Manual version 1.5 [2006-03-19]

http://www.sysdesign.ca/

For renattach version 1.2.3

Primary distribution site: http://www.pc-tools.net/

Although this software has been discontinued and there is no longer any active development, this manual is provided to help existing users or those who wish to modify the software.

Table of Contents

1. Background

1.1 Introduction

****
WARNING:
THIS SOFTWARE HAS BEEN DISCONTINUED. IT IS NO LONGER MAINTAINED.
****

The author recommends that you do not depend upon renattach to filter emails for dangerous content. As of 2006, renattach used on its own is not enough to filter potentially harmful emails. Dangerous attachments, or other attacks, may pass through the filter undetected. Please switch from renattach to some other actively developed security system.

Jem E. Berkes
2006-03-19

****

renattach is a simple filter designed to process RFC 822 messages and alter potentially dangerous file attachments to minimize the risk of virus/worm damage at an end-user's mail client. The software compiles and runs under practically any UNIX-like platform, including Linux and BSD. The intention is that the software will be installed into the mail system and provide virus protection to end-users (particularly those running Microsoft Windows). The software provides some additional benefits to the mail server: by dropping virus-infected messages it can alleviate resource burdens during periods of floods. Logging of filtered mail also helps identify points of abuse (whether internal or external).

renattach can be used to filter mail on small and large sites alike, without requiring a second dedicated virus scanning host as is often needed with resource-hungry virus scanners. Unlike other similar filters that are implemented with Perl or shell scripts, renattach is written in pure C and compiles into a fast stand-alone binary that runs with minimal overhead. Furthermore, because renattach can be used as a simple stand-alone command, it can be easily installed into any mail system that provides a facility to pipe mail to an external command.

However, renattach is definitely not as thorough as a true virus scanner when it comes to locating malicious content in emails. **Please be aware that renattach is not a virus scanner**, as it does not analyze the actual attachment content (beyond a quick check for an executable header). Nevertheless, renattach has proved to be surprisingly effective.

1.2 Modifications to mail

renattach alters message content (obviously, this is fundamental to the software). This tinkering sometimes causes noticeable problems, for example with PGP MIME parts. The following warning appears at the very start of the renattach.c source code:

> *Warning: renattach 'breaks' MIME because it rewrites MIME headers! Whenever a MIME attachment with filename is encountered, MIME headers are rewritten to a safe format (even if filenames are unchanged). MIME headers that aren't attachments with filenames are left alone.*

Care has been taken to make as few changes as possible to the essential MIME structure of the message. Only parts of the message that deal with file attachments are actually modified. Here is a summary of the changes that may be made to an email message.

- Attachment filenames (MIME and uuencoded) are renamed when filtering occurs
- MIME attachments are assigned new Content-Type fields when filtering occurs
- MIME file attachment headers are always rewritten with Content-Type, Content-Disposition, and Content-Transfer-Encoding fields. The Content-ID field *is dropped* unless the pass_contentid option is set; see Section 4.1.
- Attachments' encoded bodies are eliminated when the delete action is used
- The message Subject is modified when filtering occurs
- Warning text may be added to the plain text and/or HTML parts of an email, if desired

Note that renattach fully interprets every attachment filename in order to determine whether the attachment poses a risk. The software has built-in routines to decode RFC2047 and RFC2231 encoded filenames. The filenames are re-encoded back to the original format so that the end user doesn't notice a difference. Even though RFC2047 encoding shouldn't be used for attachment filenames, renattach will write back the filename in that format when necessary to mimic the behaviour of the original mail client.

An email that has no file attachments is unaltered by renattach, except for new message headers that are conventionally added by any email filter. The filter always adds the following headers to messages:

X-Filtered-With: renattach *version*
X-Renattach-Info: mode=*mode* action=*action* count=*filter_count* [(filtered)]

The mode and action are defined by the renattach command-line. The filter_count is a total count of all filtering changes made to the message. This filter_count may be more than 1 even if only a single attachment was caught, because that attachment may be both renamed and deleted. In addition, when filter_count is greater than 0, **(filtered)** appears in the header line to facilitate custom processing of messages that were filtered due to their attachments.

If the X-Filtered-With or X-Renattach-Info headers already existed, they are renamed to Old- to make message tracing easier.

