

# RSF Quickstart Guide

Ronald L. Rivest

September 1, 2004

## 1 What is RSF?

RSF (“Ron’s Spam Filter” or “Ron’s Spam Firewall”) is a free Python program for defeating spam. This document is a “quickstart” introduction to RSF.

- *RSF blocks email by default*—an email message is placed in your mailbox only when it is allowed by one of your RSF rules. RSF thus differs from standard spam filters such as “Spam Assassin” that by default allows all messages except those that can be identified as spam. RSF normally runs after such a filter has already removed most of the spam.
- *RSF maintains a ruleset (rule database) specifying which email is allowed.* You can create an initial ruleset from your old (good) email, and edit your ruleset at any time. You can allow new correspondents to create a rule for themselves by responding to a challenge (or “bounce”) message. You may review and edit your ruleset using an RSF console application, or remotely using by sending email to yourself. You may allow your assistant to edit your ruleset.
- *RSF does not require any global infrastructure support*, other than standard email support. It does not require any modification to your mail client program for reading and sending email. Installation is reasonably straightforward.
- *RSF emphasizes ease-of-use and familiarity over cleverness.* RSF has no deep cryptographic theory or clever protocols; its approach is quite straightforward and similar to what others have done. RSF’s most interesting aspect is perhaps the user interface for managing the ruleset. RSF is designed more to be useful than to be a research project.
- *An RSF rule may specify an allowed sender.* For example, the rule:

```
allow from alice@mit.edu
```

allows all email from `alice@mit.edu` to pass through RSF. Two extensions are supported: `allow from @mit.edu` allows all email from the domain `mit.edu` (or any subdomain), and `allow from alice@` allows all email from `alice` on any domain.

- *An RSF rule may specify an allowed password or passphrase.* For example, the rule:

```
allow password cryptography
```

allows all email containing the word “cryptography”, in the subject line or in the body of the message, to pass through RSF. Two extensions are supported:

```
allow password cryptography and conference
```

allows all email containing both “cryptography” and “conference”, while

```
allow password apple and -ipod
```

allows all email containing “apple” but not “ipod”. The keywords `from`, `password`, and `and` are just syntactic sugar, and can be omitted, as they will be in future examples.

- An *RSF* rule may specify both an allowed sender and a password. For example, the rule:

```
allow alice@mit.edu cryptography
```

allows email from `alice@mit.edu` if it contains the word “`cryptography`” in the subject line or body; other email from `alice` is not allowed by this rule. There may be several rules for `Alice`, with different passwords.

- *That’s it.* The expressive power of *RSF* is limited to what you can allow using sender email specifications and passwords. There is no specific support for mailing lists or encrypted email. By design there is no way to specify senders or passwords that should be blocked; *RSF* adheres firmly to an “allow-only” policy—a user’s involvement in spam reduction should focus on what the user wants to see rather than what the user doesn’t want to see.

## 2 Blocked mail, bounces, and quarantine

When an email to you is not allowed by any rule, it is *blocked*. A blocked email is placed in *quarantine*, waiting for a change in rules that would allow it to pass. If it is not allowed within a fixed period (nominally 30 days), it is quietly deleted from the quarantine.

When a message to you is blocked, the sender is normally sent a “bounce message” notifying him that the message was blocked and is being held in quarantine. Normally, the sender is asked just to reply to the message. This causes the blocked message to be released from quarantined and delivered to you, and it also causes a rule to be added to your *RSF* database allowing the sender to send you email in the future.

A given sender is normally limited to receiving at most one bounce message a week. This limit prevents problems should your bounce message to him elicit a bounce message in return.

When you send email to someone to whom you’ve never sent email before, you want *RSF* to be ready to receive a reply or even a bounce message; you don’t want *RSF* to block that reply or bounce message. You can do this, albeit somewhat awkwardly, by augmenting your ruleset before sending that first message. You can also install *RSF* in such a way that the ruleset is automatically augmented as necessary to allow replies and bounces when you send outgoing mail.

## 3 *RSF* management

While *RSF*’s default settings are probably fine for most purposes, you’ll need to use the *RSF* management tools to set up *RSF* for the first time. You may also want to tune your settings and ruleset later on.

*RSF* provides a rich and flexible command language for reviewing and updating your ruleset, control settings, and quarantine. Typical commands are “`show rules`”, which displays all the rules in your ruleset, and “`allow from rivest@mit.edu`”, which adds a new rule to your ruleset.

There are four ways to execute *RSF* commands:

- With *interactive mode* using the *RSF* console applications, which you can start up by typing `rsf` or `rsf.py` to your operating system command interpreter. *RSF* will start up and prompt you for commands to execute.

