

```

<?php
/**
 * Allows for easy accessing to the PhishTank API
 *
 * PhishTank Runner allows programmers to use the PhishTank API in
 * only two lines; thus saving programmers the hassle of connecting,
 * parsing, and signature generating. Fully PhpDocumentor supported,
 * programmers of any experience can jump right-in and start
 * spreading the PhishTank crave!
 *
 * How To Use:
 * 1. Fill in $shared_secret and $app_key with the strings provided by PhishTank.
 * 2. Run first_run();
 * 3. Run first_run('FROB_ID'); replacing FROB_ID with the Frob ID displayed by first_run();
 * 4. Fill in $username and $api_key with the displayed string provided by
first_run('FROB_ID');
 * 5. Start developing with PhishTank Runner
 *
 * @name      PhishTank Runner
 * @author    David Branco <David@NeoeLiteUSA.com>
 * @link      http://www.NeoeLiteUSA.com/
 * @version   0.9.5
 * @since     Oct 31, 2006
 * @license   http://creativecommons.org/licenses/by/2.5/
 */
class Phishtank {

    /**
     * Your API key that has been associated with the PhishTank account and Applcation Key
     *
     * @access private
     * @var string
     */
    var $shared_secret = 'PhishTank_Shared_Secret';

    /**
     * Your APP (Application) key that has been associated with the PhishTank account and
Shared Secret
     *
     * @access private
     * @var string
     */
    var $app_key = 'PhishTank_APP_Key';

    /**
     * Username that has been associated with the PhishTank account
     *
     * @access private
     * @var string
     */
    var $username = 'PhishTank_Username';

    /**
     * Your API key that has been associated with the PhishTank account
     *
     * @access private
     * @var string
     */
    var $api_key = 'PhishTank_API_Key';

    /**
     * @access private
     * @var string
     */
    var $serialized_response;
}

```

```
/**
 * @access public
 * @var string
 */
var $response_meta;

/**
 * @access public
 * @var string
 */
var $response;

/**
 * @access private
 * @var boolean
 */
var $session_active;

/**
 * @access private
 * @var string
 */
var $token;

/**
 * @access private
 * @var string
 */
var $frob;

/**
 * @access private
 * @var string
 */
var $frob_status;

/**
 * @access private
 * @var string
 */
var $authorization_url;

/**
 * @access public
 * @var string
 */
var $error;

/**
 * @access public
 * @var string
 */
var $last_request;

/**
 * @access private
 * @var string
 */
var $session_not_active = 'Session not active, please start a new session and try again.';

/**
 * Sets the PhishTank Runner environment automaticly upon class generation
 * @see start_session()
 */
function Phishtank($app_key = '', $shared_secret = '', $username = '', $api_key = null) {
    if(!empty($username))
```

```

        $this->username = $username;
    if(!empty($password))
        $this->password = $password;
    if(!empty($shared_secret))
        $this->shared_secret = $shared_secret;
    if(!empty($app_key))
        $this->app_key = $app_key;
    if(isset($api_key))
        $this->api_key = $api_key;
    if(isset($display_api_key))
        $this->display_api_key = $display_api_key;

    error_reporting(E_USER_ERROR | E_USER_WARNING | E_USER_NOTICE);
    $error_handler = set_error_handler(array(&$this, 'error_handler'));
}

/**
 * Starts the lengthy connection to the PhishTank API server.
 *
 * @return boolean true/false
 */
function start_session() {
    if(!$this->session_active) {
        if(!empty($this->api_key) && $this->api_key != 'PhishTank_API_Key') {
            if(empty($this->error))
                $this->auth_token_request();
            if(empty($this->error))
                $this->auth_token_status();
            ($this->return_error()) ? $this->session_active = 1 :
$this->session_active = 0;
            return $this->return_error();
        }else{
            $this->error = 'Due to the complexity of PhishTank, If you are
running PhishTank Runner for the first time please run first_run(); and make sure your APP-Key and
Shared Secret have been set before you run start_session(); Thank you.';
            trigger_error($this->error, E_USER_ERROR);
        }
    }
    return true;
}

/**
 * Run this class if you are using PhishTank Runner for the first time.
 *
 * Once you run this class and have your frob ID, please rerun this class with the frob
variable set.
 * Ex: $tanker->first_run('My_Frob_ID');
 *
 * Note: I understand that using this method makes installation of PhishTank Runner a bit
longer; however,
 * It was the only other reasonable alternative.
 *
 * @return boolean true/false
 */
function first_run($frob = '') {
    if(empty($frob)) {
        $this->auth_frob_request();
        $display = '<b>PhishTank Runner</b> can not connect to the PhishTank API
server without a human manual allowing PhishTanker Runner to use the developers account.'"<br
/>\n";

        $display .= 'Fortunately, PhishTank has made this action simple. Please
click the link provided below and accept PhishTank Runner to use your account.'"<br />\n";
        $display .= 'Once this action has been done, please run first_run again
with the frob variable set to \''.$this->frob.'"<br />\n";
        $display .= 'Ex: $tanker->first_run(\'\'.'.$this->frob.'\');.'"<br />\n";
        $display .= 'The link to allow PhishTank Runner to use your account: <a