1.3 Impact on resources

Because renattach is a compiled C program (as opposed to a shell script, Perl, Python, or Java program), it consumes negligible resources and is definitely one of the most efficient filters available. The strengths of the software are in speed and memory usage. Table 1 summarizes the resource usage:

Resource	***Typical use***	***Impact on system***
Process ID (pid)	1 pid per renattach instance 1 more pid if invoking external --pipe	Minimal (execution time is brief). The forking on --pipe may become a concern if filtering on the order of millions of messages per hour.
Memory	Constant, small amount. Very low memory overhead (compiled C, and only the essential standard libraries).	Very minimal.
CPU time	Very little processing time is required per message. Even a weaker desktop PC (1 GHz x86) can process about 500 messages/second.	Very minimal.
Disk space	tmpfile() used to buffer each message. Note that this does not actually generate a file system entry on modern OS's.	Minimal, and in most cases the operating system will not even touch the disk.

Table 1: Resource usage

Note that even searching for filenames within ZIP files is done efficiently, without invoking external software. Filtering ZIP attachments requires more processing time when the search_zip option is enabled (see Section 4.1) but the extra CPU time is negligible. Enabling search_zip *does not* require any extra pids, memory, or disk resources.

2. Compiling and Installing

2.1 Original source

renattach is developed solely by Jem Berkes, and the source code is currently distributed from www.pc-tools.net but mirrored in several other places. While there are currently renattach packages available for several platforms (Linux, FreeBSD, OS/2) these are made available by volunteers in the community and are not supported by the original developer.

The source code available at www.pc-tools.net is accompanied by both MD5 and PGP (GnuPG) signatures, and you should verify these signatures to confirm that you have an undamaged copy of the source. The official source code release will have a GPG detached signature, and during verification you should see the following:

```
$ wget http://www.pc-tools.net/files/unix/renattach-1.2.3.tar.gz
$ wget http://www.pc-tools.net/files/unix/renattach-1.2.3.tar.gz.md5
$ wget http://www.pc-tools.net/files/unix/renattach-1.2.3.tar.gz.asc
$ md5sum -c renattach-1.2.3.tar.gz.md5
renattach-1.2.3.tar.gz: OK
$ gpg --verify renattach-1.2.3.tar.gz.asc
gpg: Signature made Mon Mar 20 04:10:50 2006 UTC using DSA key ID 18761408
gpg: Good signature from "Jem E. Berkes (SysDesign) <jberkes@sysdesign.ca>"
gpg:                 aka "Jem E. Berkes (PC9) <berkes@pc9.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:         There is no indication that the signature belongs to the owner.
Primary key fingerprint: C410 709A 0B4E 44B0 DBFB  26CC 3642 3893 1876 1408
$
```

2.2 Using configure and make

Note: For FreeBSD, use the port to simplify install and uninstall.

In order to simply the build process, renattach comes with autoconf and automake scripts that can automatically adapt the build process to your specific environment. If you use the “./configure” command, renattach will be configured with default paths. You can customize these paths by passing options to the configure script, summarized in Table 2. Another noteworthy option is “--prefix”, which lets you place the entire installation under a new directory tree (for example, “./configure --prefix $HOME”).

Software component	***Default path***	***configure option to customize***
renattach (binary)	`/usr/local/bin`	`--bindir`
renattach.conf.ex (configuration)	`/usr/local/etc`	`--sysconfdir`
renattach.1 (man page)	`/usr/local/man`	`--mandir`

Table 2: Locations of software components

After the configure script has completed successfully, use the 'make' command to compile the software and watch for error output. The entire compile process may proceed as follows, starting with the .tar.gz source package:

```
% tar zxvf renattach-1.2.3.tar.gz
% cd renattach-1.2.3
% ./configure --sysconfdir=/etc
% make
```

Once the software is compiled, the resulting binary is under src/ and can be copied anywhere you wish. It's preferable to take advantage of the automated install and uninstall provided by autoconf and automake. To install the software into the configured paths, use “make install”. To uninstall, you can use “make uninstall”.

Note that the file “renattach.conf.ex” is an example configuration file. You only need to create a “renattach.conf” file if you want to set options to non-default states. You may want to do “mv renattach.conf.ex renattach.conf”.

Please see Section 4.1 for a description of all options in renattach.conf.

3. Command Usage

3.1 Understanding filter operation

Each instance of renattach is independent since the software does not run as a daemon. Whenever the program is executed, the command-line options define a specific **filter mode** and a **filter action**. In addition to this, each program instance has various **filter options** that are defined by the .conf file. Note that because this .conf file location itself can be specified on the command-line, each program instance can be made to act differently based on the command-line invocation. Tables 3 and 4 describe the filter modes and actions.

