

SMART GUARDIAN

DEVELOPERS GUIDE

Table of Contents

1.WALL MODULE.....	3
1.Authenticator Class.....	3
2.Rule Class.....	3
3.Policy Class.....	3
4.Verifier Class.....	4
2.CRAWLER MODULE.....	4
1.Crawler Class.....	4
2.ContentAnalyzer Class.....	4
3.CONTROLLER MODULE.....	4
Config .php.....	4
HtmlFunctions.php.....	5
UserProfile.php.....	5
ViewAllUser.php.....	5
DomainClassDisplay.php.....	5
RulesDislay.php.....	5
SettingsFunctions.php.....	5
Admins.php.....	5
Users.php.....	5
UsersDisplay.php.....	6
ReporterFunctions.php.....	6
StatisticsFunctions.php.....	6
StatisticsDisplay.php.....	6
StatisticsDelete.php.....	6
Login.php.....	6
Logout.php.....	6
4.SHARED SUB-MODULES.....	6
1.ConfigurationFileReader.....	6
2.HtmlParser Sub-Module.....	7
HtmlElement Class.....	7
HtmlParser Class.....	7
3.DatabaseCommunicator Sub-Module.....	7
4.WebConnector Sub-Module.....	7
Socket Class.....	7
HttpRequest (Request) Class.....	7
HttpResponse(Response) Class.....	8
ConnectionHandler Class.....	8

Project files of SmartGuardian are separated in three main directories; src, data and doc. In doc directory documentation files such as Developers Guide, Installation manual & User Manual exist. In data directory; configuration files and other necessary data such as authenticator interface pages, SQL scripts and training data files for Crawler take place. In src C++ and PHP sources files of project. In src there are four subdirectory exist and three of them include each of Project Module source files; Wall, Crawler and Controller. The other subdirectory with name Shared keeps the files of shared sub modules.

1.WALL MODULE

Wall is the core module of the project since it serves clients and it communicates with other modules and coordinates them. Most important classes of Wall are described below.

1. Authenticator Class

As its name implies Authenticator provides an authentication mechanism to client users when they attempt to access to Internet. Authenticator also identifies the user with their usernames and passwords. That enables the system to response each user in a way that which policy will be applied to which user. After the clients' identification process, the requests are forwarded to Verifier Class.

2. Rule Class

When a request or a response arrives, it's the defined rules that determine whether request or response should be allowed or denied. The action method of the Rule class is the action to take when a request or response matches that rule. If the value of this attribute is true the request/response is allowed, otherwise it is denied. The log attribute is designed to determine whether the request/response that matches this rule should be logged but not implemented yet. The priority attribute is very important in that, when a request/response matches more than one rule, the rule having the highest priority is applied. The other attributes, which are applications, domainClasses, FileRules and timeRange, are used while matching a request/response to a rule. Requests's useragent field is compared with applications of rule, if match rule then other attributes will be considered. Similarly timeRange request/response time is checked with that attribute in a way that it is in this interval or not. DomainClasses keeps a set of domains and the domain of the requested page will be compared with each of domains in that domain class. Port Range & Protocol attributes are designed to be developed but they are not implemented yet.

3. Policy Class

A policy is a set of rules that determines the action (allow/deny) of SmartGuardian Wall to take when a request or a response to a request arrives, and the policy class keeps the information of a policy. Its rules attribute keeps the rules that form the policy. insertRule method adds the given rule to the policy object. This method is used while creating the policy object. Another method of this class is the verifyRequest method. Given a request, this method decides whether the request is to be allowed or denied. It tries to match the request with the rules in the policy. When a rule matches the

request, depending on the action property of that rule the request is allowed or denied. `verifyResponse` is anonymous to `verifyRequest`. It checks whether a response to a request should be allowed or denied.

4. Verifier Class

Veriefier class keeps all the policies in policies vector and the policies assigned to each userGroup. This class always has one instance which is a global variable used by all the threads handling the clients' requests. All the rules' and policies' information are stored to these instance of this class during the initialization of the system. It has an overloaded method with name `verify` its argument is either a request or a response. This method finds the policy that will be applied to given request/response according to the user that makes the requests. Then applies the corresponding policy to resquest/response and decides whether the request/response should be allowed or denied. Afterward the allowed attribute of request /response class is set according.

