Rspamd - neue Konzepte im Antispam
[SLAC 2019]
Carsten Rosenberg <c.rosenberg@heinlein-support.de>

RSPAMD

and

a new concept of Anti-Spam

- **Heinlein Support**
  - IT-Consulting und 24/7 Linux-Support mit ~35 Mitarbeitern
  - Eigener Betrieb eines ISPs seit 1992
  - Täglich tiefe Einblicke in die Herzen der IT aller Unternehmensgrößen
- 24/7-Notfall-Hotline: 030 / 40 50 5 - 110
  - Spezialisten mit LPIC-2 und LPIC-3
  - Für alles rund um Linux & Server & DMZ
  - Akutes: Downtimes, Performanceprobleme, Hackereinbrüche, Datenverlust
  - Strategisches: Revision, Planung, Beratung, Konfigurationshilfe

Rspamd and the

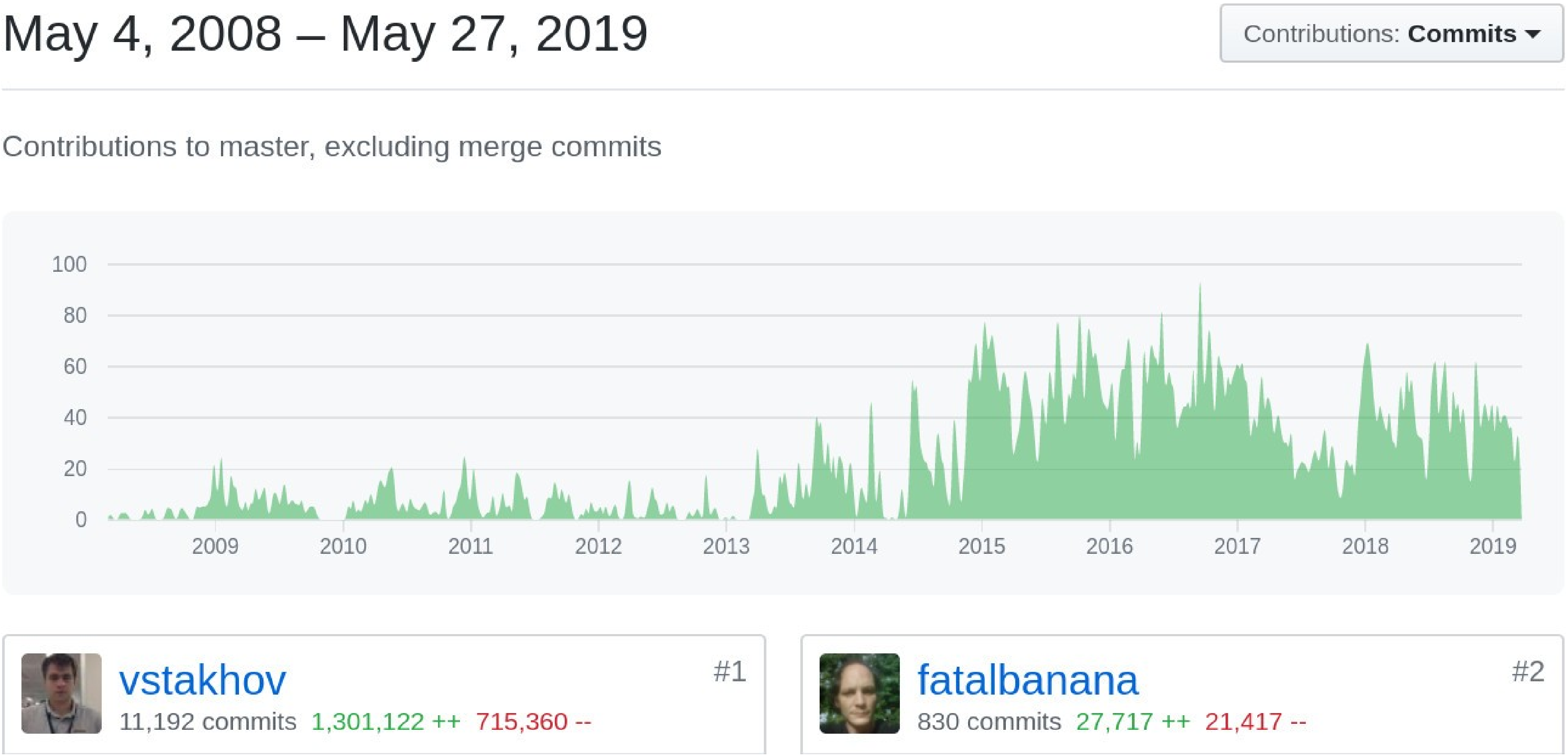
⌘ + Q

Problem

Rspamd Development

- Apache 2.0 License
- Developed directly in a Github repository
- Agile development method
- Growing test framework
- Automatic build process for every commit
- Fast integration of community PR's / patches
- Repositories for
  - Rspamd source code
  - Documentation → https://rspamd.com
  - Maps e.g. Whitelists, Mime Types, Redirectors, Freemail ...

Rspamd Development



May 4, 2008 – May 27, 2019

Contributions: **Commits** ▾

Contributions to master, excluding merge commits

vstakhov #1
11,192 commits 1,301,122 ++ 715,360 --

fatalbanana #2
830 commits 27,717 ++ 21,417 --

Rspamd is ...

- → Rspamd is (not) an Amavis / Spamassassin replacement
- → Rspamd is a Lua framework for mail processing
- → Rspamd is written in C and Lua

Rspamd - a Lua Framework for Mail

→ Mail UTF8 conversion, normalization and processing

→ Functions to provide mail object information (is_html)

→ Functions for accessing headers, mime_parts, aggregated data (e.g. urls, received headers)

→ Functions for modifying headers or body

→ Maps processing (Databases / Lists)

Rspamd – a Lua Framework for Mail

→ Libraries for TCP, UDP, Redis, DNS, HTTP, SMTP client connections

→ Signing & verifying signatures

→ Configuration processing and merging

→ Optimized regular expression progressing

→ Redundancy and Loadbalancing for network connections

→ Statistical text processing

→ Functions for HTML processing or Expression decisions

→ Helper Functions for several generic tasks

Rspamd - the code

→ Major part of the framework is written in C

→ Newer framework libraries are often written in Lua

→ Anti-Spam modules mainly written in Lua

→ Many optimizations e.g. Hyperscan, PCRE, LuaJIT, AST ...