Filter mode	***Effect***
All	Filter action is applied to every file attachment.
Badlist **(Default)**	Filter action is applied to any attached file whose file extension is on the badlist. This badlist is defined in the .conf file (default list of dangerous extensions is compiled into the software). Use this to filter only EXE, PIF, SCR ... files.
Goodlist	Filter action is applied to any attached file whose file extension is *not* on the goodlist, also defined in the .conf file. This is more aggressive because it only allows specific filenames by extension (DOC, TXT, JPG ...) while filtering everything else.

Table 3: Filter modes

Filter action	***Effect***
Rename **(Default)**	If an attachment is filtered, the filename is modified to a format that makes it difficult to open/execute accidentally. All periods in the filename are changed to underscores, and the new_extension is appended. The MIME Content-Type field is also changed to new_mime_type (see Section 4.1 for definitions of these new values). This combination makes it unlikely the attachment can be accidentally opened/executed.
Delete	If an attachment is filtered, the filename is first renamed exactly as above. Additionally, the body of the attachment (actual file contents) is dropped entirely. The attachment still exists in the email, but it now has 0 size.
Kill	If any attachment in the message is filtered, the entire email will be dropped. This is accomplished by outputting nothing to stdout. If the output is to be piped to an external command, that external command is never invoked.

Table 4: Filter actions

It is important to be aware of the current filter mode and action. Examine filtered message headers and observe the filter mode and action indicated in the **X-Renattach-Info** header.

3.2 Command options and exit codes

Usage: **renattach** [OPTIONS]

-a, --all
Filter mode: Match all attachments. See Table 3: Filter modes

-b, --badlist
Filter mode: Only match filenames that have extensions listed on the badlist. This will match only attachments with known dangerous file extensions (default). See Table 3: Filter modes

-c, --config filename
Use the specified configuration file. Run renattach with **--settings** to verify current settings.

-d, --delete
Filter action: Delete attachment body after renaming headers. See Table 4: Filter actions

-e, --excode
Extend exitcodes with a new code, 77=filtering occurred. See below for standard exit codes.

-g, --goodlist
Filter mode: Match all attachments except those with extensions on the goodlist. See Table 3: Filter modes

-h, --help
Show help, explain options.

-k, --kill
Filter action: Kill (absorb) entire email. See Table 4: Filter actions

-p, --pipe command [args]
Instead of writing output to stdout, open pipe to command (with args) and send output there. This program must return with exit code 0. This must be the last option on the command line.

-r, --rename
Filter action: Rename matching attachments (default). See Table 4: Filter actions

-s, --settings
Show current settings/configuration and terminate.

-v, --verbose
Write verbose output (including settings) to stderr.

-V, --version
Display software version and terminate.

Exit codes:

0 Success (filtered mail and wrote output)
75 Temporary failure (resource shortage; failed to write to pipe if using **--pipe**)
255 Critical failure (improper parameters; bad .conf file)

The temporary failure code allows MTAs to re-queue mail for later delivery. These exit codes are compatible with BSD-style mailers, and --excode should *not be used* without good reason because it returns a non-success code when the filter “catches” something.

3.3 Usage examples

```
cat message.in | renattach > message.out
diff message.in message.out
```

You can run this example from the shell to gain an understanding of the filtering process. If you have a plain text email in message.in, this saves a filtered copy to message.out. The 'diff' command will show you what has been changed during filtering.

```
renattach --goodlist --delete < message.in > message.out
```

This runs renattach in goodlist mode with the delete action. As with the first example, the input is read from message.in and the output is written to message.out. Note that the output now indicates different mode and actions in the X-Renattach-Info header.

```
cat /var/mail/user | formail -s renattach >> outbox
```

This command uses formail to run renattach on each message in the user's mailbox. The output is appended to a new mailbox, “outbox”.

The next few examples assume that renattach is reading messages from stdin. For example, you can accomplish this by making a procmailrc rule that pipes mail to renattach.

```
renattach --pipe /usr/sbin/sendmail -i user@domain
```

This tells renattach to pipe its output to the command “sendmail -i user@domain” (don't include quotes) rather than stdout. The filtered mail can therefore be sent to a new address.

```
renattach --kill --pipe /usr/sbin/sendmail -i user@domain
```

Same as the last example, except that the filter action is now 'kill'. If the input message contains an attachment that would be filtered by the badlist (default mode), then the sendmail command is never executed. The mail is absorbed from stdin, but not written anywhere.

