

FreePOPs Manual

26th April 2004

Contents

1	Introduction	2
1.1	Usage situations	2
1.2	Features	2
1.3	Plugins	3
2	History	3
3	FreePOPs configuration file	4
4	FreePOPs command line parameters	4
5	Email client configuration	5
6	Plugins	5
6.1	libero.lua	5
6.2	popforward.lua	5
6.3	aggregator.lua	6
6.4	flatnuke.lua	6
7	Creating a plugin	7
7.1	The interface between the C core and a plugin	7
7.2	The interface between a plugin and the C core	8
8	Submitting a bug	9
9	Authors	10
9.1	Developers	10
10	Thanks	10

1 Introduction

FreePOPs is a POP3 daemon plus a LUA interpreter and some extra libraries for HTTP and HTML parsing. Its main purpose is translating local POP3 requests to remote HTTP actions on the supported web-mails, but it is really more flexible: for example there is a plugin to read news from a website as if they were mails in a mailbox. You can easily extend FreePOPs on the fly, without even restarting it; you can add a plugin or modify an existing one simply changing the script file since the plugins are written in LUA and are interpreted on the fly.

1.1 Usage situations

FreePOPs can be useful in some situations, here we give the most obvious ones:

- You are behind a firewall that closes the 110 port but you need to read your mail and the web-mail of your mail provider sucks.
- Your mail provider does not allow you to access your mailbox with POP3 protocol, but only through the web-mail service.
- You prefer looking at your mailbox instead of browsing some websites news.
- You have to develop a POP3 server in less than a week and you want it to be reasonably fast and not so memory consuming.
- You are not a C hacker, but you want to benefit from a fast POP3 server frontend written in C and you want not to waste a month in writing the backend in C. LUA is a really simple and tiny language, one week is enough to learn it in a way that allows you to use it productively.

1.2 Features

FreePOPs is the only software I know with these features:

- POP3 server RFC compliant (not full featured but compliant).
- Portable (written in C and LUA that is written in C, so everything is written in the most portable language around the world).
- Small (in the sense of resources usage) and reasonably fast.
- Extremely extensible on the fly using a simple and powerful language.
- Pretty documented.
- Released under the GNU/GPL license (this means FreePOPs is Free Software).

1.3 Plugins

These are the plugin currently included in FreePOPs:

libero.lua This plugins fully supports the `www.libero.it` webmail for mailboxes with domains like `@libero.it`, `@iol.it`, `@inwind.it`, `@blu.it`.

popforward.lua This is a plugin mainly used for testing some FreePOPs modules. It acts as a POP3 forward, it simply works as a man in the middle of you and a real POP3 server. This allowed us to test FreePOPs without having any real plugin already written. You may consider using it to masquerade a really buggy POP3 server that can be easily compromised trough malicious requests. Obviously we suggest you to examine properly this plugin, and hack on it to prevent malicious requests to your server.

aggregator.lua Many sites provide an RSS backend for indexing their news. This plugin makes this RSS behave as a mailbox in which you can find a mail for each news.

flatnuke.lua This is a more powerful aggregator for the FlatNuke CMS¹-based websites, and allows the download of the whole news body.

2 History

FreePOPs was not born from scratch. A similar project (only in the main usage situation) is LiberoPOPs.

The ancestor of FreePOPs is completely written in C for some uninteresting reasons. LiberoPOPs supports “plugins” but in a more static and complex way. The POP3 server frontend could be attached to a backend written in C, this means you have to recompile and restart LiberoPOPs each time to change a line in a plugin. Another interesting point is that LiberoPOPs was created from scratch in a really short time (you have to be Italian and use a `@libero.it` mail address to understand why), this means it was born with a lot of bugs and FIX-ME in the code.

The LiberoPOPs project had a quick success, because everybody needed it, and this means we had a lot of users. In the opensource (and also Linux) philosophy you have to release frequently and this was exactly what we did: we used to release every two days. We were working not with Unix users, nor hackers, but mostly with Win32 users. Suddenly we realized that they were lazy/bored of updating the software every 2 days. The ugly Win-world has taught them that software auto-updates, auto-install and even auto-codes probably.