Rspamd - Lua API Reference

- Documentation: https://rspamd.com/doc/lua/
- Libraries for Task, Config, Maps, Redis ...
- Can be used in modules and short Lua functions
- Lua functions can be also used when defining some config options (e.g. Bayes learn_condition, autolearn...)

```
-- Per user mail address blacklist
rspamd_config.PER_USER_WL = {
  callback = function(task)
     local rspamd_logger = require "rspamd_logger"
     local lua_redis = require "lua_redis"
     local redis_params = lua_redis.parse_redis_server('multimaps')
     local prefix = 'RS_UWL_'
     local rcpts = task:get_recipients()
     local sender = task:get_from()   -- Normally SMTP From
     local sender_address = sender[1]['addr']

     local function redis_get_cb(err, data)
       if err ~=nil then rspamd_logger.errx(task, 'redis_get_cb received  error: %1', err)
       end
       if #data[2] > 0 then
         task:set_pre_result('accept', 'ACCEPT by Per user Whitelist.')
         return true
       end
     end

     for _,rcpt in ipairs(rcpts) do
       local ret = lua_redis.redis_make_request(task,
         redis_params, -- connect params
         nil, -- hash key
         false, -- is write
         redis_get_cb, --callback
         'SSCAN', -- command
         {  prefix..rcpt['addr'], 0, 'MATCH', sender_address, 'COUNT', '1000' } -- arguments
       )
     end

    return false
  end,
  score = 1.0,
  group = 'per_user_config',
  description = 'Per user sender address whitelist',
  type = prefilter,
  priority = 2
}
```

```
# Bayes Configuration

classifier "bayes" {

...

    per_user = <<EOD
return function(task)
    local rcpt = task:get_recipients(smtp)

if rcpt then
    one_rcpt = rcpt[1]
    if one_rcpt['domain'] then
        return one_rcpt['domain']
    end
end

return nil
end
EOD

...

}
```

How to start with Rspamd

- → Forget what you think to know about Anti-Spam systems
- → Read the Quick Start *and* the FAQ
- → Have a look at UCL, Workers, Redis configuration
- → Have a look at the Modules pages
- → Use the current stable from rspamd.com

How to start with Rspamd

- If you have questions after RTFM
  - Mailling List: https://lists.rspamd.com
  - Telegram: http://t.me/rspamd
  - IRC: #rspamd at freenode
- If you really think you have found a bug:
  - https://github.com/rspamd/rspamd/issues
- If you have found a wrong or vague documentation page
  - Hit the „Edit Page“ on any documentation site
  - https://github.com/rspamd/rspamd.com/
  - Fork, Improve and make a Pull Request (https://guides.github.com/activities/forking/)

Rspamd – Distribution packages

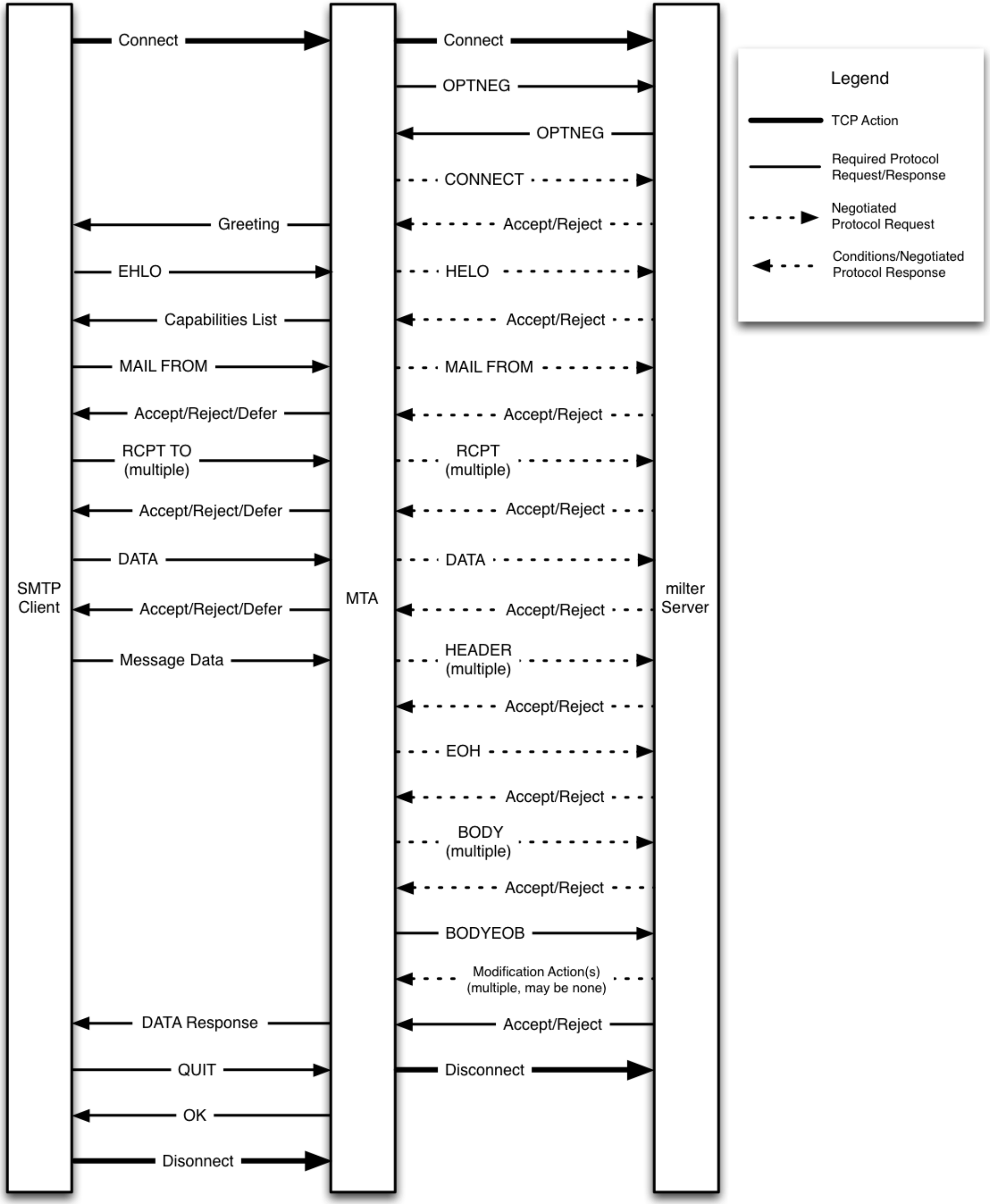
- Official Rspamd packages directly provided for
  - CentOS / Fedora
  - Debian / Ubuntu
- - Community Builds (external):
  - Alpine Linux
  - Arch Linux
  - Gentoo Linux
  - OpenSUSE
  - Void Linux
- Also BSD and MacPorts

Rspamd - general architecture

- Several worker types: Proxy, Normal (Scanner), Controller, Fuzzy
- Milter interface (Proxy Worker)
- HTTP Rest API (Scanner and Controller)
- Redis as temporary and persistent database
- Predefined rules and maps

Apropros: Milter Protocol

- Introduced by Sendmail
- Full Postfix support since 2.6 (Milter Version 6)
- Milter protocol is like Proxy-Filter + Postfix PDP
- A Milter program is a silent listener to the SMTP communication
- But it is able give a suggestion to every single SMTP command
  - REJECT / TEMPFAIL / DISCARD
  - CONTINUE
  - ACCEPT
- At End of Data / End of Mail – the Milter program is able to add headers and modify the body



SMTP Client

MTA

milter Server

Connect

Greeting

EHLO

Capabilities List

MAIL FROM

Accept/Reject/Defer

RCPT TO (multiple)

Accept/Reject/Defer

DATA

Accept/Reject/Defer

Message Data

DATA Response

QUIT

OK

Disonnect

Connect

OPTNEG

OPTNEG

CONNECT

Accept/Reject

HELO

Accept/Reject

MAIL FROM

Accept/Reject

RCPT (multiple)

Accept/Reject

DATA

Accept/Reject

HEADER (multiple)

Accept/Reject

EOH

Accept/Reject

BODY (multiple)

