Custom STMPd

Frederico Falcao

v0.1
Out 2019

1 Intro

This project builds a STMP server on ubuntu, to host a private mail server.
The idea is to filter emails and redirect stuff based on sql rules.

The server is run on a public GitHub project : "Guerrilla STMPd" writen in PHP.

Why?

1. Avoid a lot of spam, given simple rules
2. Create PDFs from newsletters, and upload them to Dropbox to be synched with Sony DPT

2 Init Database Code

```php
<?php
$code = file_get_contents(__FILE__);      // 1. Get this file as string
$code = substr($code, __COMPILER_HALT_OFFSET__);  // 2. Filter only the second part of the file
$code_shortcuts = [         // 3. List all the code shortcuts
  "ak"  => "array_keys",
  "im"  => "implode",
  "am"  => "array_map",
  "fn"  => "function",
  "rt"  => "return",
];
              // 4. Replace the shortcuts with actual keywords
$code = str_replace(array_keys($code_shortcuts),$code_shortcuts,$code);
$tmp_file = exec("mktemp");       // 5. Create temporary file
file_put_contents($tmp_file, $code);      // 6. Output: write the produced code to the temp file
include $tmp_file;          // 7. Execute: parse the temporary file
system("rm $tmp_file");         // 8. Clean-up: remove the temporary file
die();              // 9. exit
__halt_compiler();

<?php
$cols = [
  'mail_id' =>'int(11)',
  'date'    =>'datetime',
  'from'    =>'varchar(128)',
  'to'    =>'varchar(128)',
  'subject' =>'varchar(255)',
  'body'    =>'text',
  'charset' =>'varchar(32)',
  'mail'    =>'longblob',
  'spam_score'  =>'float',
  'hash'    =>'char(32)',
  'content_type'  =>'varchar(64)',
  'recipient' =>'varchar(128)',
  'has_attach'  =>'int(11)',
  'ip_addr' =>'varchar(64)'

];
?>
DROP DATABASE IF EXISTS gmail_mail;
CREATE DATABASE gmail_mail;

DROP USER IF EXISTS gmail_mail;
CREATE USER 'gmail_mail' IDENTIFIED BY 'ok';
GRANT ALL ON gmail_mail.* TO gmail_mail;

USE gmail_mail;
CREATE TABLE new_mail(<?=im(", ",am(fn($o,$k){rt "`$o` $k";},ak($cols),$cols));?>);

DELIMITER //
CREATE Procedure Received_Email (<?=im(", ",am(fn($o){rt "IN $o";},ak($cols)))?>)
BEGIN
  INSERT INTO new_mail
  (<?=im(", ",ak($cols))?>
  VALUES
  (<?=im(", ",ak($cols))?>);
END
DELIMITER ;
```

3 SMTPd Code

```
<?php
/**


Guerrilla SMTPd
An minimalist, event-driven I/O, non-blocking SMTP server in PHP

Copyright (c) 2014 Flashmob, GuerrillaMail.com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated
documentation files (the "Software"), to deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the
Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

What is Guerrilla SMTPd?
It's a small SMTP server written in PHP, optimized for receiving email.
Written for GuerrillaMail.com which processes tens of thousands of emails
every hour.

Version: 2.3
Author: Flashmob, GuerrillaMail.com
Contact: flashmob@gmail.com
License: MIT
Repository: https://github.com/flashmob/Guerrilla-SMTPd
Site: http://www.guerrillamail.com/

See README for more details

Version History:

2.3
- Default my_save_email() callback changed to use PDO

2.2
- decoding of email headers improved
- replaced imap_rfc822_parse_adrlist() with mailparse_rfc822_parse_addresses() as it's more portable
- bug in get_email_headers
- posix_seteuid() to user-level after opening port 25
- code cleanup

2.1
- Adjusted header extraction
- Memcached was a bad idea. Perhaps CouchDB?
- Compression before saving (changed MySQL mail fields to BLOB)
```

```
53
54 2.0
55 - First release, re-write of Guerrilla SMTPd 1.2 to use libevent
56 http://www.php.net/manual/en/book.libevent.php
57
58  */
59 $save_email_function = 'my_save_email'; // name of the default callback function
60 if (file_exists(dirname(__FILE__) . '/savemail.php')) {
61     // Define your own function for saving email inside savemail
62     // and assign the name of your function to $save_email_function
63     require(dirname(__FILE__) . '/savemail.php');
64 }
65
66
67 $fp = fopen(dirname(__FILE__) . "/tmp/smtpd2" . basename(__FILE__) . "_lock.txt", "w");
68
69 if (flock($fp, LOCK_EX | LOCK_NB)) { // do an exclusive lock, non-blocking
70     ftruncate($fp, 0);
71     fwrite($fp, "Write something here\n");
72
73 } else {
74     echo "Couldn't get the lock!";
75     fclose($fp);
76     die();
77 }
78
79 // It's a daemon! We should not exit... A warning though:
80 // You may need to have another script to
81 // watch your daemon process and restart of needed.
