User Guide

What is BlackSEO Detector?

Well... That question is not easy to answer.

On the one hand, “BlackSEO Detector” is actually a Ruby Class. On the other hand, a working proof of concept has been made around this class, and it is called “BlackSEO Detector” too.

In this document, the name “BlackSEO Detector” will be used to reference the working proof of concept.

What does BlackSEO Detector do?

BlackSEO Detector uses Interenet search engines, like Bing or Google, to look for strange contents in web sites. These contents may be indicators of server or infrastructure compromise.

Afterwards, it checks the server response under a couple of conditons. The results of these tests may be used to try to find out whether or not there is an actual compromise.

A report is generated with all the results collected in this process.

Features

BlackSEO Detector works on batch mode. By default it does not output any text. This way, it can be use for scheduled tasks.

All BlackSEO Detector configuration is made through a configuration file. For details on its format see “Project Configuration File”.

BlackSEO Detector generates an HTML report file with the information collected.
BlackSEO Detector has the following features:

- **Support for Search Engines**: Currently, Google Custom Search API and Bing Search API are implemented. Users can implement their own classes for other search engines and integrate them into the application.
- **Multiple queries**: Multiple search engine queries can be combined in a project.
- **Language support**: Spanish and English translation files are included. Users can implement their own translation files and integrate them into the application.
- **Proxy support**. Connections can be made directly, through a proxy or using the configuration provided by the http_prox environment variable. (Ruby 2 or newer only)
- **E-mail reporting**: Test reports can be automatically sent by email.

Usage

BlackSEO Detector main file is poc.rb. The only command line argument supported is the configuration file path. So, you can use something like:

```
ruby poc.rb path_to_config/config.txt
```

...or, if your operating system is configured to run Ruby files directly, just:

```
poc.rb path_to_config/config.txt
```

Project Configuration File

The details about the Project Configuration File format are provided in the "Configuration file format" of "Class My_configuration" in the "Software Components" documentaion.

The configuration options that can be set are:

NAME	TYPE	DEEFAULT VALUE	COMMENTS
verbose	Boolean	N	When assigned a true value, the application will show messages as it makes requests, providing real time information about its activities. **Example:** verbose=Y
language	String	spanish	Language to use when creating the report **Example:** language=english
search_engine	Pairs mandatory		Sets a name to reference a search engine. The first element of the pair is the search engine name. The second element is the name of the Ruby class that implements the search engine **Example:** search_engine=!Google!Google_cse

search_engine_option	Pairs		Sets an option for search engine object initialization. The first element of the pair is the search engine name. The second element of the pair is the option. Values will be passed to object initialization method in the same order that they are provided in the configuration file. See “Class Search_engine”, “Class Goolge_cse” and “Class Bing_api” for more details. **Example:** search_engine_option=!Google!abcd search_engine_option=!Google!efgh
title	String	Report	The main title of the report **Example:** title=BlackSEO Detector test
report_file	String mandatory		Path for the main report file. If a relative path is provided, the directory where the configuration file is located will be used as its base. **Exampe:** report_file=reports/report.html
add_date_time	Boolean	Y	If true, the date and time when the test started will be appended to the report file name. **Example:** add_date_time=N
overwrite_report	Boolean	N	If true, when report file already exists it will be overwritten. **Example:** overwrite_report=Y
proxy	String	:ENV	Proxy server. **Example:** proxy=localhost
proxy_port	String		Proxy port **Example:** proxy_port=80
proxy_user	String		Proxy user **Example:** proxy_user=username

proxy_password	String		Proxy password **Example:** proxy_password=password
user_agent	Strubg	BlackSeo Detector	User Agent used by the application to make HTTP requests **Example:** user_agent=Custom UA
max_redirects	Integer	5	Maximum number of redirections to follow when making an HTTP request **Example:** max_redirects=2
hard_check	Boolean	N	If false, tests for a URL will end as soon as a problem is found. Otherwise, all tests will be made for all URLs. **Example:** hard_check=Y
max_report	Integer	25	Maximum number of problems to investigate in a report. When this value is met, the rest of the URLs will not be downloaded. **Example:** max_report=10
max_host	Integer	15	Maximum number of problems to investigate for a host. When this value is met, the rest of its URLs will not be downloaded. **Example:** max_host=10
max_folder	Integer	5	Maximum number of problems to investigate in a folder. When this value is met, the rest of its URLs will not be downloaded. **Example:** max_folder=1
send_mail	Boolean	N	If true, the generated report will be sent by email. **Example:** send_mail=Y
mail_server	String		Server to use for sending email **Example:** mail_server=localhost

