relayd.example  
  
    block quick path "/cgi-bin*"  
    block quick path "/wp-admin*"  
    pass quick path "/nextcloud/*" forward to <cloud>  
    pass request forward to <blog>  
}  
  
relay wwwtls {  
    listen on $ext_addr port 443 tls  
    protocol wwwtls  
  
    forward to <blog> port 80 mode roundrobin \  
        check http "/" code 200  
    forward to <cloud> port 80  
}
```

*AKA Apache location directive, AKA nginx location blocks.

Solving problems with HTTP headers

😞 Software like Baikal, Mastodon or SearxNG refuse to serve unencrypted content.

😏 Pass the *X-Forwarded-Proto* HTTP header to confirm communication is secured using TLS.

```
(...)  
http protocol wwtls {  
  tls keypair blog.example  
  tls keypair nextcloud.example  
  
  block  
  pass request header "Host" value "blog.example" forward to <blog>  
  pass request header "Host" value "cloud.example" forward to <cloud>  
  
  match request header set "X-Forwarded-Proto" value "https"  
}  
(...)
```

Solving problems with HTTP headers

 Software add too many information in the HTTP headers.

 Remove leaking HTTP headers from chatterbox software.

```
(...)  
http protocol wwtls {  
  tls keypair relayd.example  
  
  pass quick path "/nextcloud/*" forward to <cloud>  
  pass request          forward to <blog>  
  
  match response header remove "X-Powered-By"  
  match response header set "Server" value "Microsoft-IIS/8.5"  
}  
(...)
```

 Fool script kiddies or deal with buggy clients by replacing a existing HTTP header.

Solving problems with HTTP headers

😞 Software don't bother about *security and privacy*..

😏 Add HTTP headers that helps protecting the user.

```
(...)  
http protocol wwwtls {  
    tls keypair relayd.example  
  
    pass quick path "/nextcloud/*" forward to <cloud>  
    pass request          forward to <blog>  
  
    match response header set "X-XSS-Protection"      value "1; mode=block"  
    match response header set "X-Content-Type-Options" value "nosniff"  
    match response header set "Permissions-Policy"    value "accelerometer=(),  
ambient-light-sensor=(),autoplay=(),camera=(),encrypted-media=(),  
focus-without-user-activation=(),geolocation=(),gyroscope=(),  
magnetometer=(),microphone=(),midi=(),payment=(),picture-in-picture=(),  
speaker=(),sync-xhr=(),usb=(),vr=()"  
}  
(...)
```

🌐 Check your Web application using tools like [Mozilla Observatory](#) or [Nextcloud Security Scan](#).

Log management

By default, `relayd(8)` logs

- are sent to `syslogd(8)`
- appear in `/var/log/{daemon,messages}`
- are not all debug friendly.

```
Sep 17 14:31:46 ebsdc relayd[75340]: startup
Sep 17 14:31:46 ebsdc relayd[87221]: adding 2 hosts from table blog:80
Sep 17 14:31:46 ebsdc relayd[87221]: adding 1 hosts from table cloud:80 (no check)
Sep 17 14:31:56 ebsdc relayd[87221]: relay wwwtls, session 1 (1 active), 0,
    203.0.113.1 -> 127.0.0.1:80, done, GET -> 127.0.0.1:80;
```

Log management

Get relayd(8) logs in a dedicated log file by configuring syslogd(8).

