Rspamd : logiciel anti-spam opensource, performant, évolutif et personnalisable

Gauthier Catteau
DSI Académie de Lille
110 avenue Gaston Berger
59000 Lille

Résumé

La disparition de la messagerie électronique est annoncée depuis de nombreuses années, mais nous pouvons constater qu'en 2019, les échanges par mail n'ont jamais été aussi présents et indispensables.

Et pourtant les messages de « phishing » sont de plus en plus pernicieux, le nombre de spam a explosé et les utilisateurs sont devenus plus exigeants que ce soit sur la qualité du filtrage que sur le temps d'acheminement des messages.

La messagerie électronique n'est aujourd'hui utilisable que grâce à l'efficacité de nos solutions anti-spam et des personnes qui les administrent.

J'ai choisi de vous présenter le logiciel libre Rspamd qui a fait son apparition il y a maintenant 10 ans, mais qui est resté longtemps méconnu.

Rspamd est une solution efficace, performante et dont la mise à l'échelle peut se faire facilement par le simple ajout de nœuds. Il répond de façon élégante à pratiquement toutes les problématiques de filtrage que j'ai pu rencontrer ces 20 dernières années. Et s'il vous manque quelque chose que vous avez imaginé, il est possible facilement d'écrire des extensions dans le langage lua pour étendre ses fonctionnalités.

Mots-clefs

Courrier électronique, anti-spam, phishing

1 Introduction

En 2016 notre solution anti-spam, basée sur les briques Amavis, Clamav, Spammassin et Cluebinger, mise en place environ 10 ans plus tôt, commençait à montrer des signes importants de saturation. Le temps moyen de traitement des messages était de plus en plus long. La moindre campagne de spam ou de mailing officiel impactait fortement le fonctionnement de la messagerie.

Du côté de l'administration de la messagerie, la mise en place de nouvelles règles de filtrage était fastidieuse. Cela demandait un niveau d'expertise relativement élevé et nécessitait un redémarrage parfois laborieux des différentes briques. Toutes ces manipulations ne faisaient qu'allonger la file d'attente des messages à traiter.

Face à ce constat, nous avons décidé d'étudier différentes solutions anti-virus et anti-spam de messagerie de préférence à base de logiciels libres. La solution proposée par RENATER avait été écartée, car à l'époque il était nécessaire de maintenir une solution anti-spam pour les flux de messages sortants.

Un premier tour d'horizon des solutions nous a permis de constater que l'offre était relativement réduite et que les logiciels historiques avaient très peu évolué ces 10 dernières années, au point que certains semblaient abandonnés.

Rspamd, le seul logiciel libre dont le projet est vraiment actif, semblait répondre de façon élégante à la plupart des problématiques que nous rencontrions, que ce soit en termes de performance ou de fonctionnalité.

Nous avons donc décidé de tester Rspamd et ensuite de le mettre en production.

2 Présentation de la solution

Rspamd est un système anti-spam qui permet d'évaluer les messages selon un certain nombre de règles, des expressions régulières, des analyses statistiques et des services personnalisés comme les listes noires d'URL. Chaque message est analysé par Rspamd et reçoit un score comme nous avons l'habitude de le voir avec spamassassin.

En fonction de ce score, Rspamd demande au MTA[1] d'appliquer une action sur ce message : par exemple laisser passer, rejeter, modifier le sujet ou juste modifier l'en-tête pour indiquer que ce message est probablement un spam.

Rspamd est écrit en C, il est conçu pour traiter simultanément des centaines de messages par seconde et dispose d'un certain nombre de fonctionnalités. Certaines écrites en C, font partie du cœur de Rspamd et d'autres dans le langage de plugin Lua[2].

1. Mail transfert agent, serveur permettant de router le courrier électronique.

2. Site officiel de Lua : https://www.lua.org

2.1 Fonctionnement

Rspamd est composé de 5 processus appelés « worker » qu'il est possible d'activer indépendamment dans les fichiers de configuration :

- le *worker* « normal » conçu pour scanner les messages ;
- le *worker* « controller » chargé des actions de configuration, de l'apprentissage et de traiter les requêtes de l'interface web ;
- le *worker* « fuzzy_storage » permet de stocker les hachages flous ;
- le *worker* « proxy » gère le transfert des requêtes et le protocole milter[3] ;
- le *worker* « lua » qui exécute les scripts Lua personnalisés.

Nous avons installé Rspamd et configuré le *worker* « proxy » sur nos MTA, mais il est possible de centraliser sur un seul serveur le routage des flux Rspamd. Les *workers* « normal », « controller » et « lua » sont configurés sur nos nœuds Rspamd.

Les *workers* « fuzzy_storage », que nous avons choisis de ne pas déployer, doivent être exécutés sur des nœuds dédiés avec une base Redis séparée.