82 set_time_limit(0);
83 ignore_user_abort(true);
84
85
86 /**
87  * Arguments
88  * -p <port>        listen on port
89  * -l <log_file>    log to log_file
90  * -v               output to console
91  */
92 if (isset($argv) && (sizeof($argv) > 1)) {
93     foreach ($argv as $i => $arg) {
94         if ($arg == '-p') {
95             $listen_port = (int)$argv[$i + 1];
96         }
97         if ($arg == '-l') {
98             $log_file = $argv[$i + 1];
99         }
100        if ($arg == '-v') {
101            $verbose = true;
102        }
103    }
104 }
105
106 if (isset($log_file)) {
107
108     if (!file_exists($log_file) && file_exists(dirname(__FILE__) . '/' . $log_file)) {
109         $log_file = dirname(__FILE__) . '/' . $log_file;
110     } else {
111         $log_file = dirname(__FILE__) . '/log.txt';
112     }
113 } else {
114
115     echo "log file not specified[]\n";
116     $log_file = false;
117 }
```

```php
if (!isset($verbose)) {

    $verbose = false;
}
##################################################################
# Configuration start
##################################################################

if (file_exists(dirname(__FILE__) . '/smtpd-config.php')) {
    // place a copy of the define statements in to smtpd-config.php
    require_once(dirname(__FILE__) . '/smtpd-config.php');
} else {
    // defaults if smtpd-config2.php is not available
    log_line('Loading defaults', 1);
    define('MAX_SMTP_CLIENTS', 1000);
    define('GSMTP_MAX_SIZE', 131072);
    define('GSMTP_HOST_NAME', 'guerrillamail.com'); // This should also be set to reflect your RDNS
    define('GSMTP_LOG_FILE', $log_file);
    define('GSMTP_VERBOSE', $verbose);
    define('GSMTP_TIMEOUT', 100); // how many seconds before timeout.
    define('MYSQL_HOST', 'localhost');
    define('MYSQL_USER', 'gmail_mail');
    define('MYSQL_PASS', 'ok');
    define('MYSQL_DB', 'gmail_mail');
    define('GSMTP_USER', 'nobody');
    define('GM_MAIL_TABLE', 'new_mail'); // MySQL table for storage
    define('GM_PRIMARY_MAIL_HOST', 'guerrillamailblock.com'); // The primary domain name of you email.
    // Allowed hosts, a list of domains accepted by this server. Comma dilimited, do not include spaces
    define('GM_ALLOWED_HOSTS',
    'guerrillamailblock.com,guerrillamail.com,guerrillamail.net,guerrillamail.biz,guerrillamail.org,sharklasers.com');
    define('GSMTP_VERIFY_USERS', false);

}
if (!defined('GSTMP_PORT')) {
    define ('GSTMP_PORT', 25);
}
if (!defined('GSTMP_LISTEN_INTERFACE')) {
    define ('GSTMP_LISTEN_INTERFACE', '0.0.0.0'); // Which ip to bind on? 0.0.0.0 listen on all
}

if (!function_exists('is_host_allowed')) {
    // Implement your own function which returns a hashtable of allowed hosts
    function is_host_allowed($host) {
        static $hosts;
        if (!isset($hosts)) {
            $tmp = explode(',', GM_ALLOWED_HOSTS);
            if (!empty($tmp)) {
                $hosts = array_flip($tmp);
            }
        }
        if (isset($hosts[$host])) {
            return is_numeric($hosts[$host]);
        }
        return false;
    }
}
##################################################################
# Configuration end
##################################################################

```

```
if (!isset($listen_port)) {
    $listen_port = GSTMP_PORT;
}

/**
 * @param \PDO $link
 *
 * @return bool
 */
function ping_db_link(\PDO $link) {
    if (!is_object($link)) {
        return false;
    }
    try {
        $link->query('SELECT 1');
    } catch (\PDOException $e) {
        log_line($e->getMessage(), 1);
        return false;
    }
    return true;
}

/**
 * Returns a PDO connection to MySQL
 * Returns the existing connection, if a connection was opened before.