mail_port	Integer	:default	Port for mail server. **Example:** mail_port=123456
mail_cypher	String	tls	Type of cypher to use in emails. Valid values are: none, tls, ssl **Example:** mail_cypher=none
mail_user	String		User name for email authentication **Example:** mail_user=mailuser
mail_password	String		Password for email authentication **Example:** mail_password=mailpasswd
mail_auth	String		Type of authentication for email. See Net::SMTP documentation for details **Example:** mail_auth=login
mail_from	String		Mail sender address **Example:** mail_from=sender@example.com
mail_to	String		Mail recipient address **Example:** mail_to=admin@example.com
mail_subject	String	BlackSeo Detector Results	Mail message subject **Example:** mail_subject=Test Results
mail_body	Multiple strings		Mail message body. Lines will be inserted into the mail body in the order provided in the configuration file. The following text replacements will be made: - Lines consisting in a single dot character will be replaced with empty lines. - [[problems_found]] will be replaced with the number of problems found - [[report_file]] will be replaced with tje report file path

			Example: mail_body=Tesk completed. mail_body=. mail_body=See [[report_file]].
Query	Pairs mandatory		Queries to make to search engines. The first element of the pair is the search engine name. The second element of the pair is the query to make **Example:** query=!Google!levitra site:example.com
allowed_domain	Multiple strings		Domains to include in the report. URLs not belonging to these domains will be ignored. Redirections pointing to domains not included in this list will cause a problem alert in the report. **Example:** allowed_domain=example.com allowed_domain=example.net
ignore_files	Multiple strings		Regular expressions. URLs matching them will be ignored **Example:** ignore_files=^.*generate_pdf=
text	Multiple strings		Texts that are used as hints for problematic situations. When one of this texts is found inside a response, URL or summary, a problem alert will be generated in the report. **Example:** text=levitra text=viagra

Configuration file example

```
# Langage
language=english

# Verbose mode
verbose=y

# Search Engines
search_engine==Google - CSE=Google_CSE
search_engine_option==Google - CSE=xxx_ api_key_xxxxx
search_engine_option==Google - CSE=yyy_engine_id_yyyy

search_engine==Bing - API=Bing_api
search_engine_option==Bing - API=zzz_api_key_zzzz

# Report title
title=Report for EXAMPLE.COM

# Report File
report_file=reports/example.com.html

# Must add date and time to file name?
add_date_time=n

# Overwrite report file if already exists
overwrite_report=y

# Keep looking for problems even if one has already been detected for the URL?
hard_check=y

# User agent to use in web access
user_agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0

# Proxy information. :ENV is used if no configuration is given
proxy=
proxy_port=
proxy_user=
proxy_password=

# Queries to make .There may be more than one 'query' line"
query==Google - CSE=site:example.com levitra online
query==Bing - API=site:example.com levitra online
```

```
# Text pointing to problems. There may be more than one 'text' line
text=viagra
text=levitra

# Maximum number of redirections to follow for each URL
max_redirects=5

# Domain allowed for reporting and redirection.
allowed_domain=example.com
allowed_domain=example.net


# Regular expression. URLs that match this regexp will not be analysed
ignore_files=.*generate_pdf

# Maximum number of files to download per report, host and folder
max_report=25
max_host=20
max_folder=10

# Whether or not to send email with results
send_mail=y

# Mail configuration
mail_server=mail.example.com
mail_port=
mail_cypher=tls
mail_user=mailusername
mail_password=mailpassword
mail_auth=login
mail_from=mailusername@example.com
mail_to=mailusername@example.com
mail_subject=BlackSEO Detector Report for EXAMPLE.COM
mail_body=A BlackSEO Detector report for EXAMPLE.COM has been generated
mail_body=.
mail_body=Problems found: [[problems_found]] (se attached file)
mail_body=.
mail_body=Full listing is located at [[report_file]]
```

Adding funcionality

Adding Search Engine Support

To add support for another Search Engine you must create a Ruby class.