```

```

href="'. $this->authorization_url.'">'. $this->authorization_url.'</a>'. "<br />\n";
    $display .= 'We are truly sorry about this inconvenience; however,
security must be upheld; we hope you understand. Hang-in there, your almost ready!';
    die($display);
}
$this->frob = $frob;
if(empty($this->error))
    $this->auth_frob_status();
$display = '<b>PhishTank Runner API key associated with the '.$this->username."
account:</b><br />\n<br />\n'. $this->api_key.'<br />\n<br />\n";
$display .= 'Please place this key in the correct variable ($api_key) and set the
username variable to '.$this->username.', Thank you.'.<br />\n";
$display .= 'I hope this wasn\'t to much trouble for you, look on the bright-side
it\'s over, just remove the first_run(); function call and your done. ';
    die($display);
}

/**
 * Ends the PhishTank API session. Note: This is required at the end of every session at
the
 * risk of having your PhishTank API rights themselves revoked.
 *
 * @return boolean true/false
 */
function end_session() {
    if($this->session_active) {
        $this->auth_token_revoke();
        ($this->return_error()) ? $this->session_active = 0 :
$this->session_active = 1;
        return $this->return_error();
    }
    return true;
}

/**
 * The misc.ping action provides a method for verifying connectivity to the API server.
 *
 * @return boolean true/false
 */
function misc_ping() {
    if($this->session_active) {
        $this->send('&action=misc.ping');
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * The auth.frob.request action allows you to request a "frob," which is used in
authenticating
 * application access to PhishTank user accounts. Your application should store the
returned frob
 * and direct the user to the supplied authorization_url.
 * @param $callback_url string An optional URL to return a user to after allowing
application access. Useful
 * for web-based applications that are requesting account access.
(Note: Remember to URL-encode
 * the callback_url option if passing it as part of a GET request.)
 * @return boolean true/false
 */
function auth_frob_request($callback_url = '') {
    $this->send('&action=auth.frob.request&callback_url='.$callback_url);
    $this->frob = $this->response['frob'];
    $this->authorization_url = $this->response['authorization_url'];
    return $this->return_error();
}