¹See [HTTP://flatnuke.sourceforge.net](http://flatnuke.sourceforge.net) for the project homepage

We tried to solve this pulling out of the C engine most of the change-prone code, but this was really hard since C is not thought to do this. After LiberoPOPs had stabilized we started to think how to solve this.

A scripting/interpreted embedded language seemed to me a nice choice and after a long search on the net and on the newsgroup of my university I found LUA.. This is not the place for telling the world how good this small language is and I won't talk more about it here. Integrating the LUA interpreter in LiberoPOPs was not so hard and FreePOPs is the result. Now it is really easier to write/test a plugin and (even if it is not implemented yet) an auto-update facility is really easy to code since there is no need to recompile the C core in most cases.

3 FreePOPs configuration file

FreePOPs doesn't really need a configuration. Most users shouldn't change the configuration file. In case you are a developer or a really curious user the configuration file is `config.lua`, placed in the program directory under `win32` or in `/etc/freepops/` in a posix environment.

Later you will learn that plugins are associated with a mail address domain, and some of these plugins are aliased to other domains to make it easier to fetch some news from some sites. Read the plugin documentation for more info about them, and maybe send as a mail with your new alias if you want it to be integrated in the next FreePOPs release.

4 FreePOPs command line parameters

The real FreePOPs configuration is made trough command line arguments. They are described in depth in the man page in Unix environments, here we present only the most useful:

- p <port>, -port <port>** By default FreePOPs binds on port 2000. To alter this behaviour just use this switch.
- P <host>:<port>, -proxy <host>:<port>** To tell FreePOPs which is your HTTP proxy.
- v, -verbose, -w, -veryverbose** This tells FreePOPs to log some interesting info for bug reporting.
- t <num>, -threads <num>** FreePOPs is able to manage multiple connections, up to num. Default is 5.

In posix environments like Debian GNU/Linux you can start FreePOPs at boot time as a standard service. In this case the command line switches are stored in `/etc/default/freepops`, in some rpm based systems you should find the same file as `/etc/sysconfig/freepops`.

5 Email client configuration

To configure your email client you must change the pop3 server settings. Usually you must use localhost as the pop3 host name, and 2000 as the pop3 port. In case you install FreePOPs in another computer of your LAN, you should use the host's name instead of localhost. In case you changed the default port with the `-p` switch you should know what you are doing. You have to use a full username, for example `something@libero.it` instead of only `something`. This is because FreePOPs chooses the plugin to load looking at your username. Later we will present all the plugins and their associated domains.

6 Plugins

Here we give a detailed description of each plugin.

6.1 `libero.lua`

This plugin allows you to read the mails you have in a `@libero.it`, `@iol.it`, `@inwind.it` and `@blu.it` mailbox. This means you can still use your favorite mail reader instead of using the webmail. This plugin acts as a browser that surfs your webmail account and make it appear as a POP3 server. For more info on this plugin you may read the LiberoPOPs (FreePOPs ancestor) website at <http://liberopops.sourceforge.net>

To use this plugin you have to use your full email address as the username and your real password as the password.

6.2 `popforward.lua`

This plugin was developed to test FreePOPs before any real plugins were written. It simply forwards to a real POP3 server your local requests. This can be used for masquerading a buggy POP3 server, but if you think you need this you should examine carefully the plugin code and add checks to improve the detection/avoidance of malicious requests, since the plugin was not born with security in mind.

To use this plugin you have to edit the `config.lua` file. This because we can't add all the existing POP3 server to this file :) The plugin wants two arguments,

the POP3 host and the port (usually 110) on which the server listens. This is an example of a configuration line for this plugin, in which all email addresses of the @virgilio.it domain are forwarded to in.virgilio.it:110:

```
-- popforward plugin
freepops.MODULES_MAP["virgilio.it"] = {
    name="popforward.lua",
    args={
        port=110,
        host="in.virgilio.it"
    }
}
```

6.3 aggregator.lua

Usually you can benefit from the W3C's RSS format when you read some website news. The RSS file indexes the news, providing a link to them. This plugin can make your mail client see the RSS file as a mailbox from which you can download each news as if it was a mail message. The only limitation is that this plugin can fetch only a news summary plus the news link.