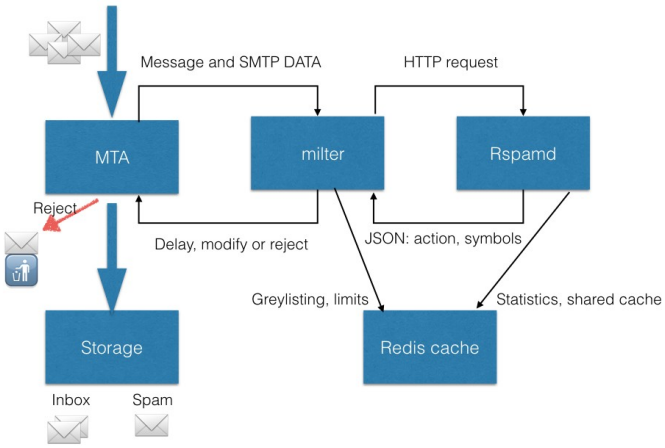

Figure 1: Fonctionnement de Rspamd

Lorsqu'un message arrive sur le MTA, il est traité par le *worker* « proxy » via le protocole *milter*. Le *worker* « proxy » envoi les données sur l'un des *workers* « normal » défini dans sa configuration via le protocole HTTP. En fonction du résultat de l'analyse, le worker « normal » retourne une action que le MTA appliquera avant la fin de la session SMTP. En fonction de cette action, le message sera délivré ou rejeté.

3. Milter : protocole permettant d'externaliser le filtrage des e-mails en dehors des MTA.

2.2 Fonctionnalités

2.2.1 Globale

- Rspamd possède une interface web d'administration que nous présentons page 6.
- Il fonctionne avec les principaux MTA open source (Postfix, Exim, Sendmail ou Haraka).
- Rspamd intègre des centaines de règles dans le langage Lua et il est également possible d'en ajouter en s'inspirant des règles existantes.
- Et il permet l'utilisation de listes dynamiques téléchargeables en http(s) ou présents dans un fichier sans nécessiter de redémarrage.

2.2.2 Analyse de contenu

- Rspamd est capable de s'interfacer avec plusieurs antivirus : ClamAV, F-Prot, Sophos, Avira et Kaspersky.
- Le filtrage par expressions régulières offre un traitement de base des messages, de leurs parties textuelles, des en-têtes MIME et des données SMTP reçues par le MTA par rapport à un ensemble d'expressions qui inclut à la fois des expressions régulières normales et des fonctions de traitement des messages.
- Les « Fuzzy hashes » ou hachages flous sont utilisés par Rspamd pour identifier des messages similaires. Ces structures visent à masquer les petites différences entre les messages de spam afin de trouver rapidement des messages communs. Cette base peut être alimentée directement depuis des remontées utilisateurs ou via des données collectées par des *honeypots*. Pour l'instant, au vu de la qualité des données remontées par nos utilisateurs, nous avons préféré ne pas utiliser ce module.
- DCC est assez similaire au module précédent, mais il utilise le service externe éponyme pour vérifier si un message est envoyé en masse.
- Le module « Chartable » permet de trouver des messages spécialement conçus pour tromper les systèmes de filtrage du spam en changeant la langue du texte et en remplaçant des lettres par leurs analogues visuellement identiques. Rspamd utilise la normalisation UTF-8 pour détecter et filtrer ces techniques couramment utilisées par de nombreux spammeurs.

2.2.3 Vérification de la politique de filtrage

Un ensemble de modules permet d'évaluer les messages transitant par le MTA.

- Les contrôles SPF[4] permettent de valider la source d'un message en utilisant la politique définie dans l'enregistrement DNS du domaine de l'expéditeur.

4. Sender Policy Framework : https://tools.ietf.org/html/rfc7208

- La politique DKIM[5] valide la signature cryptographique d'un message par rapport à une clé publique placée dans l'enregistrement DNS du domaine de l'expéditeur. Cette méthode permet de s'assurer qu'un message a été reçu du domaine spécifié sans être altéré en chemin. Rspamd permet également d'appliquer la signature DKIM pour les messages envoyés depuis notre domaine pour les utilisateurs authentifiés.
- DMARC[6] combine les techniques DKIM et SPF pour définir des politiques plus ou moins restrictives pour certains domaines. Rspamd peut également stocker des données pour les rapports DMARC dans la base de données Redis.
- ARC[7] est un ajout relativement récent au mécanisme de signature DKIM qui permet de transmettre des messages signés sur une chaîne de relais de confiance.
- Les listes blanches sont utilisées pour éviter les faux positifs pour les domaines de confiance qui passent d'autres contrôles, tels que DKIM, SPF ou DMARC.
- Les listes DNS RBL permettent d'estimer la réputation de l'adresse IP ou du réseau de l'expéditeur. Rspamd utilise un certain nombre de listes DNS, y compris des listes telles que SORBS ou SpamHaus. Cependant, Rspamd ne fait pas totalement confiance à ces listes et ne rejette donc pas le courrier uniquement sur le résultat de l'une d'entre elles, comme on peut le voir trop souvent.
- Les listes d'URL sont similaires aux listes noires DNS mais utilisent les URL dans un message pour combattre le spam et le phishing. Rspamd prend en charge les listes SURBL les plus populaires, telles que URIBL et SURBL de SpamHaus[8].
- Rspamd utilise des algorithmes sophistiqués pour trouver les URL de phishing et supporte les redirecteurs d'URL populaires (par exemple, http://t.co) pour éviter les faux positifs. Les bases de données de phishing populaires, telles que OpenPhish et PhishTank sont également prises en charge.
- Le module « Rate Limits » permet d'empêcher l'envoi de courriels en masse à partir d'un de nos comptes piratés. Il est également possible de définir des seuils en fonction de l'IP, de l'adresse e-mail source.
- Le plugin « IP reputation » permet d'ajuster progressivement la réputation des adresses IP spécifiques, des réseaux, des blocs autonomes (ASN) et même des pays en fonction de la proportion de spam reçu.
- Rspamd implémente le *greylisting*. Mais, pour ne pas pénaliser les messages légitimes, ce mécanisme n'est activé que pour les messages pour lesquels il y a

5. DomainKeys Identified Mail : http://dkim.org

6. Domain-based Message Authentication, Reporting and Conformance : https://dmarc.org

7. Authenticated Received Chain : http://arc-spec.org/

8. SURBL et SpamHaus sont gérés par la même société. Il est nécessaire d'acquérir une licence si le flux est trop élevé.

un doute. La demande de réémission du message permet parfois une meilleure catégorisation de celui-ci lorsqu'il se représente sur nos serveurs.