```

```

    }

    /**
     * This action will allow you to check on the status of a pending frob request. A frob
     will have
     * a status of pending until it is accepted by the user, or expires after 14 days. If
     the frob
     * was accepted, this function will give you the credentials needed to request tokens on
     the user's
     * behalf. (Note: The access credentials provided by this function will only be returned
     once per
     * frob, further requests for the frob will return an error. Ensure that you store this
     data within
     * your application to avoid having to re-request authorization from the user.)
     *
     * @return boolean true/false
     */
    function auth_frob_status() {
        $this->send('&action=auth.frob.status&frob='.$this->frob);
        $this->frob_status = $this->response['status'];
        $this->username = $this->response['username'];
        $this->api_key = $this->response['apikey'];
        return $this->return_error();
    }

    /**
     * The auth.token.request action should be called at the beginning of an API session to
     request a
     * token, which is required to authenticate further API commands.
     *
     * @return boolean true/false
     */
    function auth_token_request() {
        $this->send('&action=auth.token.request&api_key='.$this->api_key.'&username='.$this
->username);
        $this->token = $this->response['token'];
        return $this->return_error();
    }

    /**
     * Although any API request may result in an expired token response, developers may use
     the
     * auth.token.status action to check the validity of currently held token, without
     performing any
     * action against the associated account.
     *
     * @return boolean true/false
     */
    function auth_token_status() {
        $this->send('&action=auth.token.status&api_key='.$this->api_key.'&username='.$this-
>username.'&token='.$this->token);
        return $this->return_error();
    }

    /**
     * Calling the auth.token.revoke action is required before ending an API session.
     Applications that
     * do not properly revoke tokens may have their application keys banned to protect the
     security of
     * end users.
     *
     * @return boolean true/false
     */
    function auth_token_revoke() {
        if($this->session_active) {
            $this->send('&action=auth.token.revoke&api_key='.$this->api_key.'&username=

```

```

'. $this->username. '&token=' . $this->token);
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * The check.url action may be used to check a single URL against the PhishTank database.
(Note:
 * Care should be taken when implementing this function to avoid taking action based
solely on a
 * positive in_database response, as innocent URLs may be in the PhishTank database but
not verified
 * as being involved in a phishing attack.)
 *
 * @param    $url    string The URL you wish to check.
 * @return   boolean true/false
 */
function check_url($url) {
    if($this->session_active) {
        $this->send('&action=check.url&token=' . $this->token. '&url=' . $url);
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * The check.email action allows an email to be scanned for known phishing URLs. All URLs
found in
 * email will be returned along with their status. URLs not found by the automated parser
(such as
 * URLs encoded in MIME parts) should not be trusted simply because they were not
returned. A maximum
 * of 25 URLs will be scanned, however the remainder of a message, after the last returned
URL, may
 * be resubmitted for further checking. (Note: Care should be taken when implementing this
function
 * to avoid taking action based solely on a positive in_database response, as innocent
URLs may be in
 * the PhishTank database but not verified as being involved in a phishing attack.)
 *
 * @param    $email string The contents of the email you wish to check.
 * @return   boolean true/false
 */
function check_email($email) {
    if($this->session_active) {
        $this->send('&action=check.email&token=' . $this->token. '&email=' . $email);
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * The submit.url action allows you to submit an individual URL as a suspected phish to
PhishTank.
 * To avoid unnecessary submissions, we encourage application developers to use check.url
before submit.url
 * is called. The submit.url action should not be used for the same URL more than once. If
the submission
 * is accepted (response = true), then the phish ID and the phish detail URL will be
returned. A submission
 * will not be accepted (response = false) if the URL is invalid (by format) or the
submission already