When debugging, use the -v or --verbose option to see extra output. This is particularly useful for debugging -p or --pipe operation since verbose output shows the exact command+args used for the pipe. *Specify -v before -p*, since the pipe command must be the last option on the command line. You can also use the --settings option to view the effective configuration of a renattach instance.

4. renattach.conf

4.1 Configuration directives

The renattach.conf file (or any other configuration file specified by the **-c** option) should be a plain text file with one configuration directive per line. Settings are specified in the form: `directive = value[,value2,value3...]`

Comments preceded by # will be ignored. Some directives may only appear once, while others are additive. The additive directives can be specified multiple times for a concatenation effect. The conf file and all directives are optional, as adequate defaults are compiled into the software.

> **NOTE:** Please run **renattach --settings** to verify your configuration! All changes you make to renattach.conf should be visible in the output.

Table 5: Configuration directives

Directive	***Default value***	***Usage and notes***
delete_exe	yes	Accepts: **yes**|**no** or alternatively **1|0** Delete executable binary attachments by signature. renattach looks for encoded bytes that identify DOS/Windows executables ('MZ'). If an executable is found, the encoded attachment will be removed while the MIME header remains unchanged. This is a feature that works independently of filename-based filtering, designed as a backup. The net effect is that encoded executables are deleted. Email clients will see 0-byte attachments.
kill_exe	no	Accepts: **yes**|**no** or alternatively **1|0** Kill executable binary attachments by signature, as in the previous directive. Note that delete_exe and kill_exe are mutually exclusive. The kill action entirely drops the email. Nothing is written to stdout, and if using --pipe, the external command is never executed.
search_zip	no	Accepts: **yes**|**no** or alternatively **1|0** Search for filenames within ZIP archives using the internal ZIP parsing engine (no external software required). Any filenames found are subject to the same checks, for instance badlist or goodlist, with the notable difference that the *rename action has no effect on ZIP files*. This is due to a shortcoming in this version of the filter; by the time the ZIP file is decoded, it is “too late” to rename the attachment. Only the delete or kill actions will modify ZIP files. In order for search_zip to have an effect, you must either: • run renattach with --delete or --kill (instead of default rename) • or, use badlist mode (default) and specify in the badlist the /k or /d actions for select extensions. See below for badlist directive.

Directive	*Default value*	*Usage and notes*
pass_contentid	no	Accepts: **yes**|**no** or alternatively **1|0** Normally, MIME Content-ID fields are dropped during filtering due to their application-specific use and security risk (recently used by worms to link malicious code to embedded images). If you are sure you want to pass Content-ID fields unfiltered, enable this option. The author *does not* recommend enabling this option.
full_rename	yes	Accepts: **yes**|**no** or alternatively **1|0** Normally, all periods in filenames are replaced with underscores during renaming. Although this is the recommended mode, you can also disable full renaming if you only want the last period to be changed to an underscore.
use_syslog	no	Accepts: **yes**|**no** or alternatively **1|0** If enabled, all filtering actions will be logged via syslog. renattach logs with priority 'warning' to facility 'mail' so the log entires end up in the same place as your MTA's logs.
generic_name	filename	Accepts: ***string*** A generic filename to use when parsing fails. Since renattach rewrites all attachment headers, it's possible that corruption, lack of buffer space, or some other problem will prevent filenames from being recreated. In such a case, this generic name is used.
new_extension	bad	Accepts: ***string*** A replacement file extension to use when changing dangerous attachment filenames. This extension is appended to the previous one. For instance virus.pif becomes virus_pif.bad. Specify just # to leave the extension as is, and not rename it.
new_mime_type	application/unknown	*Accepts:* ***string*** When attachments are renamed, the MIME type is also changed to this new_mime_type for safety. This is vital because many mail clients take the Content-Type rather seriously and this field may be considered as important as the file name itself.
The following directives control how the message Subject is modified to inform the user that filtering has occurred. They have the following ORDER OF PRIORITY (starting with highest): subj_banned, subj_exec, subj_deleted, subj_renamed, add_subject By default, only add_subject is defined so any condition (whether it's a ban, executable match, delete, or rename) results in the same Subject addition. If you also define subj_exec then there could be a different Subject if an executable was caught (since it has higher priority than add_subject). Another alternative for these options is to specify the single character # to suppress Subject modification for that condition. You could use this to be quiet in case a banned attachment is caught. You can also use # to turn off add_subject, hence NEVER modify the message Subject.		
subj_banned		Accepts: ***string*** or special value # to suppress Add text to Subject if an attachment is caught by banned_files. Undefined by default, so renattach falls through to the next valid subj_ directive, and eventually to add_subject.
subj_exec		Accepts: ***string*** or special value # to suppress Add text to Subject if an attachment is caught by delete_exe. Undefined by default, so renattach falls through to the next valid subj_