This class must implement all public methods defined in “Class Search_engine”. There are some example of implementations in the “searchengines” folder of the BlackSEO Detector application.

The only method whose parameters may be different from the Search_engine implementation is initialize.

This class must be included in a Ruby code file with “rb” extension. The name of the file must be the same one as the Ruby class name.

The file must be located in the “searchengines” folder of the BlackSEO Detector application

Adding Language Support

To add support for a language, a translation file must be created in the “lang” folder of the BlackSEO Detector application.

The file extension must be “txt”. The file name must be the language name. For example: spanish.txt

The file must follow the specifications made in “Translation table file format” and translate the following definitions:

analysis_omitted_protocol:Analysis omitted for URL with non supported protocol - [[protocol]]
analysis_omitted_regexp:Analysis omitted. URL matches exlusion regexp
analysis_omitted_type:Analysis omitted for [[document_type]] file
authorized_domain:Authorized domain
cache_from:CACHE FROM
cannot_download:Could not download the resource
data:Data
desination_page_problems:Destination page may have security problems. Please check
destination_domain_problems:Destination domain may have security problems. Please check
details_for:Details for
dns_compromise:This may be an indicator of DNS server compromise
external_redirection:EXTERNAL REDIRECTION
file:FILE
fit_to_width:Fit to width

folder:FOLDER
forging_referer_of:FORGING REFERER OF
found:FOUND
generated_by:Report generated by BlackSEO Detector
hash_contains_text:URL hash string contains the text "[[text]]"
hide_data:Hide data
hide_show_analysis:Hide / Show analysis data
host:HOST
host_name_contains_text:Host name "[[host]]" contains the text "[[text]]"
html:HTML
html_response:HTML Response
http:HTTP
http_headers:HTTP Headers
info:Info
last_if_redirs:Last ones if redirections were followed
malicious_redirection:Malicious redirection?
max_problems_folder:Maximum number of problems investigated for folder met
max_problems_host:Maximum number of problems investigated for host met
max_problems_report:Maximum number of problems investigated in report met
more_details:Click here for more details
no_files_downloaded:No files were downloaded
no_mailicious_content:Could not find malicious content. Anyway, still there might be issues.
no_problems:No problems found
no_problems_in_url:No problems found in URL
no_redirection:NO EXTERNAL REDIRECTION
no_summary:No result summary to show
no_url_to_translate:Cannot find out URL to translate
not_found:NOT FOUND
number_of_problems_for_folder:Number of problems found for this folder:
number_of_problems_for_host:Number of problems found for this host
path_contains_text:Path "[[path]]" contains the text "[[text]]"
provided_by:Provided by
problems_already_detected:Problems already detected for this file
problems_found:problems found
query_string_contains_text:URL query string contains the text "[[text]]"
redirection_found:Redirection found
redirection_inside_domain:Redirection inside domain
redirection_problems:Redirection problems
redirection_to_authorized:REDIRECTION TO AUTHORIZED DOMAIN
regular_access:REGULAR ACCESS
response_code:Response Code
search_engine:Search Engine
server_compromise:This may be an indicator of server compromise
show_data:Show data

summaries:Summaries
summary:Summary
suspicious_content:Suspicious content
test:TEST
title:Title
titles:TITLES
translate_using:TRANSLATE USING [[search_engine]]
unauthorized_content:Unauthorized content? Found: "[[hint]]". Number of occurrences: [[number]]
url:URL
url_index_title:URL and Search engine indexes and titles
using_user_agent_of:USING USER AGENT OF

Software Components

DATA STRUCTURES

Search_engine_result

Hash used by Search Engine Object to represent a result
Keys:

:url	Result URL
:referer	URL for the search engine request that provided the result
:document_type	Document type (PDF, DOC, etc.)
:title	Result title
:summary	Result summary
:cache	URL for the search engine cache copy of the document
:translate	URL for translation of the document
:bot_user_agent	Search engine bot User Agent
:s_e_object	Search engine object that provide the result

Search_engine_response

Set of results provided by a search engine for a query.
It is an array of Search_engine_result

Deferred_engine_result

Hash used by Deferred Information objects to return URL information for cache or translation
Keys:

:url	URL for document cache or translation
:referer	Referer to use when requesting the URL

BSD_result

Hash used by BlackSEO_detector to represent results for a given URL.
Keys are Search engine names
Values are a Search_engine_result data structures provided by the Search Engine

BSD_URL_info

Hash used by BlackSEO_detector to represent the information collected for a URL
Keys are URLs.
Values are BSD_results for that URL.