Accept/Reject

BODYEOB

Modification Action(s) (multiple, may be none)

Accept/Reject

Disconnect

Legend

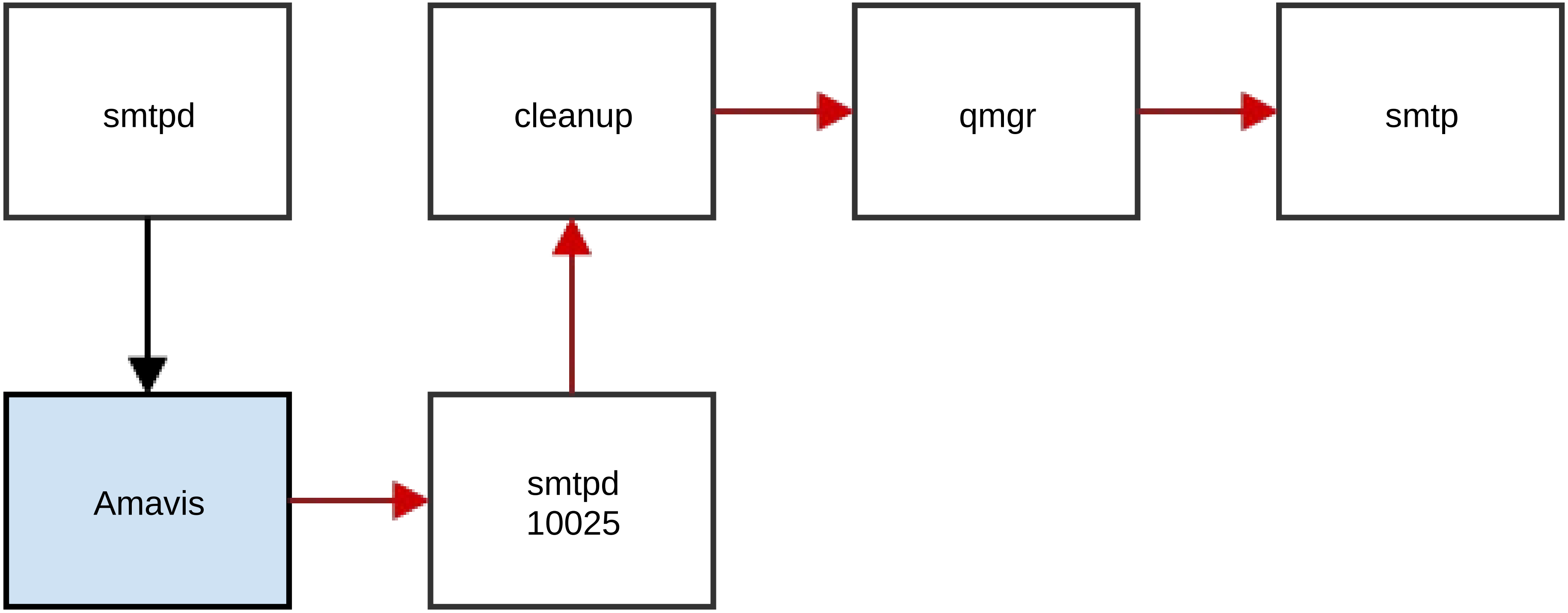
TCP Action

Required Protocol Request/Response

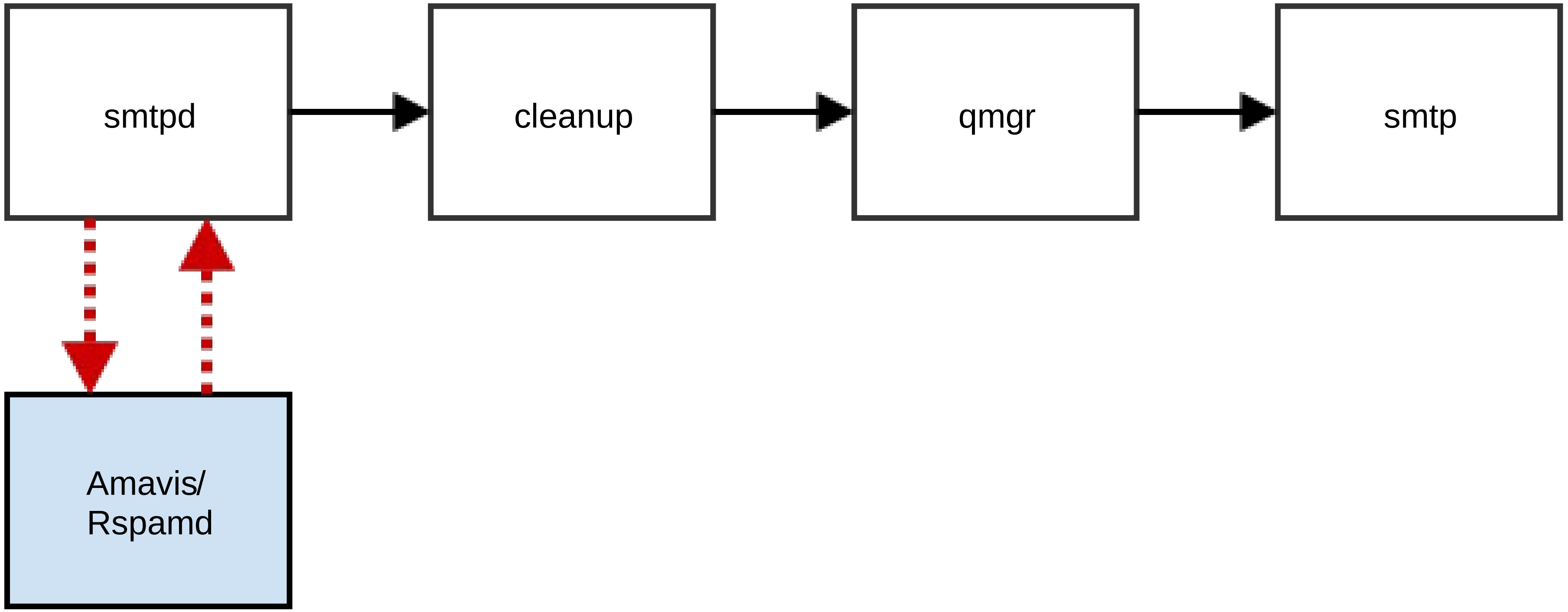
Negotiated Protocol Request

Conditions/Negotiated Protocol Response

Postfix SMTPD Proxy Filter



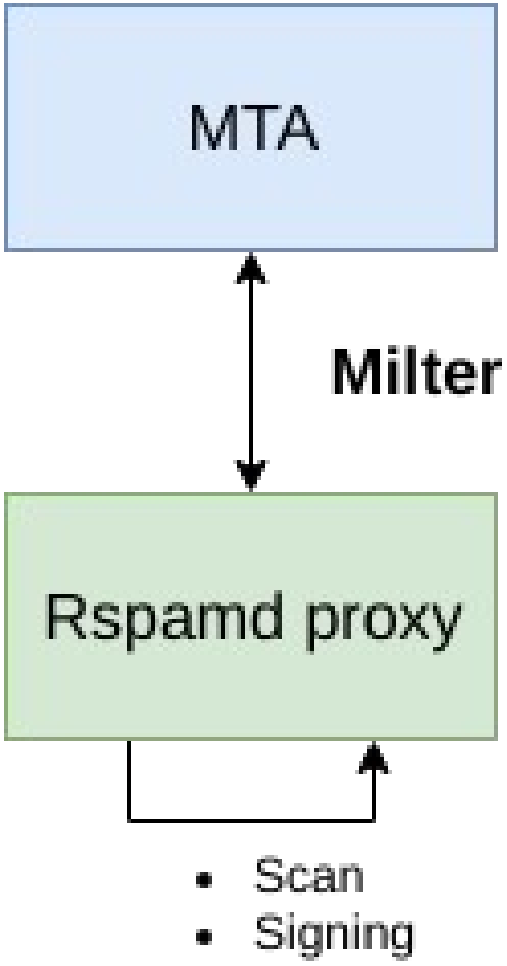
Postfix Milter



Rspamd - architecture on single systems

- → Worker Proxy deals with Milter communication
- → Worker Normal (Scanner) is processing the mail