- Le module « Réponses » met en liste blanche les messages qui répondent à nos propres messages envoyés précédemment.
- Le module « Multimaps » fournit un outil pratique pour filtrer les messages en fonction de différents attributs : en-têtes, message, IP de l'expéditeur, etc. Ce module est très utile pour construire des règles personnalisées facilement modifiables sur l'ensemble du cluster via l'interface web et sans nécessiter le redémarrage de Rspamd.
- Enfin le module « MX Check » permet de vérifier si le domaine de l'expéditeur possède au moins un MX valide et fonctionnel.

Tous ces modules d'analyse vont permettre d'assigner un ensemble de symboles et de scores à chaque message. Il est également possible grâce au module « composite » d'ajouter de nouveaux symboles et de leur attribuer un score en construisant une opération logique à partir d'autres symboles.

2.3 Interface d'administration

Rspamd propose une interface d'administration simple permettant d'effectuer les opérations courantes. Elle est accessible via un navigateur en se connectant sur l'un des nœuds du cluster.

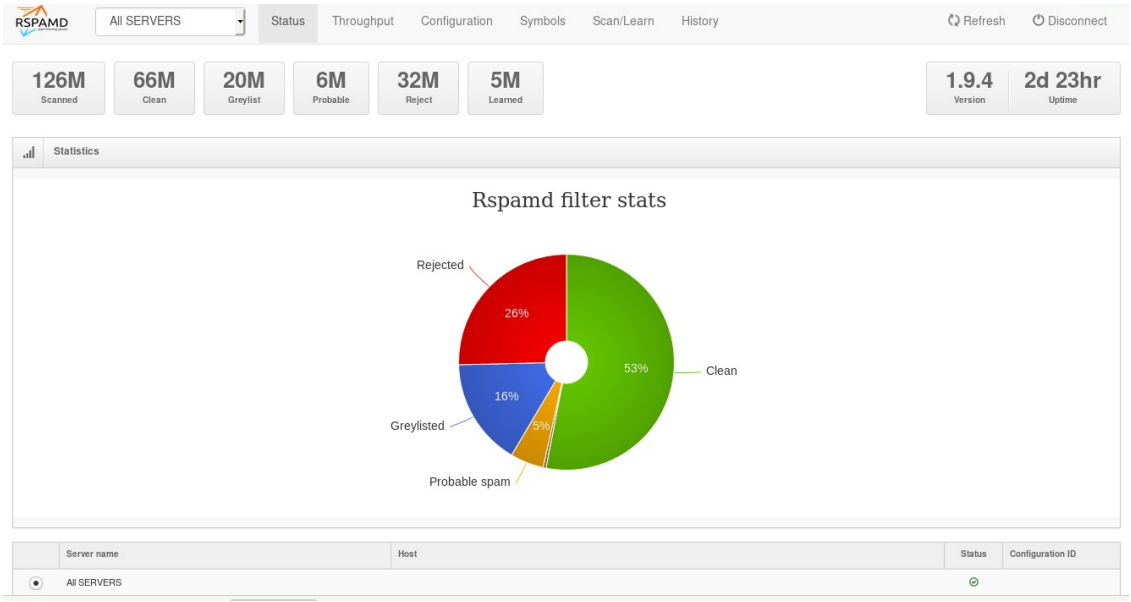

Figure 1 - Page d'accueil de Rspamd

La page d'accueil permet de connaître l'état global du cluster, de connaître la version déployée, de vérifier que chaque nœud possède la même configuration et d'avoir des statistiques globales depuis la dernière réinitialisation des compteurs.