Report_info

Hash used by the report generator to receive items (tipically, report lines or tables)

Keys:

:level	Valid values are :alert, :warning, :ok and :info
:info	Data item
:headers	Headers for tables
:parameters	Parameters for text translation

CLASSES

Class Translator

This class provides translation for HTML messages.

Instance Methods

- **initialize(file_path)**
  Creates a new instance. Its translation table will be loaded from the file given by file_path.
- **load_table(file, add = false)**
  Opens the file given by file_path, parses its contents and loads them into the translation table. If add is false, translation table will be deleted prior to loading the file.
- **translate(input, tokens={})**
  Translates an input, using the information provided by the hash “tokens”. Tokens have the format:
  <token> => <value>
  Where <token> is a Symbol and <value> is a String.

How translation works

If input is a Symbol, it will be looked up in the “tokens” hash to find its translation. If it is not found, the translation table will be used.

If it is a String, text inside double brackets (“[[...]]”) will be converted to Symbol and processed as above. The result will be used to replace the brackets and its contents.

Translation table file format

Translation table files are text files whose lines have the format:
<token>:<value>

Where <token> characters may be letters, digits and undesrcores (“_”).
When loaded into the translation table, token will be converted to Symbol.

Empty lines and lines starting with the character “#” are ignored. There may be blanks at the beginning of the line. All extra blanks at the beginning and the end of the line will be removed.

Class My_configuration

Class used to load and provide configuration information to the application.

Instance Methods

- **initialize(definitions)**
  Creates a new instance and load the configuration option definitions (see below for details)
- **read(file_path)**
  Loads configuration from the file given by file_path
- **[]=(option_to_assign, value)**
  Allows use of brackets as with Hashes. The index should be an option name, provided as a symbol
- **[](option)**
  Allows use of brackets as with Hashes
- **to_s**
  Converts configuration to a human readable string (...)

Definitions format

The definitions parameter for the initialize method is a Hash with the following keys:

- **:name**
  Name of the parameter. Only letters, digits and underscore (_) are supported.
- **:kind**
  May be one of the following:
  - **:simple** (one value) - This is the default value
  - **:multiple** (array of values)
  - **:pairs** (array of hashes)
- **:mandatory**
  Boolean value; default is false. When true, this configuration option must be assigned a value before using it
- **:type**
  Type of the values. May be:
  - **:string** This is the default value
  - :**integer**
  - **:boolean.**
    When assigning values to a Boolean option, the value will be considered true if it is one of the following “y”, “t”, “yes” or “true”. This check is not case sensitive.
- **:default**
  Default value
- **:allowed**
  Sometimes we need to store some values with types different from the one given to the option. Values in this array will be allowed regardless of their type. If

only a value is given, an array will be created with it as its only element.

Configuration file format

The configuration file is a text file. Its lines have the following formats

- **<option>=<value>**
  Used for simple and multiple value options. If it is a simple option, <value> will be assigned to it. If it is a multiple one, <value> will be added to its array of values.

- **<option>=<sep><key><sep><value>**
  Used for pairs values. Both instances of <sep> must be the same character. A hash { key => value } will be added to the array of hashes.

Empty lines and lines starting with “#” will be ignored. All extra blanks at the beginning and the end will be removed.

Class My_HTTP_client

Class for making HTTP / HTTPS requests.