- → Worker Controller provides HTTP API and Webinterface
- → Proxy calls Scanner directly
- → Single (local) Redis instance

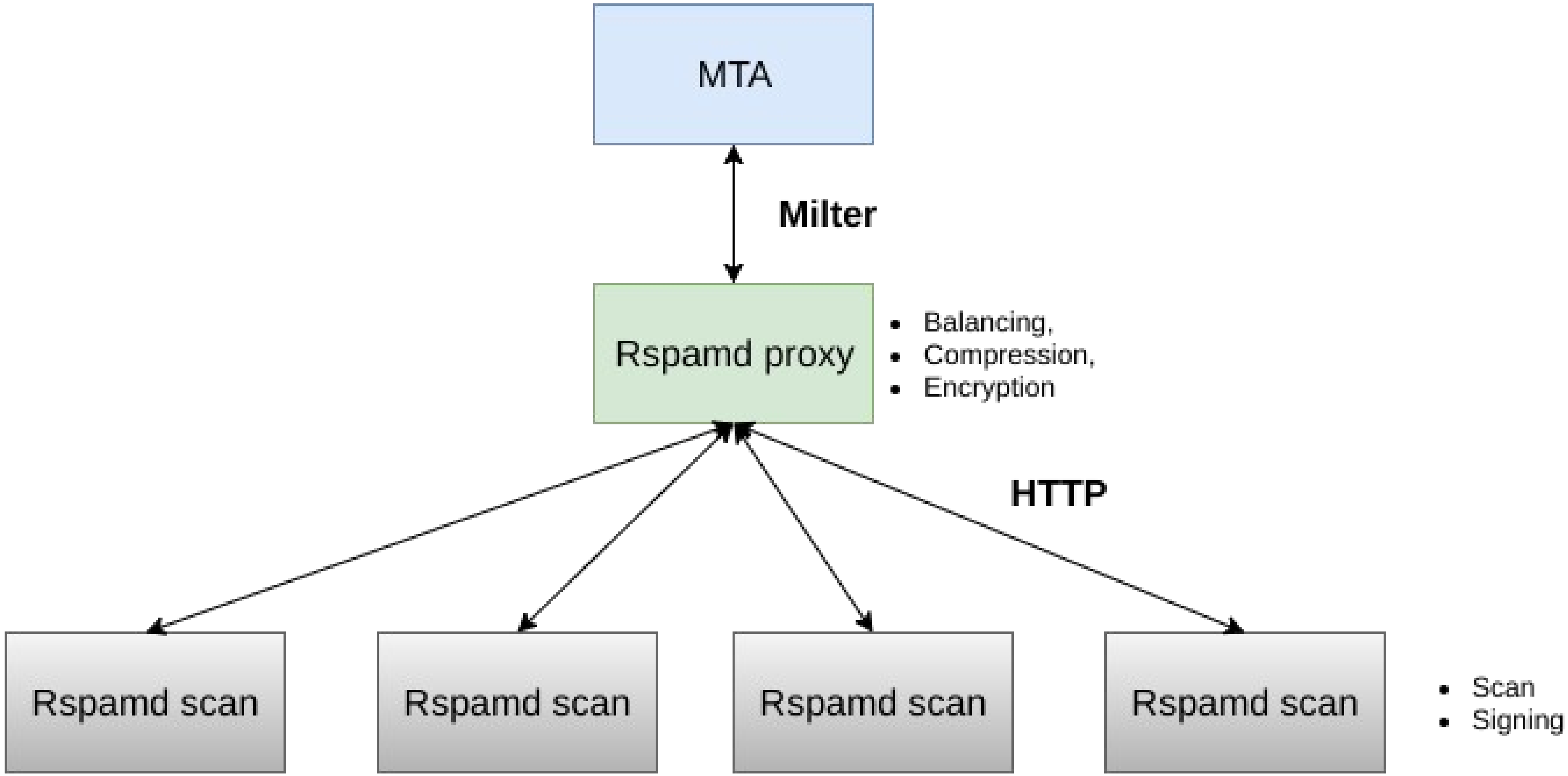
Rspamd – single server



Rspamd - Architecture - 2+ Servers (Loadbalancing)

- HTTP communication between Proxy and Scanner
- Loadbalancing / Network configuration using Upstreams
- Alternation + Priority
  - master-slave
  - round-robin
  - random
  - sequential
  - Hash
- Redis Redundancy configured directly in Redis