```

```

    * exists in the PhishTank database, among other possibilities.
    *
    * @param      $url      string The url you wish to submit as a suspected phish.
    * @return     boolean true/false
    */
function submit_url($url) {
    if($this->session_active) {
        $this->send('&action=submit.url&token='.$this->token.'&username='.$this->us
ername.'&url='.$url);
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * The submit.email action allows you to submit an individual email as a suspected phish
 to PhishTank.
 * If the submission is accepted (response = true), then the phish ID and the phish detail
 URL will be
 * returned. A submission will not be accepted (response = false) if the submission
 already exists in
 * the PhishTank database, among other possibilities. Please use POST when submitting the
 email. Submit
 * as much of the email as you possibly can, including headers and attachments. The submit
 will not fail
 * if it's missing headers, for instance, but all information is useful when analyzing the
 submission.
 *
 * @param      $email     string The email you wish to submit as a suspected phish.
 * @return     boolean true/false
    */
function submit_email($email) {
    if($this->session_active) {
        $this->send('&action=submit.email&token='.$this->token.'&username='.$this->
username.'&url='.$url);
        return $this->return_error();
    }
    $this->error = $this->session_not_active;
    return false;
}

/**
 * Gerenerates the required signature for each of the requests to the Phishtank API server.
 *
 * @param      $request_string string The partial request string that is to be sent to
 the PhishTank API server.
 * @return     boolean Returns the request string along with the generated signature
 hash.
    */
function generate_sig($request_string) {
    $sig_string = null;
    $request_string =
'version=1&responseformat=php&app_key='.$this->app_key.$request_string;

    list($parameter_string) = array_reverse(explode('?', $request_string));
    $parameters = explode('&', $parameter_string);
    sort($parameters);

    foreach($parameters as $parameter) {
        list($key, $value) = explode('=', $parameter);
        $sig_string .= $key . $value;
    };

    $sig_string = $this->shared_secret . $sig_string;

```

```

        $hash = md5($sig_string);
        $request_string .= '&sig=' . $hash;
        return $request_string;
    }

    /**
     * Sends the API request to the server and recored the results in $serialized_response.
     *
     * @param      $request_string string The complete request sting that is to be issued to
the PhishTank API
     *
     * server complete with the hashed signature.
     */
    function send($request_string) {
        $request_string = $this->generate_sig($request_string);
        $api_request = 'https://api.phishtank.com'.$request_string;
        $this->last_request = 'https://api.phishtank.com/?'.$request_string;

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, 'https://api.phishtank.com/api/');
        curl_setopt($ch, CURLOPT_BINARYTRANSFER, 1);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $request_string);
        $this->serialized_response = curl_exec($ch);
        curl_close($ch);

        $this->parse();
    }

    /**
     * Parses the results from the PhishTank API server and seperates the real results from
the meta.
     */
    function parse() {
        $deserialized = unserialize($this->serialized_response);
        $this->response_meta = $deserialized['response']['meta'];
        $this->response = $deserialized['response']['results'];
        $this->error = $this->response['errortext'];
    }

    /**
     * Checks the error string and returns true or false. This is used to shorten the overall
error
     * handling.
     *
     * @return boolean If true, there is no error; If false, an error has occured.
     */
    function return_error() {
        if(!empty($this->error))
            return false;
        else
            return true;
    }

    /**
     * PhishTank Runner's Personal Error Handler. Used mostly because of the fact I needed to
throw
     * an error if the session did not start automaticly. Had issues with $errno equaling 8
and throwing
     * some unneeded problems.
     *
     * @param      $errno int Error number that was thrown.
     * @param      $errstr string Discription of the thrown error.
     * @param      $errfile string File of which the error occured.

```

```
* @param      $errline      string Line that the error occurred at.
*/
function error_handler($errno, $errstr, $errfile, $errline) {
    if($errno != 8){
        switch ($errno) {
            case E_USER_ERROR:
                $error = "<b>PhishTank Runner Error:</b> [#$errno]
                $error .= " Fatal error in line $errline of file
                $error .= ", PHP " . PHP_VERSION . " (" . PHP_OS . ")<br
                />\n";
                die($error);
                break;
            case E_USER_WARNING:
                $error = "<b>PhishTank Runner Warning</b> [#$errno]
                $error .= " Fatal error in line $errline of file
                break;
            case E_USER_NOTICE:
                $error = "<b>PhishTank Runner Notice</b> [#$errno]
                $error .= " Fatal error in line $errline of file
                break;
            default:
                $error = "<b>PhishTank Runner Unknown Error</b> [#$errno]
                $error .= " Fatal error in line $errline of file
                break;
        }
        echo $error;
    }
}
?>
```