To use this plugin you have to use a casual username with the @aggregator suffix (ex: foo@aggregator) and as the password the URL of the RSS file(ex: <http://www.securityfocus.com/rss/vulnerabilities.xml>). For your commodity we added some alias for you. This means you have not to search by hand the URL of the RSS file. We added some domain, for example @securityfocus.com, that can be used to directly use the aggregator plugin with these website. To use these alias you have to use a username in the form something@aggregatordomain and a casual password. This is the list of aliases for the aggregator plugin.

aggregatordomain	description
freepops.rss.en	http://freepops.sourceforge.net/ news (English)
freepops.rss.it	http://freepops.sourceforge.net/ news (Italian)
flatnuke.sf.net	http://flatnuke.sourceforge.net/ news (Italian)
ziobudda.net	http://ziobudda.net/ news (both Italian and English)
punto-informatico.it	http://punto-informatico.it/ news (Italian)
linuxdevices.com	http://linuxdevices.com/ news (English)
securityfocus.com	http://www.securityfocus.com/ new vulnerabilities (English)

6.4 flatnuke.lua

This plugin is an aggregator plugin specialized in the websites made with the FlatNuke² content management system, or other sites that use the same news

²[HTTP://flatnuke.sourceforge.net](http://flatnuke.sourceforge.net)

format like the FreePOPs website. Since in a FlatNuke site the news are stored in plain xml files this plugin is able to fetch the whole news, and not only the headings as the aggregator plugin. This is really useful if you don't want to surf the website to read the news.

To use this plugin you have to use a username with the @flatnuke domain (ex: something@flatnuke) and a flatnuke homepage URL as the password (ex: <http://flatnuke.sourceforge.net/>, no need for the RSS file URL since FlatNuke puts the RSS in a fixed and well known position. There are some alias for FlatNuke sites, see the aggregator plugin documentation to know what this means):

aggregatordomain	description
freepops.en	http://freepops.sourceforge.net/ full news (English)
freepops.it	http://freepops.sourceforge.net/ full news (Italian)
flatnuke.it	http://flatnuke.sourceforge.net/ full news (Italian)

7 Creating a plugin

The best way of doing this is to read carefully the `libero.lua` file, that is a quite simple but really commented example of web-mail plugin. Then you should copy the `skeleton.lua` file and start hacking on it. Remember to edit `config.lua` to make FreePOPs associate the right mail-addresses domains to your new plugin. A simpler example is `flatnuke.lua` that is a web-news plugin.

7.1 The interface between the C core and a plugin

As we explained before the C POP3 frontend has to be attached to a LUA backend. The interface is really simple if you know the POP3 protocol. Here we only summarize the meaning, but the RFC 1939 (included in the `doc/` directory of the source distribution) is really short and easy to read. As your intuition should suggest the POP3 client may ask the pop3 server to know something about the mail that is in the mailbox and eventually retrieve/delete a message. And this is exactly what it does.

The backend must implement all the POP3 commands (like USER, PASS, RETR, DELE, QUIT, LIST, ...) and must give back to the frontend the result. Let us give a simple example of a POP3 session taken from the RFC:

```
1 S: <wait for connection on TCP port 110>
2 C: <open connection>
3 S:   +OK POP3 server
4 C:   USER linux@kernel.org
5 S:   +OK now insert the password
```

```
6 C: PASS gpl
7 S: +OK linux's maildrop has 2 messages (320 octets)
8 C: STAT
9 S: +OK 1 320
10 C: LIST
11 S: +OK 2 messages (320 octets)
12 S: 1 320
13 S: .
14 C: RETR 1
15 S: +OK 120 octets
16 S: <the POP3 server sends message 1>
17 S: .
18 C: DELE 1
19 S: +OK message 1 deleted
20 C: QUIT
21 S: +OK dewey POP3 server signing off (maildrop empty)
22 C: <close connection>
23 S: <wait for next connection>
```

In this session the backend will be called for lines 4, 6, 8, 10, 14, 18, 20 (all the C: lines) and respectively the functions implementing the POP3 commands will be called this way

```
user(p, "linux@kernel.org")
pass(p, "gpl")
stat(p)
list_all(p)
retr(p, 1)
dele(p, 1)
quit_update(p)
```

Later I will make clear what `p` is. I hope we'll remove it making it implicit for complete transparency. It is easy to understand that there is a 1-1 mapping between POP3 commands and plugin function calls. You can view a plugin as the implementation of the POP3 interface.