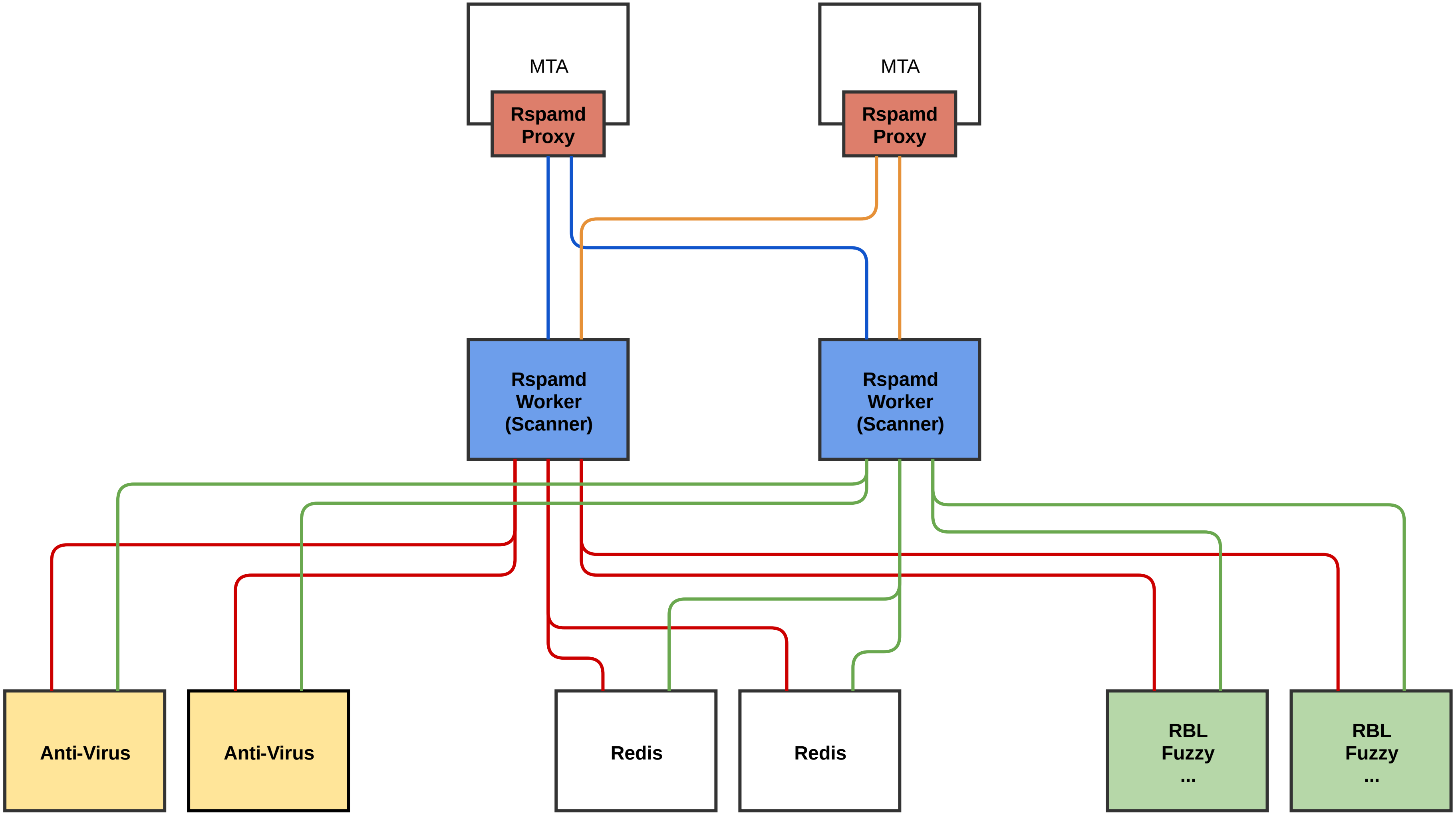
Rspamd - Multiserver



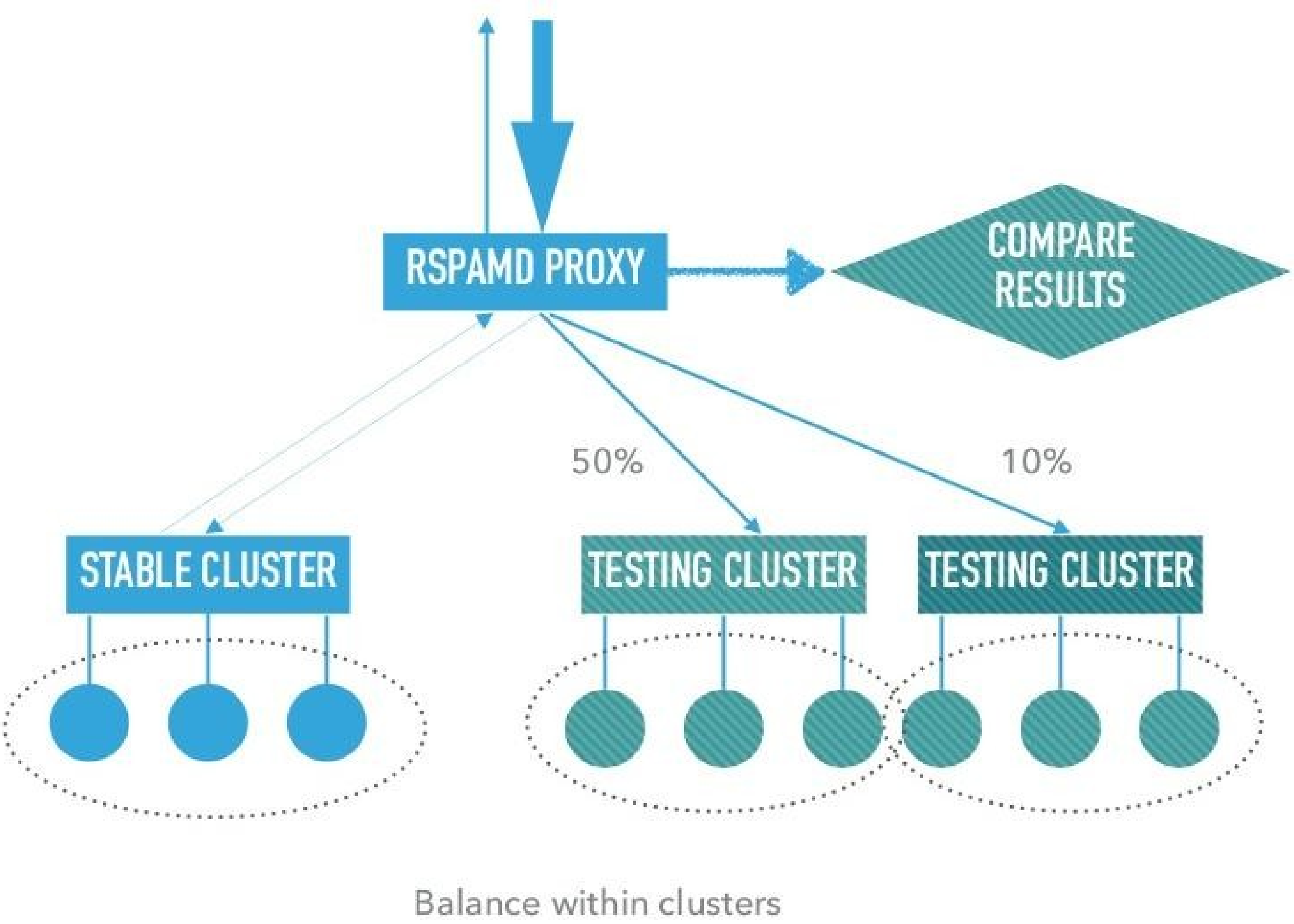
- Balancing,
- Compression,
- Encryption

- Scan
- Signing

Rspamd – Cluster with Upstreams



Rspamd – production and development cluster



Rspamd - Anti-Spam Feature Implemetation

- Features are organized in separate modules or function blocks
- Modules are massively using the Lua Framework
- Every module has its own configuration file
- Dedicated feature code is minimal compared to the functionality

Rspamd - Symbols and Scores

- → Every function registers a symbol
- → Every module exposed one or more symbols
- → Symbols can have a score
- → Symbols have to register to a processing stage
- → A symbol can have one or more dependencies on other symbols
- → Symbols can be deactivated or removed from current task
- → Symbols can be used in expressions or force_actions

Rspamd Actions

→ Actions are performed when the score sum reaches a config limit

→ Force Actions are performed when the defined expression is true

```
actions {

     greylist =        8;
     rewrite_subject = 12;
     add_header =      13;
     reject =          15;

};
```

Rspamd Features - as expected

- → Real Time Blacklists / Spam URI RBL
- → Mail RFC / DNS Tests
- → Mime-Type / Attachment Recognition
- → Antivirus scanner support
- → (Regex) Rules for mail header and body

Apropros: Antivirus - ClamAV

Do you still believe ClamAV is a wonderful tool to fight spam?