 * On the consecutive call, It will ping MySQL if not called for
 * the last 60 seconds to ensure that the connection is up.
 * Will attempt to reconnect once if the
 * connection is not up.
 *
 * @param bool $reconnect True if you want the link to re-connect
 *
 * @return bool|\PDO
 */

function get_db_link($reconnect = false)
{
    global $DB_ERROR;
    static $last_ping_time;
    static $link;
    if (isset($last_ping_time)) {
        // more than a minute ago?
        if (($last_ping_time + 30) < time()) {
            if (false === ping_db_link($link)) {
                $reconnect = true; // try to reconnect
            }
            $last_ping_time = time();
        }
    } else {
        $last_ping_time = time();
    }
    if (isset($link) && !$reconnect) {
        return $link;
    }
    $DB_ERROR = '';
    try {
        $link = new \PDO(
            'mysql:host=' . MYSQL_HOST . ';dbname=' . MYSQL_DB . ';charset=utf8',
            MYSQL_USER,
            MYSQL_PASS);
        $link->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
    } catch (\PDOException $e) {
        log_line($e->getMessage(), 1);
        return false;
    }
```

```
243     return $link;
244 }
245 
246 ###############################################################
247 # Guerrilla SMTPd, Main
248 ###############################################################
249 
250 $GM_ERROR = false;
251 // Check MySQL connection
252 
253 if (defined('MYSQL_HOST') && get_db_link() === false) {
254     die('Please check your MySQL settings');
255 }
256 
257 /**
258  * $clients array List of all clients currently connected including session data for each
        client
259  */
260 $clients = array();
261 
262 /**
263  * Setup the main event loop, open a non-blocking stream socket and set the
264  * ev_accept() function to accept new connection events
265  */
266 $errno = '';
267 $errstr = '';
268 $socket = stream_socket_server('tcp://' . GSTMP_LISTEN_INTERFACE . ':' . $listen_port,
        $errno, $errstr);
269 
270 if (!$socket) {
271     echo "Cannot create a new socket for " . GSTMP_LISTEN_INTERFACE . " $listen_port, $errno
        , $errstr\n";
272     die();
273 }
274 stream_set_blocking($socket, 0);
275 $base = event_base_new();
276 $event = event_new();
277 event_set($event, $socket, EV_READ | EV_PERSIST, 'ev_accept', $base);
278 event_base_set($event, $base);
279 event_add($event);
280 log_line("Guerrilla Mail Daemon started on port " . $listen_port, 1);
281 
282 // drop down to user level after opening the smptp port
283 $user = posix_getpwnam(GSMTP_USER);
284 posix_setgid($user['gid']);
285 posix_setuid($user['uid']);
286 $user = null;
287 
288 event_base_loop($base);
289 
290 
291 /**
292  * Handle new connection events. Add new clients to the list. The server will write a
        welcome message to each client
293  * Sets the following functions to handle I/O events
294  * 'ev_read()', 'ev_write()', 'ev_error()'
295  *
296  * @param $socket resource
297  * @param $flag   int A flag indicating the event. Consists of the following flags:
        EV_TIMEOUT, EV_SIGNAL, EV_READ, EV_WRITE and EV_PERSIST.