Directive	*Default value*	*Usage and notes*
		directive, and eventually to add_subject.
subj_deleted		Accepts: ***string*** or special value # to suppress Add text to Subject if an attachment is deleted for any reason. Undefined by default, so renattach falls through to the next valid subj_ directive, and eventually to add_subject.
subj_renamed		Accepts: ***string*** or special value # to suppress Add text to Subject if an attachment is renamed for any reason. Undefined by default, so renattach falls through to add_subject.
add_subject	[filtered]	Accepts: ***string*** or special value # to suppress Add text to Subject if an attachment is filtered in any way. This has lowest priority, and is only used if the previous subj_ options are undefined. To prevent the Subject from ever being modified, make sure the previous subj_ directives are undefined and use only: add_subject = #
htmlwarn_pos	html, body	*Accepts:* ***case insensitive list*** *with comma/space delimiter* When inserting a warning into HTML parts of messages (warning_html), this tag defines the preferred position to insert the new HTML. If the first tag in the list is found, the warning position is placed just after this tag. As subsequent tags are found, the position advances after each. With the provided default, the warning_html is inserted after the <body> tag if there is also an <html> tag before it. If there is no <body> tag, the warning is inserted after <html>. If there is no <html> tag at all, then the warning is just inserted at the start of the HTML message part.
warning_text		Accepts: ***string*** (additive) If an attachment is filtered, this lets you specify some warning text that will be inserted into any plain text portion(s) of the email. This is effective for informing users of filtered files, but the act of inserting arbitrary text into an email can cause new problems. Use with caution.
warning_html		Accepts: ***string*** (additive) Inserts a warning message into HTML portions of the email when filtering occurs. The HTML is inserted at a position determined by htmlwarn_pos (see above) which provides a good hope for adding a visible warning. Unfortunately, inserting arbitrary HTML is tricky due to the complexity of markup interactions. Inserting warnings in HTML may thoroughly disrupt the original message, so use with caution.
add_header		Accepts: ***string*** (additive) When enabled, these arbitrary new headers will be added to the message to inform the user about filtering that occurred. Each add_header directive will result in a new header line being added. Make sure that each line defines a *different* header field. Example: add_header = X-Notice-0: * PLEASE NOTE * add_header = X-Notice-1: We removed dangerous attachments from add_header = X-Notice-2: this mail, as per our Terms of Service.

Directive	*Default value*	*Usage and notes*
banned_files		Accepts: ***case insensitive list*** with comma/space delimiter (additive) Catch specifically named, banned attachment filenames and optionally take an action (r=rename, d=delete, k=kill). If the name begins with a forward slash ('/'), this substring has to be found; '/foo' matches 'foobar' and 'eatfoo'. Otherwise, the whole name has to match. To specify an action on matching filename, append /r (rename), /d (delete), or /k (kill) to the filename as in: your_details.zip/k
goodlist	DOC, PDF, RTF, SXC, SXW, TXT, ZIP	Accepts: ***case insensitive list*** with comma/space delimiter (additive) A list of good (known-safe) attachment file extensions to use in goodlist filtering mode. This makes filtering quite strict because any file type that is not on this list is filtered according to the current action.
badlist	ADE, ADP, BAS, BAT, CHM, CMD, COM, CPL, CRT, EML, EXE, HLP, HTA, HTM, HTML, INF, INS, ISP, JS, JSE, LNK, MDB, MDE, MSC, MSH, MSI, MSP, MST, NWS, OCX, PCD, PIF, REG, SCR, SCT, SHB, SHS, URL, VB, VBE, VBS, WSC, WSF, WSH	Accepts: ***case insensitive list*** with comma/space delimiter (additive) A list of bad (known-dangerous) attachment file extensions to use in badlist filtering mode. To specify an action for an extension, append /r (rename), /d (delete), or /k (kill) to the filename. This overrides the default action for the filter and can be used to provide special handling for some extensions. An additional switch can be used to specify an action *only for files found within ZIP archives*. For instance, EXE/k/d tells the filter to kill emails containing EXE attachments, but if the EXE was found inside a ZIP then the attachment is deleted, not killed. badlist = ADE, ADP, ... , EXE/k/d, SCR/k, PIF/k, ... Note that the default list is quite extensive. If you specify your own list, be sure to list all dangerous extensions.