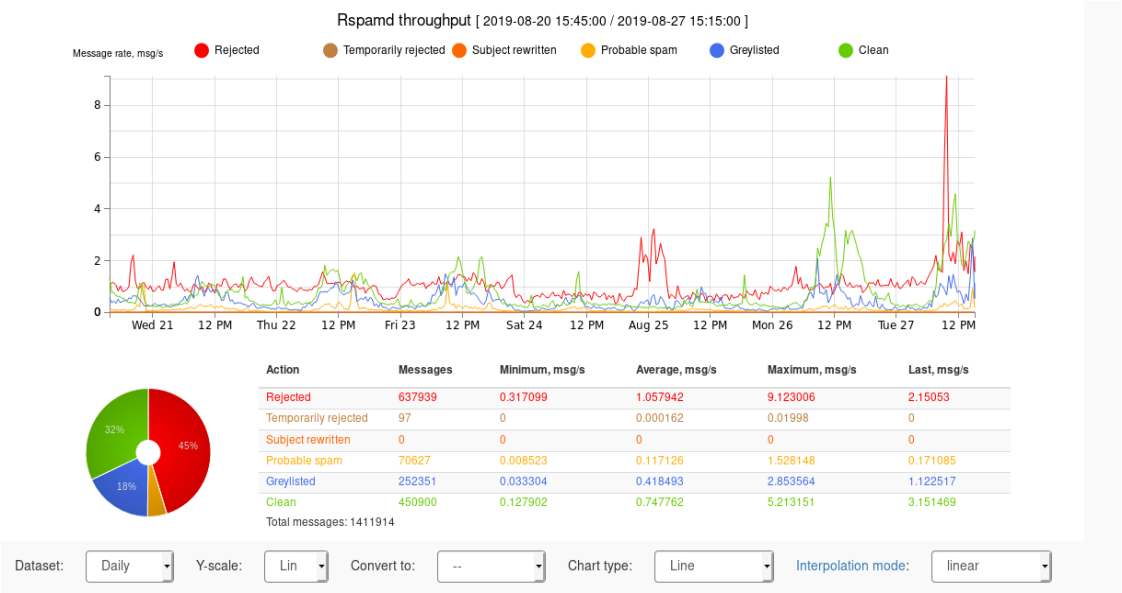

Figure 2 - Onglet « Throughput »

Le deuxième onglet permet d'avoir un aperçu sous forme de graphique et de tableau du nombre de messages traités, ainsi que leur répartition dans le temps. Ces données sont consultables par jour, semaine, mois ou année.

Sur une année, nous avons rejeté 31 millions de messages, tagué 5 millions en tant que spam probable et délivré 55 millions.

Pour obtenir ce type de métrique, nous utilisons également le module Rspamd « metric_exporter » qui permet d'envoyer ces informations sur notre serveur graphite et les afficher avec l'application Grafana.

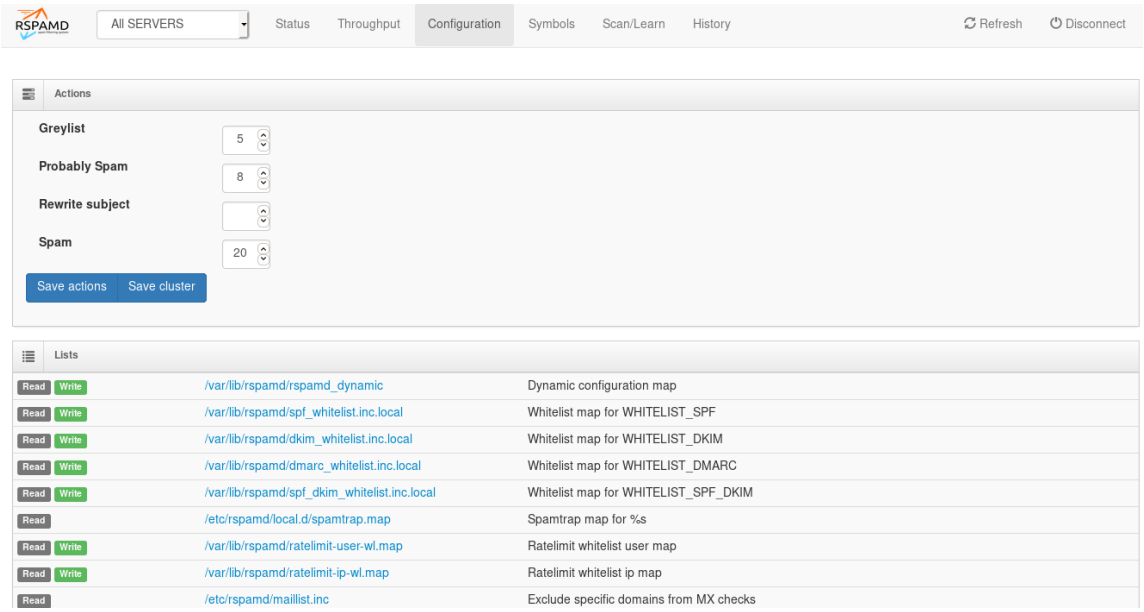

Figure 3 - Onglet « Configuration »

L'onglet configuration permet dans la partie haute de la page de modifier les seuils d'actions sur l'ensemble du cluster.

Dans la partie basse, il est possible de modifier les listes *multimap* en un clic sur l'ensemble du cluster. Il est possible d'ajouter une expression régulière, une chaîne de caractère, une adresse e-mail, un domaine, une IP, un pays. Toutes ces listes seront analysées et permettront l'ajout d'un symbole et d'un score sur les messages analysés.

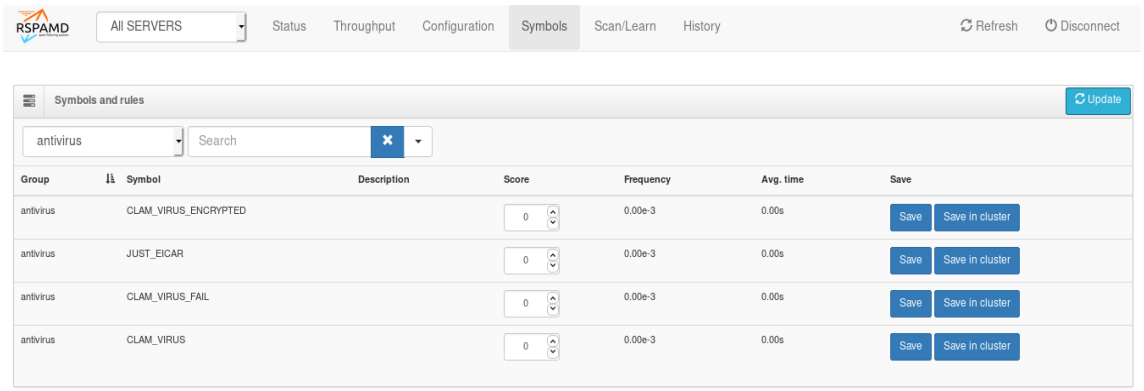

Figure 4 - Onglet « Symbols »

L'onglet « Symbols » permet de visualiser et de modifier la valeur de chaque symbole définie dans Rspamd.