298  * @param $base   resource created by event_base_new()
299  */
300 function ev_accept($socket, $flag, $base)
301 {
302 
```

```
303     global $clients;
304     static $next_id = 0;
305 
306     $connection = stream_socket_accept($socket);
307     stream_set_blocking($connection, 0);
308 
309     $next_id++;
310 
311     $buffer = event_buffer_new($connection, 'ev_read', 'ev_write', 'ev_error', $next_id);
312     event_buffer_base_set($buffer, $base);
313     event_buffer_timeout_set($buffer, GSMTP_TIMEOUT, GSMTP_TIMEOUT);
314     event_buffer_watermark_set($buffer, EV_READ, 0, 0xffffff);
315     event_buffer_priority_set($buffer, 10);
316     event_buffer_enable($buffer, EV_READ | EV_PERSIST);
317 
318     $clients[$next_id]['socket'] = $connection; // new socket
319     $clients[$next_id]['ev_buffer'] = $buffer; // new socket
320     $clients[$next_id]['state'] = 0;
321     $clients[$next_id]['mail_from'] = '';
322     $clients[$next_id]['helo'] = '';
323     $clients[$next_id]['rcpt_to'] = '';
324     $clients[$next_id]['error_c'] = 0;
325     $clients[$next_id]['read_buffer'] = '';
326     $clients[$next_id]['read_buffer_ready'] = false; // true if the buffer is ready to be
    fetched
327     $clients[$next_id]['response'] = ''; // response messages are placed here, before they
    go on the write buffer
328     $clients[$next_id]['time'] = time();
329 
330     $address = stream_socket_get_name($clients[$next_id]['socket'], true);
331     $clients[$next_id]['address'] = $address;
332 
333     process_smtp($next_id);
334 
335     if (strlen($clients[$next_id]['response']) > 0) {
336         event_buffer_write($buffer, $clients[$next_id]['response']);
337         add_response($next_id, null);
338     }
339 }
340 
341 /**
342  * Handle error events, including timeouts
343  *
344  * @param $buffer resource Event buffer
345  * @param $error  int flag (EV_TIMEOUT, EV_SIGNAL, EV_READ, EV_WRITE and EV_PERSIST)
346  * @param $id     int client id
347  */
348 function ev_error($buffer, $error, $id)
349 {
350     global $clients;
351     log_line("event error $error", 1);
352     event_buffer_disable($clients[$id]['ev_buffer'], EV_READ | EV_WRITE);
353     event_buffer_free($clients[$id]['ev_buffer']);
354     fclose($clients[$id]['socket']);
355     unset($clients[$id]);
356 }
357 
358 function ev_write($buffer, $id)
359 {
360     global $clients;
361     // close if the client is on the kill list
362 
363     if (!empty($clients[$id]['kill_time'])) {
364         event_buffer_disable($clients[$id]['ev_buffer'], EV_READ | EV_WRITE);
365         event_buffer_free($clients[$id]['ev_buffer']);
```

```
366         fclose($clients[$id]['socket']);
367         unset($clients[$id]);
368     }
369 
370 }
371 
372 function ev_read($buffer, $id)
373 {
374     global $clients;
375     global $redis;
376     while ($read = event_buffer_read($buffer, 1024)) {
377         $clients[$id]['read_buffer'] .= $read;
378     }
379 
380 
381     // Determine if the buffer is ready
382     // The are two states when we determine if the buffer is ready.
383     // State 1 is the command state, when we wait for a command from
384     // the client
385     // State 2 is the DATA state, when the client sends the data
386     // for the email.
387 
388     if ($clients[$id]['state'] === 1) {
389         // command state, strings terminate with \r\n
390         if ((strlen($clients[$id]['read_buffer']) > 1)
391             && strpos($clients[$id]['read_buffer'], "\r\n", strlen($clients[$id]['
    read_buffer']) - 2) !== false
392         ) {
393             $clients[$id]['read_buffer_ready'] = true;
394         }
395 
396     } elseif ($clients[$id]['state'] === 2) {
397         // DATA reading state
398         // not ready unless you get a \r\n.\r\n at the end
399         $len = strlen($clients[$id]['read_buffer']);
400         if (($len > GSMTP_MAX_SIZE)
401             || (($len > 4)
402                 && (strpos(
403                     $clients[$id]['read_buffer'],
404                     "\r\n.\r\n",
405                     $len - 5
406                 )) !== false)
407         ) {
408             $clients[$id]['read_buffer_ready'] = true; // finished
409             $clients[$id]['read_buffer'] = substr(
410                 $clients[$id]['read_buffer'],
411                 0,
412                 $len - 5
413             );
414         }
415     } elseif ($clients[$id]['state'] === 3) {
416         $clients[$id]['read_buffer_ready'] = true; // finished
417     }
418 
419 
420     if (!$redis) {
421         //error_log("no redis ev read: ".var_dump($redis));
422     }
423 
424     process_smtp($id);
425 
426     if (strlen($clients[$id]['response']) > 0) {
427 
428         event_buffer_write($buffer, $clients[$id]['response']);
429         add_response($id, null);
```

```
    }


}


////////////////////////////////////////////////

/**
 * SMTP server state machine. Use read_line() to get input from the buffer, add_response() to queue things
 * to the output buffer, kill_client() to stop talking to the client. $save_email_function() to store the email.
 *
 * @param $client_id int
 */
function process_smtp($client_id)
{

    global $clients;
    global $GM_ERROR;
    global $save_email_function;

    switch ($clients[$client_id]['state']) {
        case 0:
            add_response(
                $client_id,
                '220 ' . GSMTP_HOST_NAME .