A note regarding the **--settings** option: to help you see how individual entries in ***lists*** are parsed, renattach displays the list entries separated by vertical bars (|). Verify the output to ensure that your lists are properly parsed. For example, a goodlist with 7 entries:

```
goodlist: {DOC|PDF|RTF|SXC|SXW|TXT|ZIP}
```

Similarly, when using **--pipe**, the verbose output will show the argv[] list for the external command to execute. You should be certain that the parameters are being parsed as you expect. For example, when opening a pipe to the sendmail command, **--settings** may show:

```
Writing output to:
  {/usr/sbin/sendmail|-i|-f|sender@domain|--|recipient@domain}
```

4.2 Example renattach.conf

An example configuration taken from a server handling over 10,000 emails daily is provided below. Renattach would be invoked with its default options (badlist mode, rename action). This is an aggressive configuration that drops (kills) any executable attachments, including those inside ZIP files. No worms reach users.

The kill_exe option alone would catch most viruses and worms, regardless of their filenames. As well, the badlist has been modified with /k (kill switches) for specific attachment types that are most commonly used by viruses. When BAT, COM, etc. attachments are “killed”, nothing arrives at the recipient's account. The downside is that the recipient is not aware of attachments being filtered. However, in today's environments where users can receive hundreds of viruses or worms daily, many sites find it reasonable to eliminate this excessive traffic. Attachments such as EML or HTML are just renamed, not killed.

The other common arrival mode for viruses and worms are through ZIP files. The search_zip option tells renattach to parse ZIP attachments (regardless of the .zip extension) for filenames. If BAT, COM, etc. file extensions are found inside, then the email is killed. Note, however, that if an HTML file arrives within a ZIP file, renattach does nothing – this version of renattach is unable to rename ZIP files after matching a filename inside.

```
# Drop mail carrying executable attachments (DOS/Windows exec signature)
delete_exe = no
kill_exe = yes

# Search for filenames inside ZIP files
search_zip = yes

# Log filtered mail (delete, kill) to syslog mail facility
use_syslog = yes

# Delete winmail (MS proprietary) attachments without modifying Subject,
# also drop emails containing annoying scanner-generated warning bounces
banned_files = /winmail/d, /warn.txt/k, DELETED0.TXT/k
subj_banned = #

subj_deleted = [deleted attachment]
subj_renamed = [renamed attachment]

# When these file types are encountered, rename the attachment (assuming
# filter is invoked with default action=rename). However, kill mail containing
# any BAT, COM, etc. attachments even if they are inside ZIP files. There is
# risk of collateral damage. EML//d means delete ZIPs that contain EML.
badlist = ADE, ADP, BAS, BAT/k, CHM, CMD/k, COM/k, CPL/k, CRT, EML//d, EXE/k
badlist = HLP, HTA/k, HTM, HTML, INF, INS, ISP, JS, JSE, LNK, MDB
badlist = MDE, MSC, MSH, MSI, MSP, MST, NWS, OCX, PCD, PIF/k, REG/k
badlist = SCR/k, SCT, SHB, SHS, URL, VB, VBE, VBS/k, WSC, WSF, WSH
```

5. Installing into mail system

5.1 procmail for individual users

Individual users that are able to pipe incoming emails to a program can take advantage of renattach. The best way to do this is with procmail, which provides a simple method to process every incoming email through a filter command.

If your site uses procmail, then you can create the ".procmailrc" file in your home directory to instruct procmail to run certain scripts. The following basic ".procmailrc" will filter all incoming mail through renattach. Please see "man procmailrc" to learn more about the format of this file:

```
# Filter mail through renattach, and wait for success exit code
:0 wf
| /path/to/renattach
```

There is a minor complication when using renattach's "kill" feature since nothing will leave the filter. Procmail does not know to stop processing, so it will deliver a blank stub of a message to your mailbox. To prevent this, you can add this rule after renattach to handle killed messages:

```
# If message is blank (killed), drop it and stop processing.
:0
*! .
/dev/null
```

5.2 procmail for entire site

If your site uses procmail, then you can take advantage of procmail's site-wide filtering capabilities via the /etc/procmailrc configuration file. You can use the same filtering 'rule' as in Section 5.1. Please see "man procmail" for more information.