Class Methods

- **detect_redirection(response)**
  Checks a Net::HTTPResponse for redirections. Returns an array with three elements:
  - i**s_there_a_redirection**
    True if response contains a redirection
  - **redirection_type**
    May be :http or :html
  - **url_to_redirect**
    URL for the redirection
- **repair_url(url)**
  Makes correction to a string with a URL to make sure that this URL is valid

Instance Methods

- **initialize(user_agent="My HTTP Client", proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)**
  Creates a new object with the User Agent and proxy information provided.
  Ruby 2 uses the proxy info provided by the environment variable “http_proxy” when “proxy” = :ENV. If no proxy must be used, proxy should be assigned a nil value.
- **set_server(protocol, server, port=:default, update_cookies=false, usr=nil, pwd="")**
  Provides information about the host that will be used. The parameter “protocol” may be :http or :https. When “update_cookies” is true, cookies returned by the server will be stored and used in next request. “usr” and “pwd” will be used as credentials for simple HTTP authentication
- **change_cookie(new_cookie)**
  Sets a cookie. It must be provided as a string
- **change_user_agent(new_user_agent)**
  Sets client user agent
- **change_auth(usr, pwd)**
  Changes credentials for simple HTTP authentication
- **post(path="/", get_parameters={}, post_data="", headers={}, retries=2)**
  Makes a POST request. If there are problems, the maximum number of request retries will be given by “retries”. “get_parameters” and “headers” will be a hash of “name” => “value”, being both name and value strings. “post_data” may be a hash too, to represent a post form, or a string to make a raw POST request

- **get(path="/", get_parameters={}, headers = {}, retries=2)**
  Makes a GET request. See “post” method for parameter explanation.
- **generic_request(path="/", get_parameters={}, post_data="", headers={}, method=:get, max_retries=2)**
  Utility method that allows both GET and POST requests. “method” may be :get or :post. See “post” method for information about other parameters.

Class My_mailer

Class to send e-mails

Instance Methods

- **initialize(server, port=:default, cypher=:tls, user=nil, password=nil, auth_type=nil)**
  Sets server information.
  - **server**
    Mail server
  - **port**
    May be :default or a port number
  - **cypher**
    May be :ssl, :tls or :none
  - **auth_type**
    See Net::SMTP documentation for details. A typical value would be :login
- **send(body, from, to, subject="", body_content_type="text/plain", attachments = [], cc=nil, bcc=nil, delim="___MAIL_PART_BEGINS_ENDS")**
  Sends an e-mail
  - **body**
    Text or HTML code for the body
  - **body_content_type**
    MIME type for the body. Typical values would be “text/html” or “text/plain”
  - **attachments**
    Hash with these keys:
    - **:file_name** : path for the file
    - **:content_type** : MIME type for the file
  - **delim**
    String that will be used internally to mark the parts of the message.

Class Search_engine

Class to represent a generic Internet Search Engine.

Instance Methods

- **initialize(protocol, server, port=:default, path="/", query_parameter="q", regexps = {}, paging = {},**
  base_get_parameters ={}, base_post_parameters="", headers={},
  user_agent="Search Engine Requester",
  search_engine_bot = nil,
  max_requests=5,
  request_method=:get, update_cookies=false,
  query_parameter_method=:get,
  proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil,
  max_redirs=5)

Creates a new instance.

- **protocol**
  May be :http or :https
- **server**, **port** and **path**
  Represent the base URL for queries
- **query_parameter**
  Parameter where the query is sent
- **regexp**
  Hash of regular expressions to match result data. Each one, with the exception of the ones in ":replaces" must have a submatch identified by the name "data"
  - **:replaces**
    Hash of <regexp> => <text> replacements to make before parsing the HTML response from the search engine
  - **:begin_result**
    Regular expression to match the beginning of a result
  - **:end_results**
    Regexp to match the ending of all results
  - **:document_type**
    Regexp to match the document type of the result
  - **:title**
    Regexp to match result title
  - **:url**
    Regexp for the URL
  - **:summary**
    Regexp for the summary
  - **:cache**
    Regexp for the search engine cache URL for the result