                ' SMTP Guerrilla-SMTPd #' . $client_id . ' (' . sizeof($clients) . ') ' . gmdate('r')
            );
            $clients[$client_id]['state'] = 1;

            break;
        case 1:

            $input = read_line($clients, $client_id);
            if ($input) {
                log_line('[' . $client_id . '] cmd:' . $input);
            }
            if ($input) {

                if (stripos($input, 'HELO') === 0) {
                    $temp = explode(' ', $input);
                    $clients[$client_id]['helo'] = trim($temp[1]);
                    add_response(
                        $client_id,
                        '250 ' . GSMTP_HOST_NAME . ' Hello ' . trim($temp[1]) .
                        ' [' . $clients[$client_id]['address'] . '], got some spam for me?'
                    );
                } elseif (stripos($input, 'EHLO') === 0) {
                    $temp = explode(' ', $input);

                    $clients[$client_id]['helo'] = trim($temp[1]);
                    add_response(
                        $client_id,
                        '250-' . GSMTP_HOST_NAME . ' Hello ' . trim($temp[1]) .
                        '[' . $clients[$client_id]['address'] . ']' . "\r\n" .
                        "250-SIZE " . GSMTP_MAX_SIZE . "\r\n" . //"250-PIPELINING\r\n" .
                        //"250-AUTH PLAIN LOGIN\r\n" .
                        "250 HELP"
                    );
                } elseif (stripos($input, 'XCLIENT') === 0) {
                    $clients[$client_id]['address'] = substr($input, 13);
                    if ($pos = strpos($clients[$client_id]['address'], ' ')) {
```

```
                        $clients[$client_id]['address'] = substr($clients[$client_id]['
    address'], 0, $pos);
                    }
                    add_response($client_id, '250 Ok');
                } elseif (stripos($input, 'MAIL FROM:') === 0) {
                    $clients[$client_id]['mail_from'] = substr($input, 10);
                    add_response($client_id, '250 Ok');
                } elseif ((stripos($input, 'RCPT TO:') === 0)) {

                    $email = extract_rcpt_email(substr($input, 8));
                    // do not allow CC, RCPT TO is allowed only once
                    if (empty($clients[$client_id]['rcpt_to']) && ($email)) {
                        $clients[$client_id]['rcpt_to'] = $email;
                        add_response($client_id, '250 Accepted');
                    } else {
                        log_line(
                            'mailbox unavailable[' . $input . '] input:' .
                            $input,
                            1
                        );
                        // do not let CC.
                        kill_client(
                            $client_id,
                            '550 Requested action not taken: mailbox unavailable'
                        );
                    }
                } elseif (stripos($input, 'DATA') === 0) {
                    add_response(
                        $client_id,
                        '354 Enter message, ending with "." on a line by itself'
                    );
                    $clients[$client_id]['state'] = 2;

                    $clients[$client_id]['read_buffer'] = '';
                } elseif (stripos($input, 'QUIT') === 0) {

                    log_line("client asked to quit", 1);
                    kill_client($client_id, '221 Bye');
                    continue;

                } elseif (stripos($input, 'NOOP') === 0) {

                    log_line("client NOOP from client", 1);

                    add_response($client_id, '250 OK');

                } elseif (stripos($input, 'RSET') === 0) {

                    $clients[$client_id]['read_buffer'] = '';
                    $clients[$client_id]['rcpt_to'] = '';
                    $clients[$client_id]['mail_from'] = '';
                    add_response($client_id, '250 OK');

                } else {
                    log_line('[' . $client_id . ']unrecoginized cmd:' . $input, 1);
                    add_response($client_id, '500 unrecognized command');
                    $clients[$client_id]['error_c']++;
                    if (($clients[$client_id]['error_c'] > 3)) {
                        kill_client($client_id, '500 Too many unrecognized commands');
                        continue;

                    }
                }
            }
            break;
```

```
        case 2:

            $input = read_line($clients, $client_id);

            if ($input) {


                list($id, $to) = $save_email_function(
                    $input,
                    $clients[$client_id]['rcpt_to'],
                    $clients[$client_id]['helo'],
                    $clients[$client_id]['address'],
                    $clients[$client_id]['mail_from']
                );

                if ($id) {
                    add_response($client_id, '250 OK : queued as ' . $id);
                    // put client back to state 1
                    $clients[$client_id]['state'] = 1;
                    $clients[$client_id]['read_buffer'] = '';
                    $clients[$client_id]['error_c'] = 0;
                } else {
                    // The email didn't save properly, usualy because it was in
                    // an incorrect mime format or bad recipient

                    kill_client(
                        $client_id,
                        "554 Transaction failed (" . strlen($input) . ") " .