Please check to make sure that procmail does not run procmailrc with root privileges. renattach SHOULD NOT run as 'root'.

If you are using Postfix, please see the next section for a better way to install renattach for the entire site. The Postfix integration method has additional benefits over this procmail method.

5.3 Postfix integration

The Postfix MTA has a flexible modular design that makes it relatively easy to add content filters such as renattach. There are several advantages to installing renattach at the MTA level:

- Tighter integration into the mail system, ensuring that all mail is filtered (instead of relying on procmail operating correctly)
- Possibility for both inbound and outbound SMTP filtering
- Superior security, since the filter can run under dedicated UID
- Superior feedback on filtering errors (mail bounces or temporary failures) due to strict interpretation of filter exit codes. In other words, if renattach encounters a temporary error such as resource shortage or dead external pipe, mail will remain queued and delivery will be attempted later. If renattach encounters a fatal error (bad command usage) then mail will bounce.

To install renattach, you only need to modify **/etc/postfix/master.cf**
The following instructions are based on Postfix's FILTER_README file and describe how to install renattach as a content filter for SMTP.

1) Create a new user 'filter' with a disabled password, unique group, no home directory and no login shell. /etc/passwd would have something like:

```
filter:*:952:952:filter:/dev/null:/dev/null
```

2) Insert the following into /etc/postfix/master.cf to define the 'filter' service that uses the Postfix pipe program. Note the long second line; this instructs Postfix to pipe mail into renattach using privileges of the dedicated filter user. You can specify any renattach options you wish. Here, we MUST use renattach's -p (or --pipe) to send the filtered output to the Postfix 'sendmail' command. This re-injects the filtered message back into the mail system. Do not include quotes around the sendmail command line (this has changed in version 1.2.2).

```
filter    unix  -       n       n       -       -       pipe
  flags=q user=filter argv=renattach -p sendmail -i -f ${sender} -- ${recipient}
```

The pipe option must be the last option specified, with everything after it being taken as the command and arguments. Be sure to specify the full path to the renattach and sendmail commands.

3) Now that you have defined a new 'filter' module, you can instruct the smtp module to use 'filter' as a content filter. This is done simply by editing /etc/postfix/master.cf and adding the `-o content_filter=filter` option to smtpd. This means that the smtp service should appear like this:

```
smtp      inet  n       -       n       -       -       smtpd
      -o content_filter=filter
```

4) Save master.cf and run 'postfix reload'. Now test your configuration. Any mail coming into your system via SMTP should now leave with the additional renattach headers. The system mail logs should indicate mail being passed with relay=filter. There should also be log entries from the postfix/pipe program. Make sure use_syslog is enabled so that your mail logs show filtered attachments and message-id's. Send some dangerous attachments to confirm that they are filtered and logged. Please refer to Section 6.1.

It's important to understand how mail now flows through your system. The following illustrates step-by-step how mail is processed with the above configuration:

a) Client connects to server:25 (smtpd) and uploads an email
b) smtpd relays the mail to the content filter, the 'filter' service
c) The 'filter' service pipes the mail to renattach
d) renattach pipes its output to the Postfix sendmail command, specifying as command line options the original sender and recipient(s). If that sendmail command fails in any way, renattach's exit code tells Postfix to keep the mail queued and retry later (no mail is lost).
e) The sendmail command locally re-injects the mail into the system
f) Delivery continues as normal

Gracefully uninstalling the renattach content filter:

Once mail is received by smtpd it will always be sent to the filter service. Therefore, abruptly removing the filter service can prove disastrous if mail is still queued for the filter service. The proper way to remove content filtering is as follows:

1. Remove `-o content_filter=filter` from smtpd in master.cf, then 'postfix reload'.
2. Any new mail is no longer sent to the filter service. Wait until previously queued mail is cleared (through the filter service).
3. When sufficient time has passed and there is no longer any mail queued for content filtering, you can safely remove the filter service from master.cf. Again, 'postfix reload'.

5.4 Sendmail integration

While the procmail-based methods described earlier are probably the best way to install renattach on a sendmail server, it is also possible to install renattach as the local delivery agent (Mlocal option). In this configuration, renattach sits between sendmail and procmail or another local delivery agent such as mail.local.