ClamAV as Spam-Recognition Tool

- ClamAV is able to load unsigned external signatures
- Create your own signatures using *sigtool*
- Google Safe Browsing database via freshclam
- Many disabled options
  - DetectPUA
  - OLE2BlockMacros
  - ...
- extra Signatures from Sanesecurity, Securiteinfo, Malwarepatrol, Yara Rules

ClamAV - use of external signatures

- Do not reject every ClamAV hit as virus
- Have a look to the signatures False Positive classification
- Convert the hits to symbols and scores
  - Good Signature → high score
  - Low Risk → high score
  - High Risk → low score

Rspamd - ClamAV sample patterns

```
patterns {
  # symbol_name = "pattern";
  CLAM_JUST_EICAR = "^Eicar-Test-Signature$";
  CLAM_DOC_MALWARE = "^Doc\.Malware\..*";
  CLAM_HTML_PHISH = "^Html\.Phishing\..*";
  CLAM_WIN_WORM = "^Win\.Worm\.Mydoom-.*";
  # Heuristics
...
  CLAM_PUA_WIN = "^PUA\.Win\..*";
  # Extra Signatures
  CLAM_G_SAFEBROWSING = "^Heuristics.Safebrowsing.*";
  CLAM_SANESEC_JUNK = "^Sanesecurity\.Junk.*";
  CLAM_SANESEC_BLURL = "^Sanesecurity\.Blurl.*";
  CLAM_SECI_JPG = "^SecuriteInfo\.com\.JPG.*";
  CLAM_MP_EVILMACRO = "^MiscreantPunch.EvilMacro\..*";
  CLAM_YARA = "^YARA\..*\.UNOFFICIAL$";
}
```

Rspamd - noticeable Features

→ Multimaps - matching (regex) lists against mail values

→ MX-Check - really connects to remote MX

→ ASN - provides advanced GEO-IP information

→ Spamassassin - includes Spamassassin rules

→ E-Mails - Mail address blacklist

→ Spamtrap - learn mails to specified addresses as spam

Rspamd - noticeable Features #2

- External Services
  - Detailed Office Macro recognition - oletools
  - Query ICAP (HTTP-Proxy) virus scanners - icap
  - DCC / Pyzor (soon™) / Razor (soon™)
  - ...
- URL Redirector - resolve URL shorteners
  - But does not open the final link ;-)

Rspamd - DKIM / ARC signing

- → Rspamd supports DKIM / ARC signing using RSA and ED25519 keys
- → Rspamd supports multi-key signing
- → Obviously also DKIM / ARC signature verification
- → OpenDKIM config support
- → Key search based on Maps/Vault and with fallback
- → DNS Key verification for signing key
- → DMARC Policy support with sending Reports

Rspamd – DKIM / ARC Whitelist

- Based on a map (list) of domains like e.g. paypal.com
- Positive DMARC verification will add -7 points
- failed Verification will add +12 points

Rspamd Settings - Scan Profiles

→ Per - *not only user* - but nearly everything scan profiles

→ Adjust limits and symbol scores

→ Enable or disable Symbols or Symbol Groups

→ Set mime (blocking) configuration

Rspamd - Config Merging

- Single options will be overwritten (enabled=true;)
- Tables (Arrays) or blocks will be merged
- Merge path for a module:
  - Module defaults
  - modules.d config
  - local.d config
  - dynamic config (webinterface)
  - (settings config)
  - special: override.d config

Jinja2 Support - Config files as Templates

- Rspamd can load a variables file on startup
- Enviromnent variables are used
- {= =} enclosed tags will be processed

```
servers = "{= clamav_upstreams =}";
```

Rspamd vs Spamassassin - lets fight

... wait

Rspamd vs Spamassassin - let‘s team up

- Instead of just replacing Spamassassin we could integrate it
- We are fighting on the same side
- Spamassassin has a great ruleset and nice plugins - but does not work best at pre-queue scanning
- Spamassassin is very ressource hungry compared to rspamd
- But we still trust Spamassassin's opinion

Rspamd vs Spamassassin - let‘s team up

- Rspamd can directly load Spamassassin rules
- Rspamd can query Spamd in external_services
  - Disable all remote checks in Spamassassin
  - Maybe disable all rules and just use your favorite SA plugins
  - Rspamd supports a dynamic scan option (1.9.3)
    - Runs modules at postfilter stage (after all other checks)
    - If 0 < score < 2 * reject level
    - Then scan additionally query spamd
- Use the Spamassassin rules in Rspamd‘s spamassassin plugin whenever possible

Rspamd – the missing modules

- Bayes - statistical text analysis
- Fuzzy – other statistical text analyses and hashing
- Reputation - generic reputation using any available variable
  - Client IP, Domains, X-Mailer …
- Clustering - multi-value dependend reputation
- Ratelimit - adaptive limits using any available variable
- Replies - evaluate recurring communication
- Greylisting - defer some messages based on scan results
- Neural Network - postprocess Ham- / Spam-Sets using basic AI algorithm
- Spamtrap - learns mails to specified addresses as spam

Rspamd – the missing modules

- Bayes - statistical text analysis
- Fuzzy – (other? Wie in anderer Folie) statistical text analysis and hashing
- Reputation - generic reputation using any available variable
  - Client IP, domains, X-Mailer ...
- Ratelimit - adaptive limits using any available variable (clustering nicht?)
- Replies - evaluate recurring communication
- Greylisting - defer some messages based on scan results
- Neural Network - postprocess Ham / Spam-Sets using basic AI algorithm
- Spamtrap - learns mails to specified addresses as spam

- **Those modules need to be fed to work reasonable**

Rspamd - IPScore (new Reputation)

IP_SCORE(4.28) [ip: (9.91), ipnet: 89.163.128.0/17(7.24),

asn: 24961(4.28), country: DE(-0.04)]

- Self-learning personal IP Reputation Database
  - RBL + GeoIP
- The Score for a Client-IP is calculated from IP-Score itself, Network Score, ASN Score, Country Score with descending weights
  - ip_score = action_multiplier * tanh (e * (metric_score/score_divisor))
  - Scores { ip = 1.0; ipnet = 0.8; asn = 0.5; country = 0.1; }

Current common setup of Anti-Spam systems

- Reject as early as possible
  - smtpd_recipient_restrictions
  - postscreen
- Reject everything non-RFC compliant
- Reject on single RBL hit
- Firewall Mitm Rejects

Current common setup of Anti-Spam systems

```
smtpd_recipient_restrictions =
...
 reject_non_fqdn_sender
 reject_non_fqdn_recipient
 reject_unknown_sender_domain
 reject_unknown_recipient_domain
 reject_unknown_reverse_client_hostname
 reject_invalid_helo_hostname
 reject_non_fqdn_helo_hostname
  # RBL checken!
 reject_rbl_client zen.spamhaus.org,
 reject_rbl_client ix.dnsbl.manitu.net,
  # policyd-weight
 check_policy_service inet:127.0.0.1:12525
...
```