```
# touch /var/log/relayd

# vi /etc/syslog.conf
!!relayd
*. * /var/log/relayd
!*
(...)

# vi /etc/newsyslog.conf
(...)
/var/log/relayd root:_relayd 640 7 * $D0 ZB

# rcctl restart syslogd
```

Log management

Get more HTTP details in relayd(8) logs by using FILTER RULES.

```
http protocol wwwtls {
  tls keypair relayd.example

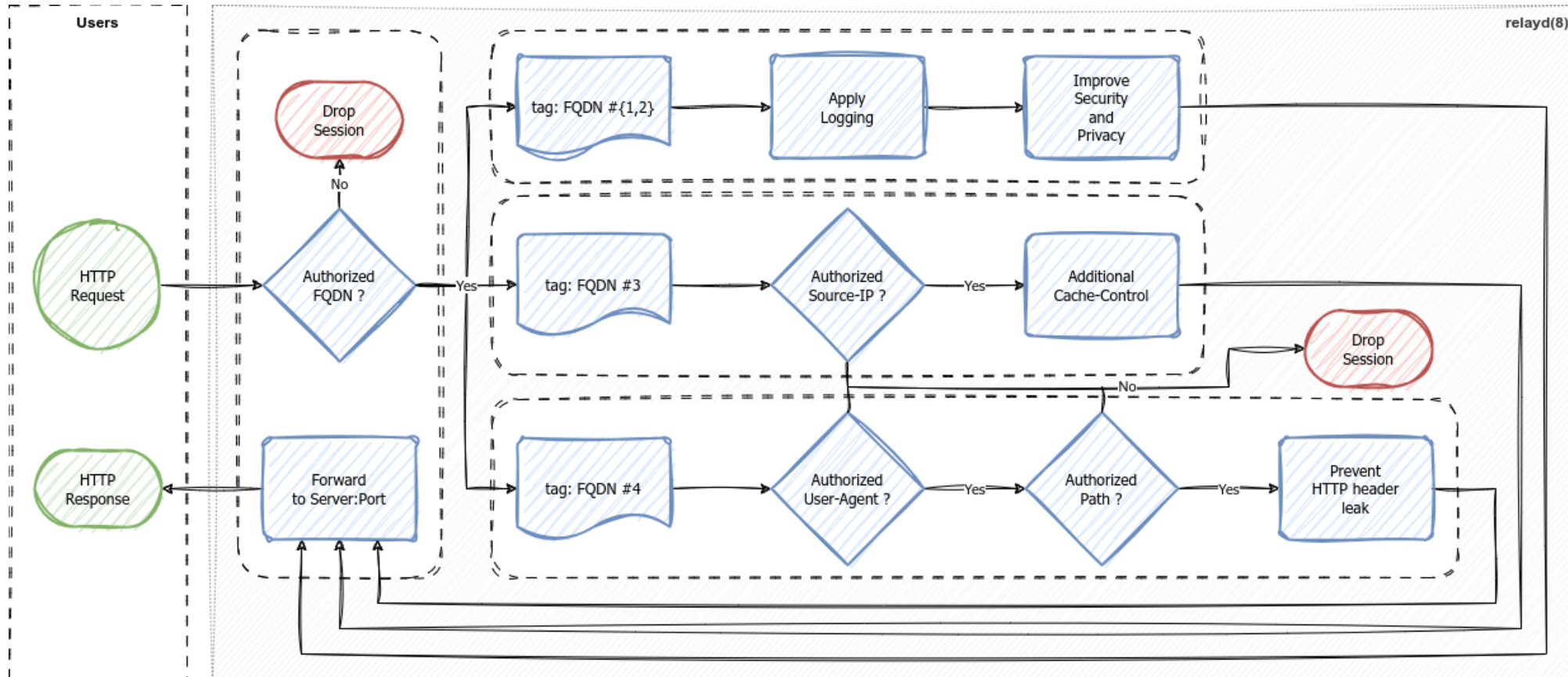
  match url log
  match header log "Host"
  match header log "User-Agent"
  match response header log "Content-Type"
  match response header log "Content-Length"

  pass quick path "/nextcloud/*" forward to <cloud>
  pass request                forward to <blog>
}
```

```
Sep 17 14:35:12 ebsdc relayd[34137]: relay wwwtls, session 1 (1 active), 0,
    203.0.113.1 -> 127.0.0.1:80, done,
    [blog.example/about] [Host: blog.example] [User-Agent: curl/8.2.0]
    GET -> 127.0.0.1:80 {Content-Type: text/html} {Content-Length: 41};
```

Conditional filtering

Using TAGS and INCLUDES to perform *different computation and actions depending on whether or not conditions evaluate to true or false.*



Conditional filtering: `/etc/relayd-ssg.conf`

- Use `TAG` to mark connections matching filter rules.
- Use `TAGGED` to match marked connections.

```
# Mark using hostnames
match request header "Host" value "www.example"    tag "ssg"
match request header "Host" value "blog.example"   tag "ssg"

# Apply additional logging
match header log "Host"          tagged "ssg"
match header log "User-Agent"    tagged "ssg"
match url    log                  tagged "ssg"

# Improve Security and Privacy
match response tagged "ssg" header set \
    "Strict-Transport-Security" value "max-age=31536000; includeSubDomains; preload"
match response tagged "ssg" header set \
    "X-XSS-Protection" value "1; mode=block"
match response tagged "ssg" header set \
    "X-Content-Type-Options" value "nosniff"
```


Conditional filtering: `/etc/relayd-nextcloud.conf`

- Use `TAG` to mark connections matching filter rules.
- Use `TAGGED` to match marked connections.

```
# Mark using hostname
match request header "Host" value "cloud.example" tag "nextcloud"

# Block User Agents
block request quick tagged "nextcloud" header "User-Agent" value "Googlebot/*"
block request quick tagged "nextcloud" header "User-Agent" value "YandexBot/*"

# Only allow "admin" path from specific subnet
match request url "cloud.example/admin/" tag "forbidden"
match request from 192.0.2.0/24 url "cloud.example/admin/" tag "nextcloud"

# Don't let version leak via HTTP header
match response tagged "nextcloud" header remove "Server"
```

Conditional filtering: `/etc/relayd-grafana.conf`

- Use `TAG` to mark connections matching filter rules.
- Use `TAGGED` to match marked connections.

```
# Mark using client source IP and path
match request from 192.0.2.0/24 url "metrics.example/" tag "grafana"
match request from 198.51.100.8/32 url "metrics.example/" tag "grafana"

# Overwrite caching
match request tagged "grafana" path "*.css" tag "g-cache"
match request tagged "grafana" path "*.js" tag "g-cache"
match request tagged "grafana" path "*.png" tag "g-cache"

match response tagged "g-cache" header set "Cache-Control" value "max-age=86400"
```

Conditional filtering: `/etc/relayd.conf`

- Use `INCLUDE` to dispatch filter rules in dedicated configuration files.
- Use `TAG` to mark connections matching filter rules.
- Use `TAGGED` to match marked connections.

```
table <blog>    { $whost1, $whost2 }
table <cloud>   { $whost3 }
table <grafana> { $whost4 }

http protocol wwwtls {
    tls keypair www.example
    tls keypair cloud.example
    tls keypair metrics.example

    block

    include "/etc/relayd-ssg.conf"
    include "/etc/relayd-nextcloud.conf"
    include "/etc/relayd-grafana.conf"

    pass request tagged "ssg"          forward to <blog>
    pass request tagged "nextcloud"    forward to <cloud>
    pass request tagged "grafana"      forward to <grafana>
    pass request tagged "g-cache"      forward to <grafana>
}
```

Thank you!
Any questions?

See you on    .