- Using *command-line mode*, you can execute a single RSF command by typing it as command line arguments to `rsf`, as in `rsf show rules`. RSF will execute that one command, and then return. This is useful when you wish to save the output of an RSF command into a file (e.g. with a command such as “`rsf show rules >saved.rules`”).
- Using *redirection* (i.e. *piping*). You can execute a file `abc.txt` of RSF commands via “`python rsf.py <abc.txt`”. (Warning: you need to explicitly say `python` in Windows; it fails due to a known Windows bug if you try to say just `rsf.py <abc.txt`.)
- Using *email mode*. You can operate RSF remotely using email. Just send an email to yourself whose subject line is “`rsf password`”, where *password* is your RSF password, and where your RSF commands are given in the body of the message. The output of these commands will be returned to you by email. Your RSF password is established when RSF is installed.

## 4 Starting off

The initial installation of RSF is not described here; if you need to install RSF see the installation instructions. Note that the operating system environment variable `RSF_OWNER_EMAIL` must be set to your email address before RSF can operate.

Once RSF is installed, you may wish to have RSF derive an initial ruleset from your saved archive of email. Suppose you have a large of saved email in the directory or file “`/myhomedir/email`”. If you execute the RSF command “`makerules /myhome/email`”, RSF will examine your archive of saved email and create one rule for each email address found there. If you have corresponded with “`joe@abc.com`”, a rule of the form `allow joe@abc.com` will be created and added to your ruleset. Rules will be created only for those you have corresponded with within the last two years.

## 5 Useful commands

Here are some useful RSF commands to get you started.

`help` – See the starting help page; RSF has an extensive help system.

`allow rivest@mit.edu` – Create a rule allowing any email from `rivest@mit.edu`, and release from quarantine any email from him.

`allow cryptography` – Create a rule allowing any email containing the word “`cryptography`”, and release from quarantine any email containing this word.

`allow rivest@mit.edu cryptography` – Create a rule allowing any email from `rivest@mit.edu` containing the word “`cryptography`”, and release from quarantine any email meeting these conditions.

`show rules` – Show all your rules.

`show rules rivest@mit.edu` – Show all rules with sender `rivest@mit.edu`

`show quarantine` – Show the sender, date, and subject line for each message in quarantine.

`show quarantine rivest@mit.edu` – Show sender/date/subject for quarantined messages from `rivest@mit.edu`

`show quarantine conference` – Show sender/date/subject for quarantined messages containing “`conference`”.

`release` – Release all messages from quarantine (i.e., put them in your in-box).

`release joe@abc.com` – Release from quarantine all messages from `joe@abc.com`.  
`release "conference"` – Release from quarantine all messages containing "conference".  
`delete rules joe@abc.com` – Delete all rules allowing email from `joe@abc.com`.  
`show variables` – Show the names and values of all RSF control variables.  
`show variable variable_name` – Show the value of the variable with the given name.  
`set variable value` – Set the given *variable* to the given *value*.  
`quit` – Quit.

RSF rules may be organized into disjoint *groups*; the default group is named "". The command `allow berries group food` creates a new rule with password `berries`, and places this rule in group `food`. The command `show group food` shows all rules in group `food`. The command `delete rules group food` deletes all rules in group `food`. The command `show groups` lists each group's name and size.

Each rule is stored with eight attributes: `from password group created times_used last_used expires aka`. To see them all when using the `show` command, do

```
set rule_format "from password group created times_used last_used expires aka"
```

Here "aka" gives the "real name" of the sender, e.g. "Ronald L. Rivest" for `rivest@mit.edu`. The "show" command is quite flexible: you can "show created 2004" to see all rules created in 2004, and "show aka ron" to see all rules whose `aka` attribute contains "ron" (case-insensitive), as well as combinations. The "delete rules ..." command can be used to delete any set of rules that "show ..." can show. The "copyto group *groupname*" and "moveto group *groupname*" can be used to copy or move the most recently set of rules shown into the named group. In general, `copyto` and `moveto` can be used to change any attributes: the command `show joe@abc.edu` followed by `moveto joe@mit.edu` can be used when joe moves to MIT.)

## 6 Levels of protection

Used most simply, RSF has one rule for each allowed sender; passwords aren't used. RSF creates a rule for a new correspondent when you send him email, or when he replies to a bounce message.

However, spammers often forge email pretending to be from a friend or associate of yours. If this happens, you may start to utilize passwords on your RSF rules. You may require that correspondents establish a password in a reply to a bounce message. (You can even require that they contact you or your assistant personally in order to obtain a password.)

## 7 Trouble-shooting

If a piece of spam gets through RSF, you can examine the "X-RSF" header of that email to see which RSF rule(s) allowed that email to pass. You may then wish to edit your ruleset.

The RSF databases are stored in human-readable form; you can examine them or edit them directly, although this shouldn't be necessary. RSF also maintains multiple backup copies (of varying ages) of your databases, so it is likely that you will be able to recover if your database is damaged from a mistyped command or other accident. (The backups are easy to find...)

You may be able to find more information on RSF at:  
<http://theory.lcs.mit.edu/~rivest/rsf/index.html>.

You may also be able to examine or edit the code to identify or fix problems. I've tried to make the code reasonably clean and readable.

I hope you find RSF useful in defeating spam!