Current common setup of Anti-Spam systems

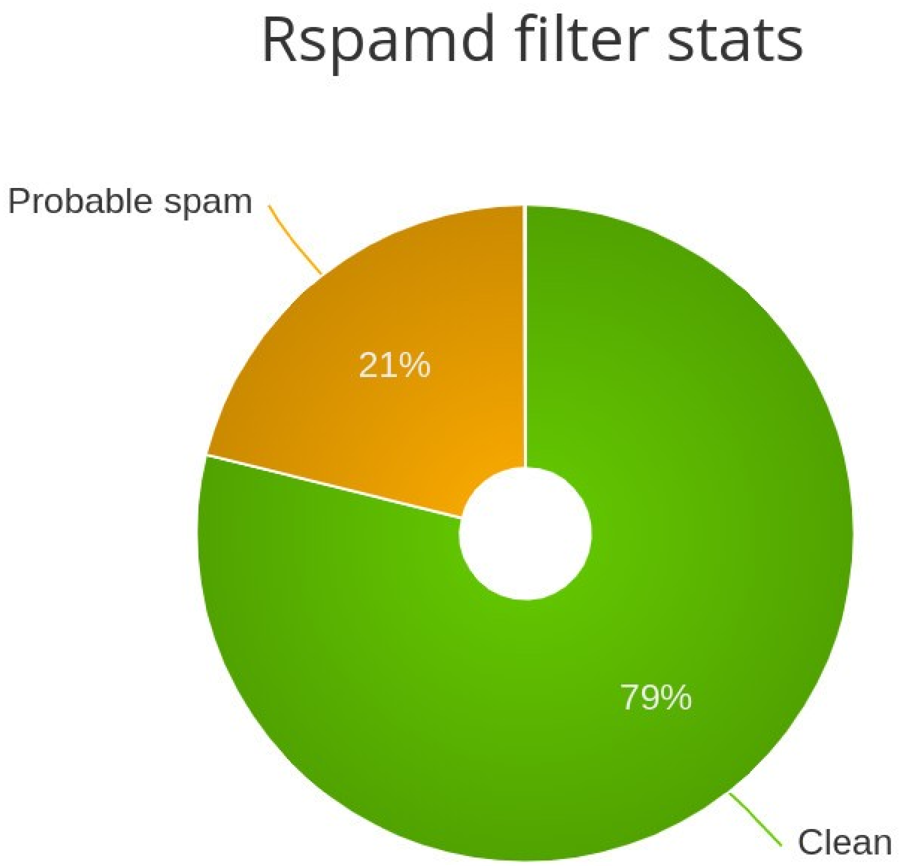
→ 20-60% early rejects

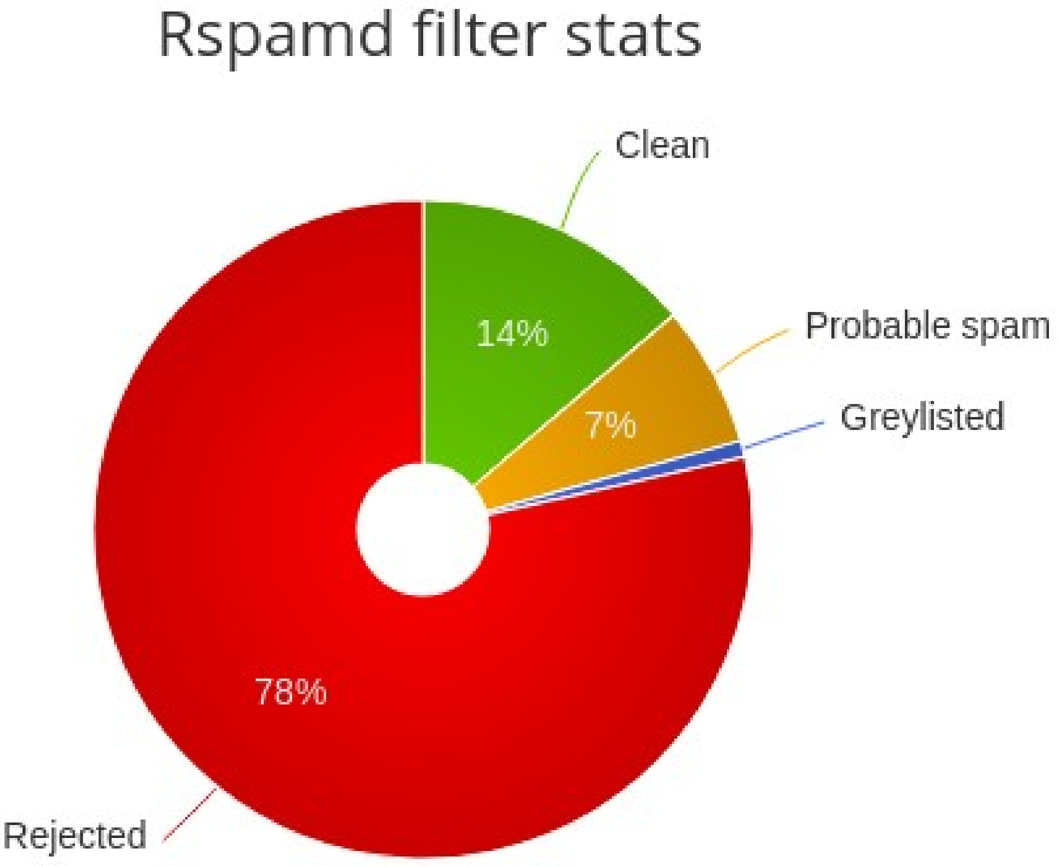
→ Reject at RCPT TO stage

→ Connection closed by sniffing Firewall

→ No chance for any Anti-Spam System to process the content

Rspamd in those setups ...