                        $clients[$client_id]['rcpt_to'] . " !$id! \{$GM_ERROR\} "
                    );
                    log_line(
                        "Message for client: [$client_id] failed to [$to] {"
                        . $clients[$client_id]['rcpt_to'] . "}, told client to exit.",
                        1
                    );
                }
                continue;
            }
            break;

    }
}


/**
 *
 * Log a line of text. If -v argument was passed, level 1 messages
 * will be echoed to the console. Level 2 messages are always logged.
 *
 * @param string  $l
 * @param integer $log_level
 *
 * @return bool
 */
function log_line($l, $log_level = 2)
{
    $l = trim($l);
    if (!strlen($l)) {
        return false;
    }
    if (($log_level == 1) && (GSMTP_VERBOSE)) {
        echo $l . "\n";
    }
    if (GSMTP_LOG_FILE) {
        $fp = fopen(GSMTP_LOG_FILE, 'a');
```

```
621         fwrite($fp, $l . "\n", strlen($l) + 1);
622         fclose($fp);
623     }
624     return true;
625 }
626 
627 /**
628  * Queue a response back to the client. This will be sent as soon as we get an event
629  *
630  * @param             $client_id
631  * @param null|string $str response to send. \r\n will be added automatically. Use null to
        clear
632  */
633 function add_response($client_id, $str = null)
634 {
635     global $clients;
636     if (strlen($str) > 0) {
637         if (substr($str, -2) !== "\r\n") {
638             $str .= "\r\n";
639         }
640         $clients[$client_id]['response'] .= $str;
641     } elseif ($str === null) {
642         // clear
643         $clients[$client_id]['response'] = null;
644     }
645 
646 }
647 
648 /**
649  * @param             $client_id
650  * @param null|string $msg message to the client. Do not kill untill all is sent
651  *
652  * @internal param $clients
653  */
654 function kill_client($client_id, $msg = null)
655 {
656     global $clients;
657     if (isset($clients[$client_id])) {
658 
659         $clients[$client_id]['kill_time'] = time();
660         if (strlen($msg) > 0) {
661             add_response($client_id, $msg);
662         }
663     }
664 }
665 
666 /**
667  * Returns a data from the buffer only if the buffer is ready. Clears the
668  * buffer before returning, and sets the 'read_buffer_ready' to false
669  *
670  * @param array $clients
671  * @param int   $client_id
672  *
673  * @return string, or false if no data was present in the buffer
674  */
675 function read_line(&$clients, $client_id)
676 {
677 
678 
679     if ($clients[$client_id]['read_buffer_ready']) {
680         // clear the buffer and return the data
681         $buf = $clients[$client_id]['read_buffer'];
682         $clients[$client_id]['read_buffer'] = '';
683         $clients[$client_id]['read_buffer_ready'] = false;
684         return $buf;
```

```
685     }
686     return false;
687 
688 }
689 
690 ##############################################################################################
691 # Mail Parsing and storage to MySQL
692 /**
693  * Use php's ability to set an error handler, since iconv may sometimes give
694  * warnings. this allows us to trap these warnings
695  */
696 function iconv_error_handler($errno, $errstr, $errfile, $errline)
697 {
698     global $iconv_error;
699     $iconv_error = true;
700 
701 }
702 
703 /**
704  * mail_body_decode()
705  * Decode the mail body to binary. Then convert to UTF-8 if not already
706  *
707  * @param string $str           string to decode
708  * @param string $encoding_type eg. 'quoted-printable' or 'base64'
709  * @param string $charset       and of the charsets supported by iconv()
710  *
711  * @return string decoded message in a string of UTF-8
712  */
713 function mail_body_decode($str, $encoding_type, $charset = 'UTF-8')
714 {
715 
716     global $iconv_error;
717     $iconv_error = false;
718 
719     if ($encoding_type == 'base64') {
720         $str = base64_decode($str);
721     } elseif ($encoding_type == 'quoted-printable') {
722         $str = quoted_printable_decode($str);
723     }
724     $charset = strtolower($charset);
725     $charset = preg_replace("/[-:.\/\\\]/", '-', strtolower($charset));
726     // Fix charset
727     // borrowed from http://squirrelmail.svn.sourceforge.net/viewvc/squirrelmail/trunk/
    squirrelmail/include/languages.php?revision=13765&view=markup
728     // OE ks_c_5601_1987 > cp949
729     $charset = str_replace('ks_c_5601_1987', 'cp949', $charset);