- **:translate**
  Regexp for the search engine translation of the result

- **paging**
  Information about how to manage result paging. It is a hash with the following keys:
  - **:parameter**
    Parameter name for page number
  - **:parameter_method**
    How the parameter is send: :get, :post, :cookie
  - **:size_parameter**
    Parameter for page size
  - **:size_parameter_method**
    How size parameter is sent: :get, :post, :cookie
  - **:size_parameter_value**
    Value to assign to the size parameter
  - **:size**
    Actual page size as integer
- **base_get_parameters, base_post_parameters, headers**
  Hashes of "name"=>"value" pairs for parameters
- **user_agent**
  User Agent to use in requests
- **search_engine_bot**
  User Agent of the Search Engine bot
- **max_requests**
  Maximum number of requests to make when "request_all" method is invoked.
- **request_method**
  How the request is sent: :get, :post
- **update_cookies**
  Whether to update or not cookies with each server response
- **query_parameter_method**
  How the query parameter is sent: :get, :post, :cookie
- **proxy, proxy_port, proxy_user, proxy_password**
  Proxy configuration, if needed. Ruby 2 uses the proxy info provided by the environment variable "http_proxy" when "proxy" = :ENV. If no proxy must be used, proxy should be assigned a nil value.
- **max_redirs**
  Maximum number of redirections to follow on each request.

- **connect_through(proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)**
  Sets proxy information if needed. See above for details on parameters.

- **set_deferred(deferred)**
  Some search engines don't provide all data in the results page. Cache or

translation URLs are examples of information items that may require additional requests. To avoid making too much requests, these one should be made only if they are necessary. A deferred information object may manage these scenarios. See “Deferred information objects” chapter for more details.

- **deferred_info(type, result, only_one=false)**
  Gets information on demand using the deferred object. Returns a Deferred_engine_result data structure. Parameters:
  - **type**
    Kind of data requested: :cache, :translate
  - **result**
    The search engine result (see Search_engine_result data structure for details)
  - **only_one**
    True if only a URL is needed. This way, we save search engine requests.
- **request(query, page=1)**
  Makes a request to the search engine for “query” to get a SERP. The number of the SERP is given by the “page” parameter. Returns a Search_engine_response data structure.
- **request_all(query, pause=1, max_results=1000)**
  Requests all data from the Search Engine for the given query. “pause” is the number of seconds to pause before making each request. “max_results” is the maximum number of results needed. Returns a Search_engine_response data structure.

Class Goolge_cse

Class to make requests to Google. Subclass of Search_engine

Redefined public instance methods

- **initialize(key, engine, user_agent="G-API-RubyProg", max_requests=3, proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)**
  - **key**
    Google API Key
  - **engine**
    Google CSE engine id
  - **user_agent**
    User Agent to use in requests
  - **max_requests**
    Maximum number of requests to make when “request_all” method is invoked.
  - **proxy, proxy_port, proxy_user, proxy_password**
    Proxy configuration, if needed. Ruby 2 uses the proxy info provided by the environment variable “http_proxy” when “proxy” = :ENV. If no proxy must be used, proxy should be assigned a nil value.

Class Bing_api

Class to make requests to Bing. Subclass of Search_engine

Redefined public instance methods

- **initialize(key, user_agent="B-API-RubyProg", max_requests=3, proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)**
  - **key**
    Bing API Key
  - **user_agent**
    User Agent to use in requests
  - **max_requests**
    Maximum number of requests to make when “request_all” method is invoked.
  - **proxy, proxy_port, proxy_user, proxy_password**
    Proxy configuration, if needed. Ruby 2 uses the proxy info provided by the environment variable “http_proxy” when “proxy” = :ENV. If no proxy must be used, proxy should be assigned a nil value.

Deferred information objects

Not actually a class, but a set of methods a Deferred information Object must provide

Instance Methods

- **initialize(user_agent, proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)** :
  - **user_agent**
    User Agent to use in requests
  - **proxy, proxy_port, proxy_user, proxy_password**
    Proxy configuration, if needed. Ruby 2 uses the proxy info provided by the environment variable “http_proxy” when “proxy” = :ENV. If no proxy must be used, proxy should be assigned a nil value.
- **connect_through(proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil)**
  Sets proxy information if needed. See above for details on parameters.
- **get_info(type, result, only_one=false)**
  Gets information on demand. Returns a Deferred_engine_result data structure. Parameters:
  - **type**
    The kind of data requested: :cache, :translate
  - **result**
    The search engine result (see Search_engine_result data structure for details)
  - **only_one**
    True if only a URL is needed.