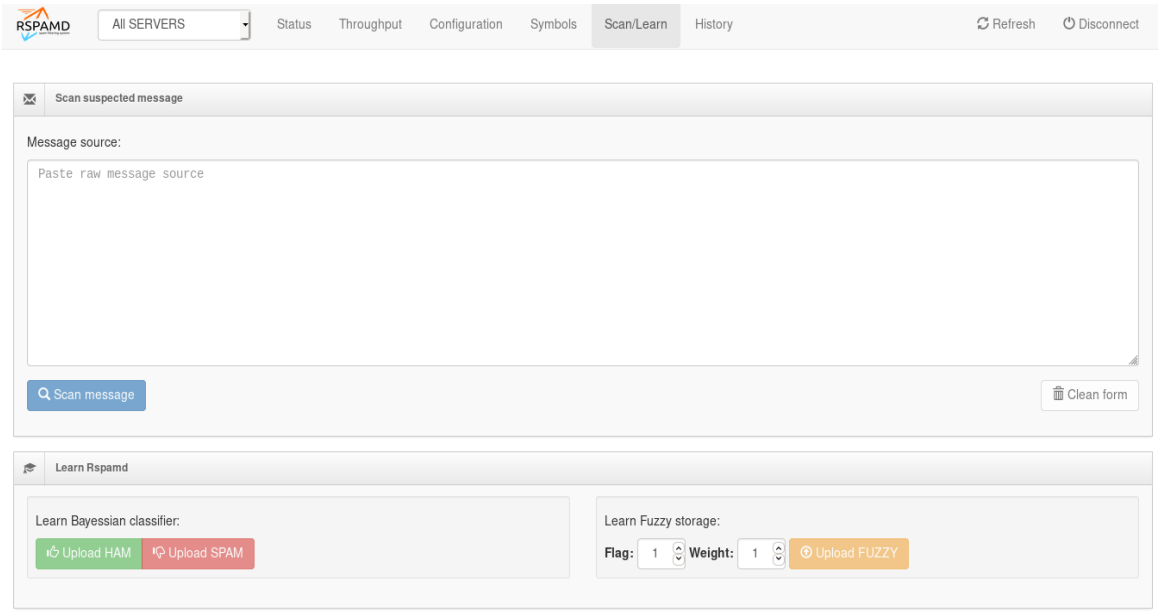

Figure 5 - Onglet « Scan/Learn »

L'onglet « Scan/Learn » est l'onglet utilisé pour le débogage des règles et l'apprentissage manuel des spams et faux positifs.

Pour cela, nous venons copier/coller le corps du message dans la partie « Message source » et nous cliquons sur « Scan message » pour connaître le score et la répartition des points sur les règles qui ont été prises en compte pour ce message.

Une fois le message scanné il est possible de demander manuellement l'apprentissage de ce message dans la base de données bayésienne en tant que spam ou ham.

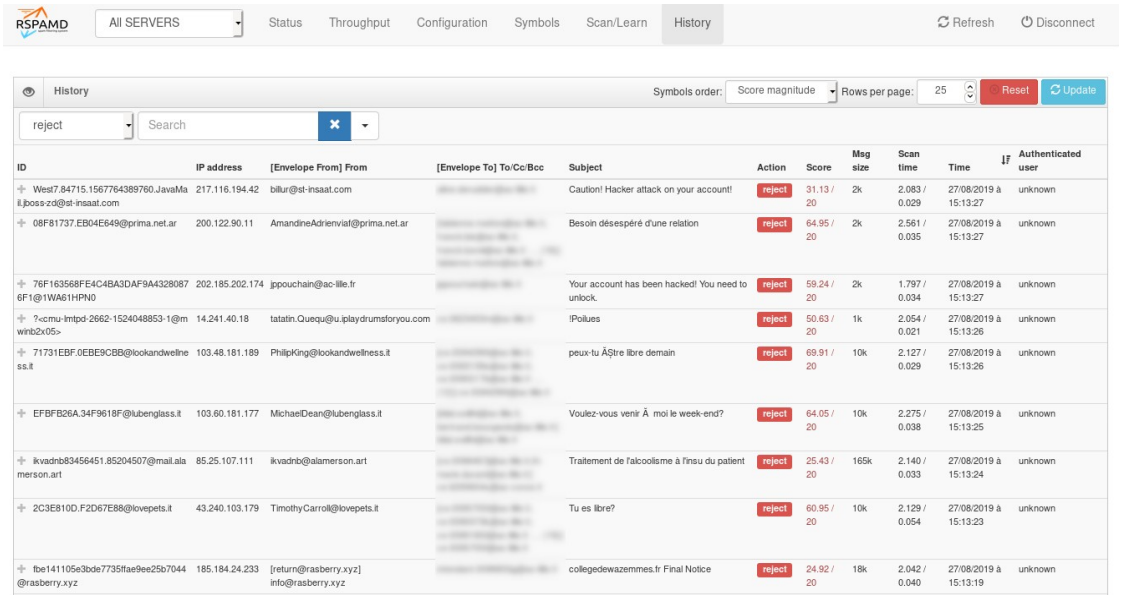

Figure 6 - Onglet « History »

Le dernier onglet « History » permet d'avoir une vision en temps réel des messages analysés sur l'ensemble des nœuds de la plateforme anti-spam.