730     // Moz x-euc-tw > euc-tw
731     $charset = str_replace('x_euc', 'euc', $charset);
732     // Moz x-windows-949 > cp949
733     $charset = str_replace('x_windows_', 'cp', $charset);
734     // windows-125x and cp125x charsets
735     $charset = str_replace('windows_', 'cp', $charset);
736     // ibm > cp
737     $charset = str_replace('ibm', 'cp', $charset);
738     // iso-8859-8-i -> iso-8859-8
739     // use same cycle until they'll find differences
740     $charset = str_replace('iso_8859_8_i', 'iso_8859_8', $charset);
741 
742     if (strtoupper($charset) != 'UTF-8') {
743         set_error_handler("iconv_error_handler");
744         $str = @iconv(strtoupper($charset), 'UTF-8', $str);
745         if ($iconv_error) {
746             $iconv_error = false;
747             // there was iconv error
748             // attempt mbstring concersion
```

```
            $str = mb_convert_encoding($str, 'UTF-8', $charset);
        }
        restore_error_handler();
    }

    return $str;

}

/**
 * extract_email()
 * Extract an email address from a header string
 *
 * @param string $str
 *
 * @return string email address, false if none found
 */
function extract_rcpt_email($str)
{
    $addr = extract_email($str);
    if ($addr) {
        $pos = strpos($addr, '@');
        if ($pos) {
            $hostname = strtolower(substr($addr, $pos + 1));
            if (is_host_allowed($hostname)) {
                return $addr;
            }
        }
    }
    return false;
}

/**
 * extract_from_email()
 * See extract_email
 * @param string $str
 * @return string
 */
function extract_email($str)
{
    $arr = mailparse_rfc822_parse_addresses ($str);
    if (is_array($arr) && !empty($arr)) {
        foreach ($arr as $item) {
            return strtolower($item['address']);
        }
    }
    // if that didn't work:
    preg_match('/[\\w\\.\\-+=*_]*@[\\w\\.\\-+=*_]*/', $str , $m);
    if (!empty($m[1])) {
        return $m[1];
    }
    return false;
}


/**
 * Save mail callback
 * Accepts an email received from a client during the DATA command.
 * This email is processed, the recipient host is verified, the body is
 * decoded, then saved to the database.
 * This is an example save mail function.
 * Of course, yours can be better! Eg. add buffered writes, place email on message queue,
    encryption, etc
 *
 * @param string $email
```

```
 * @param        $rcpt_to
 * @param        $helo
 * @param        $helo_ip
 *
 * @return array, with the following elements array($hash, $recipient)
 * where the $hash is a unique id for this email.
 */
function my_save_email($email, $rcpt_to, $helo, $helo_ip)
{

    global $GM_ERROR;
    $mimemail = null;
    $hash = '';
    $email .= "\r\n";
    list($to, $from, $subject) = get_email_headers($email, array('To', 'From', 'Subject'));
    $rcpt_to = extract_rcpt_email($rcpt_to);
    $from = extract_email($from);
    list($mail_user, $mail_host) = explode('@', $rcpt_to);
    if (is_host_allowed($mail_host)) {
        $db = get_db_link();
        if ($db === false) {
            // could not get a db connection
            $GM_ERROR = 'could not get a db connection';
            return array(false, false);
        }
        $to = $mail_user . '@' . GM_PRIMARY_MAIL_HOST; // change it to the primary host
        if (GSMTP_VERIFY_USERS) {
            // Here we can verify that the recipient is actually in the database.
            // Note that guerrillamail.com does not do this - you can send email
            // to a non-existing email address, and set to this email later.
            // just an example:
            if (array_pop(explode('@', $to)) !== 'sharklasers.com') {
                // check the user against our user database
                $user = array_shift(explode('@', $to));
                $sql = "SELECT * FROM `gm2_address` WHERE `address_email`=" .
                    $db->quote($user.'@guerrillamailblock.com') . " ";
                $stmt = $db->query($sql);
                if ($stmt->rowCount() == 0) {
                    $GM_ERROR = 'could not verify user';
                    return false; // no such address
                }
            }
        }
        $hash = md5($to . $from . $subject . microtime());
        // add 'received' headers
        $add_head = '';
        $add_head .= "Delivered-To: " . $to . "\r\n";
        $add_head .= "Received: from " . $helo . " (" . $helo . "  [" . $helo_ip . "])\r\n";
        $add_head .= "  by " . GSMTP_HOST_NAME . " with SMTP id " . $hash . "@" .