Installations typically rely on procmail. The following example from a Linux system shows a standard configuration (from sendmail.cf) before renattach is installed:

```
Mlocal,     P=/usr/bin/procmail, F=lsDFMAw5:/|@qSPfhn9,
            S=EnvFromL/HdrFromL, R=EnvToL/HdrToL, T=DNS/RFC822/X-Unix,
            A=procmail -t -Y -a $h -d $u
```

Instead of invoking procmail as the local mail delivery agent, sendmail can invoke renattach and instruct the filter to pipe its output to procmail (or any other program, as required). Few changes are required to the above configuration:

```
Mlocal,     P=/usr/local/bin/renattach, F=lsDFMAw5:/|@qSPhn9,
            S=EnvFromL/HdrFromL, R=EnvToL/HdrToL, T=DNS/RFC822/X-Unix,
            A=renattach --pipe /usr/bin/procmail -f $g -t -Y -d $u
```

Make sure you restart the sendmail daemon after making the configuration changes (either to .cf directly, or using m4). Note the following:

- The F=f (lowercase 'f' flag) must be removed from the flags field, because this automatically places the -f option in the wrong place. The procmail -f option is manually specified after --pipe.
- The procmail -a feature (for passing meta data) appeared to be causing strange problems. Removing it solved delivery problems we encountered.

m4 configuration with mail.local, from a FreeBSD system:

```
define(`LOCAL_MAILER_PATH', `/usr/local/bin/renattach')
define(`LOCAL_MAILER_FLAGS', `PSn9')
define(`LOCAL_MAILER_ARGS', `renattach -p /usr/libexec/mail.local -f $g $u')
```

LMTP (Local Mail Transfer Protocol) warning

While FreeBSD installations and some others are configured by default to speak lmtp with mail.local, renattach does not use this protocol. Therefore, you have to ensure that `FEATURE(local_lmtp)` *disabled*. The 'z' flag in the local mailer flags also has to be removed because it instructs sendmail to speak lmtp with the local mailer. Finally, if using mail.local, make sure you pass it the $u (user) parameter and *not* the -l (lmtp) option from within the renattach pipe command-line.

6. Monitoring filter operation

6.1 Manual checks

Once renattach is installed into your mail system, you should send multiple test messages to observe the effect of the filter. This will help you verify that the filter is having the intended effect. In most cases, any problems with the configuration will be immediately visible after trying these simple checks:

- Messages should now have the new headers (see Section 1.2)
- Messages without attachments should otherwise pass through unmodified
- Messages with (safe) file attachments should have a modified MIME header for attachments due renattach rewriting the values using a known good template
- Messages with (unsafe) file attachments should be caught as dictated by the filter. Please refer to Table 3: Filter modes and Table 4: Filter actions

Observe the new message headers, because these indicate the filter mode and action used by the renattach instance as invoked by your mail system. Remember that each renattach instance is independent, so the behaviour of the filter depends solely upon the command-line specified and the contents of the .conf file used.

If you're not sure how your renattach.conf file is being interpreted, run:

```
renattach -c /path/to/renattach.conf --settings
```

Remember that the current settings as displayed by the above command depend not only on renattach.conf but also on the command-line.

6.2 System logs

If **use_syslog** is enabled in renattach.conf (see Table 5: Configuration directives) then your system's mail logs should include entries from renattach whenever an attachment is caught. These log entries look like this:

```
Dec 22 02:35:19 imdwalrus renattach[6813]: kill "Q323254.exe" in Message-Id: <xxx>
```

The log entry includes the original filename, Message-ID, and filter action that was used (in this example, the message was killed and therefore not delivered). You can periodically check your mail logs for records of attachments being caught:

```
grep renattach /var/log/maillog
```

If you use the 'kill' filter action, you should check your system logs periodically to confirm that the intended messages are being killed. Because killed messages leave no trace at either the sender or receiver, there is the risk of sending users' mail down a black hole.

An additional benefit of logging filtering is to locate users whose infected hosts are trying to send worms or viruses through your mail server. By examining the logs, you can easily locate an infected host and deal with the problem at its source.

For example, the following command can process a Postfix log file and locate the email addresses that are sending mail that gets caught by the filter. Note that this is a very crude technique, and the sender addresses are likely forged.

```
grep renattach /var/log/maillog -B 1 | grep -o '<.*>'
```