Nous avons paramétré nos nœuds Rspamd pour qu'ils stockent dans la base Redis les informations des 500 derniers messages analysés, soit un total de 3000 messages dans le cas d'un cluster de 6 nœuds. Cela nous permet de visualiser les messages des dix dernières minutes environ. Un nombre de messages trop important rend l'onglet « History » inutilisable.

Toutes ces informations sont également envoyées dans ElasticSearch grâce au module « elastic » de Rspamd, nous utilisons donc plutôt Kibana pour consulter ces informations sur une période beaucoup plus grande et pour bénéficier de la puissance du moteur de recherche Lucent.

Figure 7 - Tableau de bord Rspamd sur ElasticSearch

3 Architecture et déploiement

3.1 La phase de test

Il est difficile de tester une solution anti-spam dans un environnement de développement ou de qualification, comme nous avons l'habitude de le faire pour la majorité des logiciels que nous mettons en place. Ce type d'outil doit pouvoir être testé avec de vraies données de production, mais de façon transparente. De plus il est indispensable de « chauffer » l'outil pendant plusieurs semaines ou mois pour affiner ses réglages, enrichir les filtres bayésiens ainsi que la réputation des adresses IP.

Le schéma d'architecture de la solution Rspamd nous montre qu'il est possible de rediriger une partie du flux de messagerie vers un cluster de test.

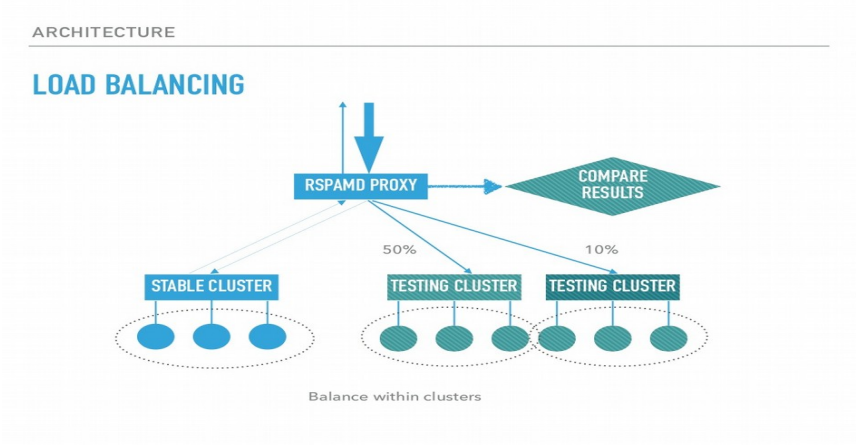

Figure 8 - Schéma d'architecture du worker « proxy »

Nous avons donc installé, comme sur le schéma, un cluster de « production temporaire ». Composé d'une seule machine virtuelle avec Rspamd et sa base de données, il avait pour consigne de tout laisser passer vers l'ancienne solution anti-spam.

Notre cluster de test (futur cluster de production), est lui composé de 6 machines virtuelles avec Rspamd et Clamav. Une copie de la totalité du trafic était envoyée sur ce cluster pour analyse, mais ce résultat n'entraînait aucune action sur le flux de messagerie.

Cela nous a permis d'affiner les règles de filtrage pendant plusieurs mois et de tester la montée en charge de notre cluster. Nous avons pu constater que le service Rspamd était très efficace en ressource. Les machines virtuelles sont tout de même configurées avec 2vCPU et 8Go de mémoire vive pour absorber la charge induite par l'analyse antivirale de l'utilitaire Clamav déployé sur chacun des nœuds.

Une fois qu'il était évident que le cluster était plus efficace que l'ancienne solution, il ne restait plus qu'à le mettre en production et retirer le routage des messages vers l'ancienne solution.

Ci-dessous un exemple de configuration du *worker* « proxy » pour répartir les demandes entre les nœuds d'un cluster et envoyer 10 % du trafic sur un nœud de test :

```
# local.d/worker-proxy.inc
# Main scan layer
upstream "scan" {
```

```
  default = "yes";
  hosts = "rspamd1:11333,rspamd2:11333,rspamd3:11333";
  key = "..."; # Public key for encryption
  compression = yes; # Use zstd compression
}



mirror "test" {
  hosts = "rspamd-test:11333";
  probability = 0.1; # Mirror 10% of trafic
  key = "..."; # Public key for encryption, generated by rspamadm
keypair
  compression = yes; # Use zstd compression
}
```

3.2 Mise en production

La mise en production de la solution a été effectuée en permutant les nœuds Rspamd dans le fichier de configuration du *worker* « proxy » et en simplifiant le routage des messages. Ces derniers n'ayant plus nécessité de passer par les anciennes briques anti-spam, pouvaient être déposés directement dans les stores de messagerie ou vers les serveurs MX des destinataires extérieurs à notre domaine.