            GSMTP_HOST_NAME . ";\r\n";
        $add_head .= "  " . gmdate('r') . "\r\n";
        $email = $add_head . $email;
        $body = 'gzencode';
        $charset = '';
        $has_attach = '';
        $content_type = '';

        $email = gzcompress($email, 6);
        $redis = getRedis();
        if (is_object($redis)) {
            // send it of to Redis
            try {
                if ($redis->setex($hash, 3600, $email) === true) {
                    $body = 'redis';
                    $email = '';
```

```
                    //log_line("saved to redis $hash $to");
                } else {
                    log_line("save failed");
                }
            } catch (\RedisException $e) {
                log_line("redis exeption" . var_dump($e, true));
            }
        } else {
            log_line("no redis for you\n");
        }

        $sql = "CALL Received_Email(
                '" . gmdate('Y-m-d H:i:s') . "',
                " . $db->quote($to) . ",
                " . $db->quote($from) . ",
                " . $db->quote($subject) . ",
                " . $db->quote($body) . ",
                " . $db->quote($charset) . ",
                " . $db->quote($email) . ",
                0 " .",
                " . $db->quote($hash) . ",
                " . $db->quote($content_type) .",
                " . $db->quote($to) . ",
                " . $db->quote($has_attach) . ",
                " . $db->quote($helo_ip) . "
            ) ";
        try {
            $db->query($sql);
            $id = $db->lastInsertId();
            if ($id) {
                $sql
                    = "UPDATE
                            gm2_setting
                       SET
                            `setting_value` = `setting_value`+1
                       WHERE
                            `setting_name`='received_emails'
                            LIMIT 1";
                $db->query($sql);
            }
        } catch (\PDOException $e) {
            $GM_ERROR = 'save error ' . $e->getMessage();
            log_line('Failed to save email From:' . $from . ' To:' . $to . ' ' . $e->
    getMessage() . ' ' . $sql, 1);
        }
    } else {
        $GM_ERROR = " -$mail_host- not in allowed hosts:" . $mail_host . " ";
    }
    return array($hash, $to);
}

function get_email_headers($email, $header_names = array())
{
    $ret = array();
    $pos = strpos($email, "\r\n\r\n");
    if (!$pos) {
        // incorrectly formatted email with missing \r?
        $pos = strpos($email, "\n\n");
    }
    $headers_str = substr($email, 0, $pos);
    $headers = iconv_mime_decode_headers($headers_str, ICONV_MIME_DECODE_CONTINUE_ON_ERROR,
    'utf-8');
    if (!$headers) {

    }
```

```
    $headers = array_combine(array_map("ucfirst", array_keys($headers)), array_values(
    $headers));
    foreach ($header_names as $i => $name) {
        if (is_array($headers[$name])) {
            $headers[$name] = implode(', ', $headers[$name]);
        }
        if ((strpos($headers[$name], '=?') === 0)) {
            // workaround if iconv_mime_decode_headers() - sometimes more than one to decode
            if (preg_match_all('#=\?(.+?)\?([QB])\?(.+?)\?=#i', $headers[$name], $matches))
    {
                $decoded_str = '';
                foreach ($matches[1] as $index => $encoding) {
                    if (strtolower($matches[2][$index]) === 'b') {
                        $decoded_str = mail_body_decode($matches[3][$index], 'base64',
    $encoding);
                    } elseif (strtolower($matches[2][$index]) === 'q') {
                        $decoded_str = mail_body_decode($matches[3][$index], 'quoted-
    printable', $encoding);
                    }
                    if (!empty($decoded_str)) {
                        $headers[$name] = str_replace($matches[0][$index], $decoded_str,
    $headers[$name]);
                    }
                }
            }
        }
        $ret[$i] = $headers[$name];
    }
    return $ret;
}

function getRedis()
{
    static $redis;
    if (!class_exists('Redis')) {
        $redis = false;
        error_log('no such class as redis!');
        return false;
    }
    try {
        if (isset($redis)) {
            return $redis;
        }
        $redis = new \Redis();
        $redis->connect('localhost', 6379);
        $redis->setOption(\Redis::OPT_SERIALIZER, \Redis::SERIALIZER_NONE); // don't
    serialize data
    } catch (\RedisException $e) {
        error_log('redis exeption');
        return false;
    }
    return $redis;
}
```