7.2 The interface between a plugin and the C core

Let us take in exam the call to `pass(p, "linux@kernel.org")`. Here the plugin should authenticate the user (if there is a sort of authentication) and inform the C core of the result. To achieve this each plugin function must return an error flag, to be more accurate one of these errors:

Code	Meaning
POPSERVER_ERR_OK	No error
POPSERVER_ERR_NETWORK	An error concerning the network
POPSERVER_ERR_AUTH	Authorization failed
POPSERVER_ERR_INTERNAL	Internal error, please report the bug
POPSERVER_ERR_NOMSG	The message number is out of range
POPSERVER_ERR_LOCKED	Mailbox is locked by another session
POPSERVER_ERR_EOF	End of transmission, used in the popserver_callback
POPSERVER_ERR_TOOFAST	You are not allowed to reconnect to the server now, wait a bit and retry
POPSERVER_ERR_UNKNOWN	No idea of what error I've encountered

In our case the most appropriate error codes are `POPSERVER_ERR_AUTH` and `POPSERVER_ERR_OK`. This is a simple case, in which an error code is enough. Now we analyze the more complex case of the call to `list_all(p)`. Here we have to return an error code as before, but we have also to inform the C core of the size of all messages in the mailbox. Here we need the `p` parameter passed to each plugin function (note that parameter may become implicit in the future). `p` stands for the data structure that the C expects us to fill calling appropriate functions like `set_mailmessage_size(p,num,size)` where `num` is the message number and `size` is the size in bytes. Usually it is really common to put more functions all together. For example when you get the message list page in a webmail you know the number of the messages, their size and uidl so you can fill the `p` data structure with all the informations for LIST, STAT, UIDL.

The last case that we examine is `retr(p,num,data)`. Since a mail message can be really big, there is no pretty way of downloading the entire message without making the mail client complain about the server death. The solution is to use a callback. Whenever the plugin has some data to send to the client he should call the `popserver_callback(buffer,data)`. `data` is an opaque structure the popserver needs to accomplish its work (note that this parameter may be removed for simplicity). In some cases, for example if you know the message is small or you are working on a fast network, you can fetch the whole message and send it, but remember that this is more memory consuming.

8 Submitting a bug

When you have problems or you think you found a bug, please follow strictly this *iter*:

1. Update to the most recent version of FreePOPs.
2. Try to reproduce the bug, if the bug is not easily reproducible we are out of luck. Something can still be tried: if the software crashed you could

compile it from the sources, install valgrind, run freepopsd with valgrind and hope the error messages are interesting.

3. Clean the log files
4. Start FreePOPs with the -w switch
5. Reproduce the bug
6. Send to the developers the log, plus any other info like your system type and how to reproduce this bug.

9 Authors

This manual has been written by Enrico Tassi <gareuselesinge@users.sourceforge.net> and revised and translated by Nicola Cocchiario <ncocchiario@users.sourceforge.net>

9.1 Developers

FreePOPs is developed by:

- Enrico Tassi <gareuselesinge@users.sourceforge.net>
- Alessio Caprari <alessiofender@users.sourceforge.net>
- Nicola Cocchiario <ncocchiario@users.sourceforge.net>
- Simone Vellei <simone_vellei@users.sourceforge.net>

LiberoPOPs is developed by:

- Enrico Tassi <gareuselesinge@users.sourceforge.net>
- Alessio Caprari <alessiofender@users.sourceforge.net>
- Nicola Cocchiario <ncocchiario@users.sourceforge.net>
- Simone Vellei <simone_vellei@users.sourceforge.net>
- Giacomo Tenaglia <sonicsmith@users.sourceforge.net>

10 Thanks

Special thanks goes to the users who tested the software, to the hackers who made it possible to have a free and reliable development environment as the Debian GNU/Linux system.