Le premier bénéfice immédiatement perçu, fut la réduction notable du temps moyen d'acheminement des messages, qui avoisine maintenant les 2 secondes pour les messages légitimes.

Toutes les nouvelles configurations sont appliquées sur notre serveur Rspamd de test qui reçoit une copie de 10 % du trafic. Nous historisons les fichiers de configuration avec GIT et nous répliquons la configuration sur les 6 nœuds de production après avoir évalué l'impact du changement.

Pour les changements effectués directement dans l'interface Web de Rspamd, il est également possible de les appliquer uniquement sur le serveur de test et de tester l'impact en analysant le message dans l'onglet « Scan/Learn » avant d'appliquer ce changement sur l'ensemble du cluster.

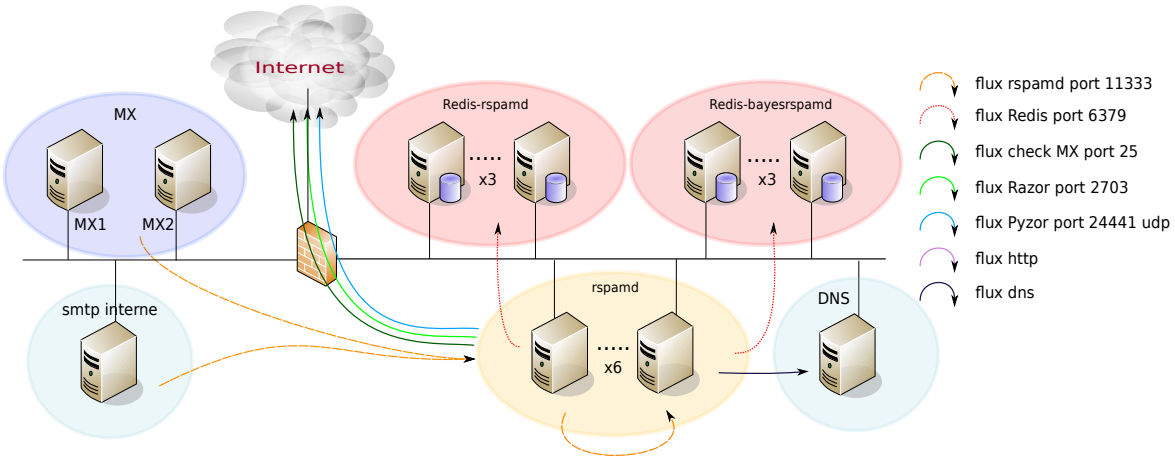

Figure 9 - Schéma de flux pour Rspamd

4 Extensions locales

4.1 Module de notification

L'idée, lors de la mise en place de Rspamd, était d'obtenir la plus grande transparence possible envers nos utilisateurs en ce qui concerne les messages mis en quarantaine.

Lorsque le message est probablement indésirable, Rspamd modifie son en-tête afin que le store de messagerie — via une règle *sieve*[9] globale — délivre ce message douteux dans le dossier *Junk* de l'utilisateur.

Lorsqu'un message est refusé, l'expéditeur est notifié directement par le serveur de son fournisseur d'accès ou par son client de messagerie. Le message est bloqué avant d'être accepté par nos serveurs. Il n'est donc pas nécessaire de notifier l'utilisateur de l'ensemble des messages rejetés à destination de son adresse e-mail.

Dans Rspamd il existe déjà un module Lua appelé *history_redis* stockant dans la base Redis les *n* derniers messages permettant de visualiser dans l'interface web les en-têtes et les règles appliqués sur ces messages (voir l'exemple de la figure Onglet « History » page 10).

Dans un premier temps nous pensions interroger directement la base Redis pour générer chaque jour l'e-mail de notification. Mais la quantité d'entrées stockées à interroger, uniquement pour connaître les messages dont l'en-tête avait été modifié, surchargeait inutilement la base Redis et l'onglet « History » de l'interface web de Rspamd.

Nous avons pour cela copié le fichier *history_redis.lua* en *notify_redis.lua*, remplacé *history* par *notify* et restreint l'enregistrement des données dans la base Redis aux actions *add_header*.

9. Sieve : langage pour filtrer les courriels qui sont déposés dans la boîte (http://sieve.info)

```
    local action = task:get_metric_action('default')
    if not (action == 'add header') then
          return
    end
```

4.2 De nombreux filtres

Prenons un exemple tiré d'une expérience récente. Les « spammeurs » ont pour mauvaise habitude de toujours choisir la nuit ou le week-end pour envoyer du spam depuis un compte piraté.

Nous avons donc créé une règle nous permettant d'identifier les connexions SMTP authentifiées en dehors des heures normales de travail. Pour cela nous modifions le fichier */etc/rspamd/lua/rspamd.local.lua* en ajoutant ceci :

```
rspamd_config.AUTH_NOT_WORK_HOURS = {
 callback = function(task)
        local rspamd_util = require "rspamd_util"
        local today = os.date('*t', rspamd_util.get_time())
        local hour = today.hour
        local wday = today.wday
        local uname = task:get_user()



        local Sunday = 1
        local Saturday = 7
        if uname then
                if ( hour < 7 or hour > 19 ) or ( wday ==
Sunday ) or ( wday == Saturday and hour > 12 ) then
                        return 1
                end
        end
  end,
  name = "AUTH_NOT_WORK_HOURS",
  score = 0.1,
  description = 'Authenticated mail sent outside working hours',
}
```

Tous les messages arrivant sur nos serveurs SMTP authentifiés entre 20 h et 7 h du matin, ainsi que le samedi après 13 h et le dimanche toute la journée se voient attribuer le symbole « AUTH_NOT_WORK_HOURS » et un score de +0,1.