Class Report_generator

Class for report generation

Instance methods

- **initialize(file, overwrite=false, lang='lang/spanish.txt')** :
  Create a new instance
  - **file**
    Path for the main report file
  - **overwrite**
    If true and report file exists, it will be overwritten
  - **lang**
    Language file for translations. Must have format as indicated in “Translation table file format”
- **begin(title)**
  Starts writing the report. The main header for the report will be the one provided in “title”
- **end**
  Ends the report
- **begin_test(title)**
  Starts a TEST section. Its header will be provided by “title”.
- **end_test**
  Ends a TEST section
- **begin_host(host)**
  Starts a HOST section. Its header will contain the text contained in “host”
- **end_host**
  Ends a HOST section
- **begin_folder(folder)**
  Starts a FOLDER section. Its header will contain the text contained in “folder”
- **end_folder**
  Ends a FOLDER section
- **begin_file(file)**
  Starts a FILE section. Its header will contain the text contained in “file”
- **end_file**
  Ends a FILE section
- **not_tested(url, bsd_result, text, params={})**
  Shows information about a URL for which no tests were made
  - **url**
    URL not being tested
  - **bsd_result**
    Data structure with info about the URL

- **text**
  String or symbol with info about why the URL was not tested. It will be translated using a Translator object.
- **params**
  Parameters for text translation

- **problem(url, bsd_result, analysis, to_highlight)**
  Shows information about a URL with problems.
  - **url**
    The URL
  - **bsd_result**
    Data structure with info about the URL
  - **analysys**
    Report_info data structure with info about the tests made.
  - **to_highlight**
    Array of strings to highlight in the text of responses, summaries, etc.
- **def maybe_ok(url, bsd_result)**
  Shows information about a URL with no problem found.
  - **url**
    The URL
  - **bsd_result**
    Data structure with info about the URL

Report structure

The report has the following components:

- One main HTML report file, with information about the Search Engine results and analysis, URL analysis and server responses.
- One detail HTML file for every URL with problems. This one has one column for every test made. Information about test results, HTTP headers sent by the server and its HTML contents is provided in this file.

The report uses JavaScript to allow hiding and showing elements of the report and adjust table columns width.

The detail files are stored in a directory that is stored in the same directory as the mail report file.

The name of the detail file directory is the same of the mail file with a ".dir" end.

Class BlackSEO_detector

Class to analyze unauthorized web modifications used for BlackSEO techniques

Instance methods

- **initialize(search_engines, user_agent="BlackSEO-detector", proxy=:ENV, proxy_port=nil, proxy_user=nil, proxy_password=nil, verbose=true)**
  Creates a new object
  - **search_engine**
    Hash of “search engine name”=>search_engine_object
  - **user_agent**
    User Agent to use in requests
  - **proxy, proxy_port, proxy_user** y **proxy_password**
    Proxy configuration, if needed. Ruby 2 uses the proxy info provided by the environment variable “http_proxy” when “proxy” = :ENV. If no proxy must be used, proxy should be assigned a nil value.
- **investigate(queries, text, report_generator, report_name='Test', hard_check=false, max_redirects=5, valid_domains=[], ignore_files=[], max_report=20, max_host=12, max_folder=5)**
  Makes a test
  - **queries**
    Hash of “search engine name” => “query” with the queries to make to each search engine
  - **text**
    Array of texts used as hints to detect unauthorized modifications
  - **report_generator**
    Object used for report generation. See “Class Report_generator” for details
  - **report_name**
    Main title for the report
  - **hard_check**
    If true, the object will keep making tests for a URL even when problems have already been detected for it. Otherwise, tests will end as soon as a problem is detected for the URL
  - **max_redirects**
    Maximum number of redirections to follow when requesting data to the server
  - **valid_domains**
    Array of domains allowed to be included in the report an to follow redirections to. If a redirection to a domain not included in this array is detected, it will not be followed.
  - **ignore_files**
    Array of regular expression. URLs matching any one of them will not be tested

- **max_report, max_host, max_folder**
  Maximum number of URLs to investigate for report, host and folder.

Table of Contents