Rspamd filter stats
Clean
14%
Probable spam
7%
Greylisted
78%
Rejected

The problem of early rejects

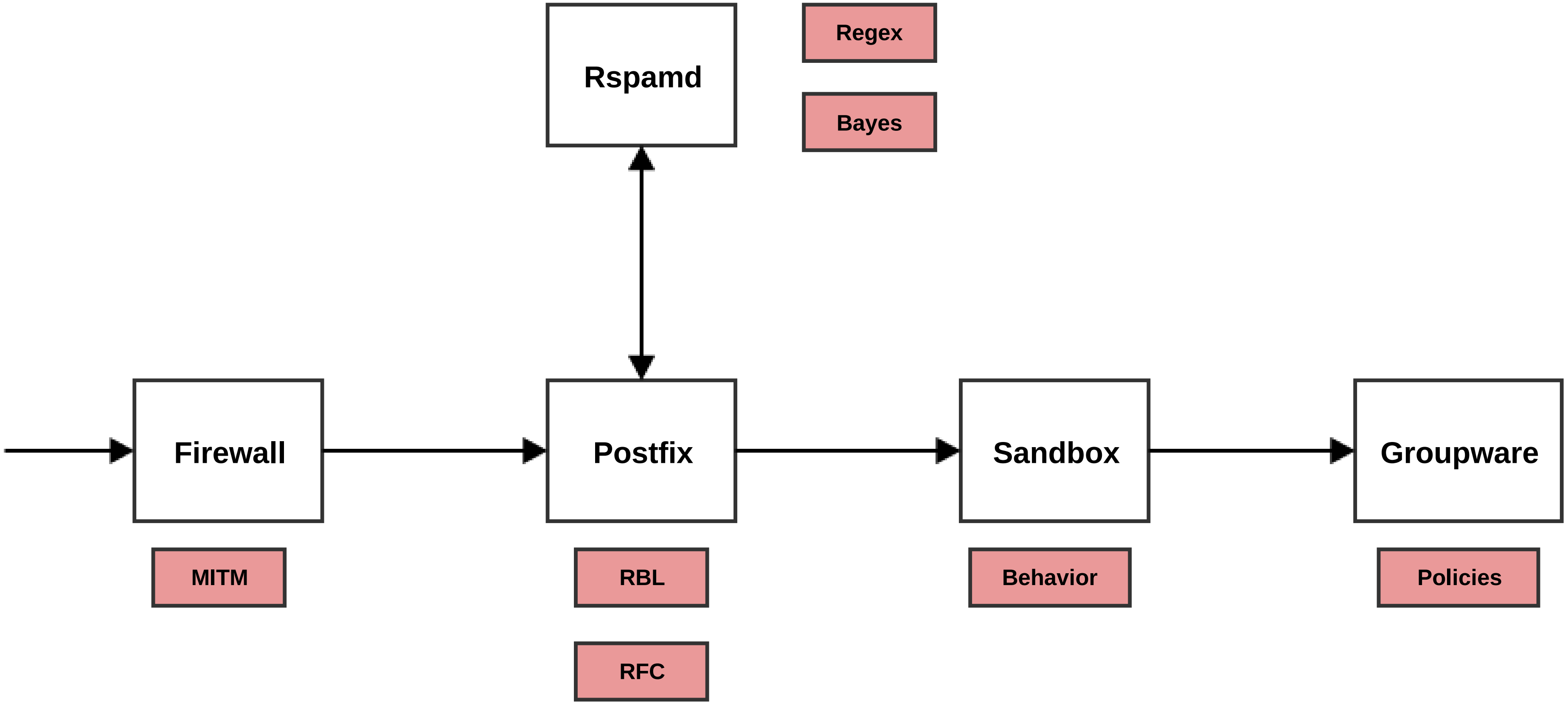
- → Rspamd can't see the "easy to recognize" Spam
- → Rspamd learns too much ham und too little spam
- → Text statistics and reputation data is unprecise

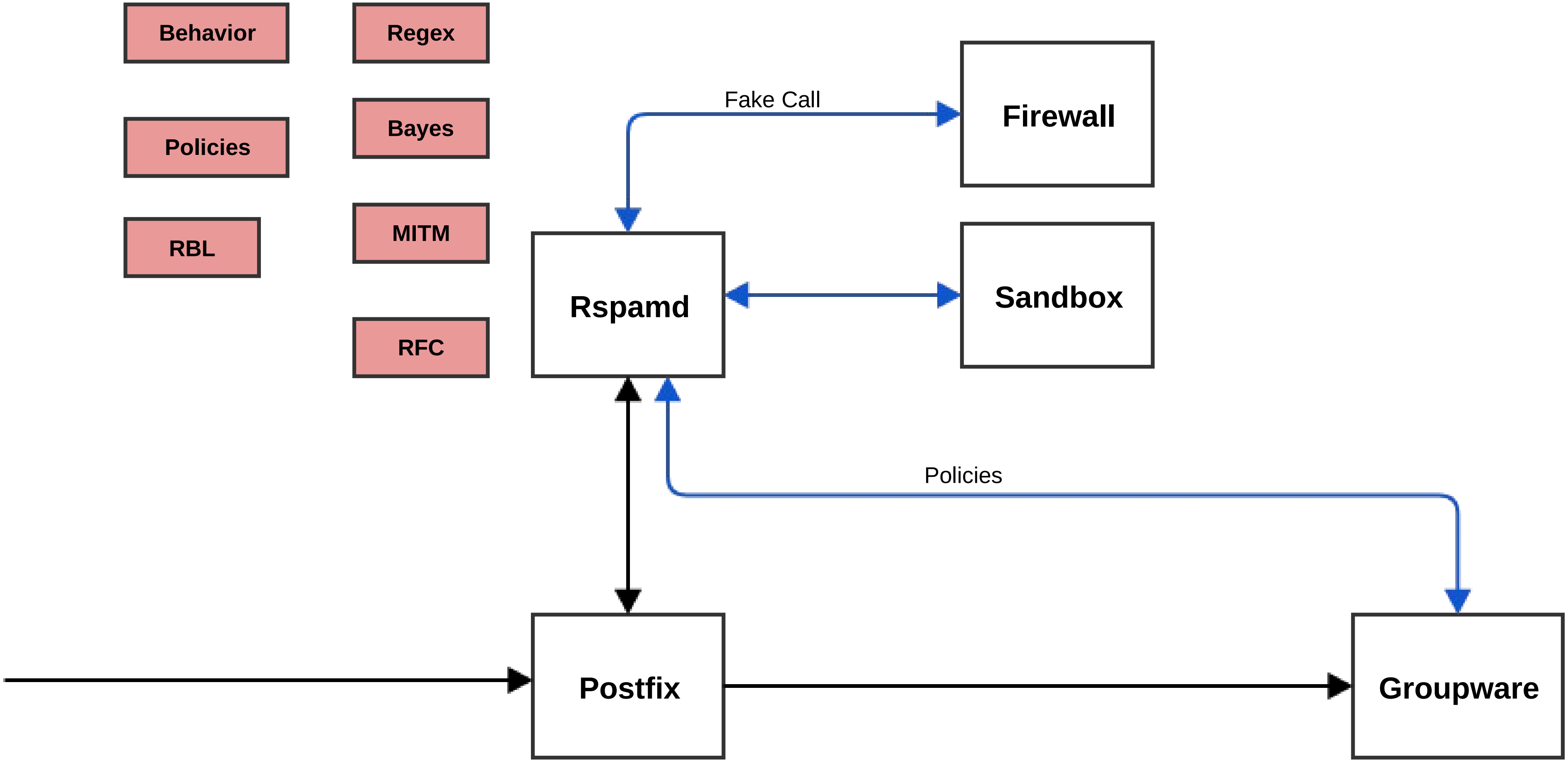
Open the Doors

- → Combine all possible information to clarify the ham / spam decision
- → Static results will help the dynamic modules to improve their data
- → Disable RBL checks, Postscreen, RFC checks, Greylisting, Policyd-weight in Postfix
- → Disable Firewall / Appliance Scanning (before Rspamd)
- → Use Spamtrap Module when possible
- → Build dedicated Honeypots
- → Learn from known outgoing spam
- → Implement your security companies policies into Rspamd

BUT this will burn my system

- → Rspamd itself is heavily optimized
- → Just take care about expensive external sources
- → Maybe run some modules dynamically at postfilter stage
- → **AND - It's much more time consuming to manually improve a blind Anti-Spam system**


Rspamd
Regex
Bayes
Firewall
Postfix
Sandbox
Groupware
MITM
RBL
RFC
Behavior
Policies


Behavior
Regex
Policies
Bayes
RBL
MITM
RFC
Fake Call
Firewall
Rspamd
Sandbox
Policies
Postfix
Groupware

Soweit, so gut.

Gleich sind Sie am Zug:
Fragen und Diskussionen!

Wir suchen:
Admins, Consultants, Trainer!

Wir bieten:
Spannende Projekte, Kundenlob, eigenständige Arbeit, keine Überstunden, Teamarbeit

...und natürlich: Linux, Linux, Linux...

http://www.heinlein-support.de/jobs

Heinlein Support hilft bei allen Fragen rund um Linux-Server