Jusque-là rien d'anormal, nos utilisateurs ont tout à fait le droit d'envoyer des mails depuis chez eux en dehors des heures de travail.

Par contre nous allons pouvoir, grâce à ce nouveau symbole, écrire une règle composite qui attribuera un score supérieur lorsque le comportement sera suspect.

Imaginons que nous ne souhaitons pas autoriser l'envoi de message en SMTP authentifié vers de faux destinataires en dehors des heures de travail. Pour cela nous créons une règle composite dans le fichier */etc/rspamd/local.d/composites.conf* en ajoutant ceci :

```
AUTH_NOT_WKHOURS_FORGED {
    expression = "AUTH_NOT_WORK_HOURS and FORGED_RECIPIENTS and
FREEMAIL_RCPT and not (SAFE_COUNTRY | RFC1918)"
    score = 10.0;
}
```

SAFE_COUNTRY est un fichier *multimap* qu'il est possible d'alimenter directement depuis l'interface web de Rspamd et qu'il faut préalablement déclarer dans le fichier */etc/rspamd/local.d/multimap.conf* comme ceci :

```
SAFE_COUNTRY {
    type = "country";
    map = "${DBDIR}/safe_country.map";
    description = "Countries that can be considered safe";
    score = -0.1;
}
```

Le tableau de bord Kibana permet de voir très rapidement le compte qui a été piraté. Nous avons une procédure qui permet de changer rapidement le mot de passe de ce compte et nous lui interdisons de poster en l'ajoutant dans le fichier *multimap banned_users* afin d'éviter qu'une connexion encore ouverte sur le serveur SMTP ne soit utilisée pour continuer à envoyer du spam ou des messages de phishing.

```
local banned_users_map = rspamd_config:add_map({
   type = 'set',
   url = '/var/lib/rspamd/banned_users.map',
   description = 'List of UIDs to be banned for authenticated
```

```
SMTP'
})



rspamd_config.BANNED_USER = {
   callback = function (task)
     local user = task:get_user()
     if not user then return end
     if not banned_users_map then return end
     local localpart, domain = string.match(user, '(.+)@(.+)')
     if not localpart then localpart = user end
     if banned_users_map:get_key(localpart) then
       task:set_pre_result('reject', 'You are forbidden')
       return true
     end
   end,
   type = 'prefilter',
}
```

La règle BANNED_USER étant en « prefilter » aucune analyse ne sera effectuée sur le message, l'expéditeur obtient instantanément le message « You are forbidden ».

Si vous gérez plusieurs domaines de messagerie avec votre cluster Rspamd, il sera nécessaire de renseigner les utilisateurs à bannir comme ceci uid@domain.fr dans le fichier *multimap*.

Ce petit exemple permet d'imaginer facilement l'étendue des possibilités offertes par l'écriture d'extension dans le langage lua.

5 Mutualisation

Pour l'instant nous n'avons mis en place Rspamd que pour gérer le filtrage anti-spam de notre domaine de messagerie, mais Rspamd permet de mutualiser la plateforme de filtrage pour plusieurs sites, plusieurs domaines de messagerie.

La communication entre le MTA et Rspamd via le module proxy peut être chiffrée et compressée. Assurant ainsi un gain et une confidentialité des données, lors de la mise en place d'une solution Rspamd mutualisée.

Le seul bémol concernant la mutualisation se situe au niveau de l'interface d'administration. Celle-ci n'autorise pas pour l'instant la gestion de plusieurs utilisateurs sur des domaines différents. En pratique un seul compte ayant tous les droits est utilisé. Il peut donc changer le paramétrage de l'ensemble du cluster.

6 Conclusion

Rspamd, comme toute solution de sécurité, demande un peu de temps d'administration et d'apprentissage.

Sa mise en place n'est pas particulièrement complexe, mais nécessite comme prérequis pour le déploiement d'un cluster, d'avoir les connaissances suffisantes pour l'installation et l'administration d'un cluster Redis.

Tous les messages sont délivrés en moins de 10 secondes, en moyenne moins de 2 secondes, il est donc nécessaire de bien verrouiller les envois de messages via le SMTP authentifié, sinon les spammeurs bénéficieront d'une plateforme très performante pour envoyer leur spam.

Le module « ratelimit » n'est pas aussi complet que ce que peut offrir cluebringer (policyd). Il n'est, par exemple, pas possible de définir des seuils différents pour les messages envoyés en interne et ceux envoyés vers des domaines externes. Par contre, comme ce module est en lien direct avec le mécanisme de score, il est capable d'adapter le quota dynamiquement en fonction de la qualité des messages.

L'utilisation conjointe avec ElasticSearch et Kibana permet d'étudier efficacement l'impact de nouvelles règles. Notamment les règles composites que l'on peut modéliser directement dans Kibana afin de mesurer son effet et de s'assurer qu'elles n'engendrent pas de faux positifs. ElasticSearch avec son interface Kibana est le partenaire idéal pour un usage efficient de votre solution anti-spam.