2.CRAWLER MODULE

1. Crawler Class

Crawler class is for crawling the web sites in domain classes and generating word - frequency lists. It uses `ContentAnalyzer` class for getting the words out of web sites (i.e. stripping the html tags) and uses `DatabaseCommunicator` class for writing the data to the database. The word, frequency table is updated on the database and new words are added to it. The *classify* method of this class finds the domain class to which the given web site belongs.

2. ContentAnalyzer Class

Tjis class generates the necessary information for classifying a web page. The *analyse* method of this class analyses the given HTML code and generates a word&frequency map that will be used by Crawler class for guessing the domain class of the given page.

3.CONTROLLER MODULE

SmartGuardian Contrroller is totally web based, it checks, the incoming requests, (Administrative Command) for authentication. If the user (network admin) is not logged in, it displays the login page. Login requests and responses are handled by its sub module called Admin Authentication. This module handles all the administrative requests and displays the results of these requests. It can also display system logs via its sub module called Reporter. Below are the explanations file of this module.

Config .php

Provides a configuration to related PHP files in order not to type same data all the time and provide a brief interface to user to apply changes.

HtmlFunctions.php

Have functions to adjust html header, html body and tables in definite structure. Besides that it keeps function confirmdelete that ask administrator when applying a deletion of a row in database.

UserProfile.php

It provides a set of functions for add, display, insert, delete, and update etc. choices for both user group and users of the system.

ViewAllUser.php

It query User Group table in database and for each group it creates a sub tables in main frame and query users in that group to display with a update and delete options.

DomainClassDisplay.php

It query DomainClass and Domain tables and it provides a set of functions to administrator such as; add, delete, insert, update and display both DomainClasses and domains in DomainClasses.

RulesDislay.php

This file contains most important functions and most frequently used functions of the Controller module. It consists of getter & adder functions of policy, rule and every attributes in a rule. It provides a main which handle get & post methods of links and buttons in interface. Other PHP files in directory keeps insert, delete, update functions of each table in database. Besides having frequent queries to database of system, display functions in source file provides an indented tables to administrator to easily recognize the result of queries.

SettingsFunctions.php

Adjustment of body of the window of a page for user administrative purposes is being done by functions in here. Here User and Admin directories exist each have PHP files for achive modification purposes of Admin & User tables in Database.

Admins.php

It both consists of main loop & admin functions which provide add, delete, update purposes which effects Admin table in database.

Users.php

Main functions of users' methods handle the conditional statements of get & post requests of administrators.

UsersDisplay.php

This PHP file provides display methods for user, user group tables of the database.

ReporterFunctions.php

This php file provides routine GET POST handler functions and display functions for the skeleton of reporter pages.

StatisticsFunctions.php

Some of the GET POST variable handler functions for statistics pages are implemented in this file.

StatisticsDisplay.php

The functions involving displaying the statistics of the system usage are implemented in this file. There are also search functions which query the database for specified input. GET and POST variable processing is also necessary for properly displaying the statistics so there are some helper functions which handle those tasks.

StatisticsDelete.php:

This file contains query functions which delete the statistics for some given criteria.

Login.php:

This file contains login control functions and display functions for login forms. Moreover every index file in controller includes this file and calls its loginMain() method to prevent unauthorized access to the system.

Logout.php

Implements methods for logging out from the system.

4.SHARED SUB-MODULES

This directory contains sub modules which are used by both Wall & Crawler Module.

1. ConfigurationFileReader

As it's understood from the name this module is used for reading the configuration files. Each line of the configuration files contains a name & value pair, separated by an "=" character. The constructor method of this class takes a file name as an argument, parses the file and creates the name & value map.

2. HtmlParser Sub-Module

HtmlElement Class

An HtmlElement can represent a start tag (e.g. <HTML>, , etc.), an end tag (e.g. </HTML>), the text between two successive tags, or the comments in the HTML file (e.g. <!-- COMMENT -->). Each of these elements have types START_ELEMENT, END_ELEMENT, TEXT_ELEMENT, COMMENT_ELEMENT respectively.

HtmlParser Class

It keeps a vector of HtmlElement's. An HtmlElement can represent a start tag (e.g. <HTML>, , etc.), an end tag (e.g. </HTML>), the text between two successive tags, or the comments in the HTML file (e.g. <!-- COMMENT -->). parse method parses the given HTML code and creates the HTML elements, so that the code can be manipulated. It returns the number of characters parsed. On success this is equal to the bufferLength. In case of an error it just stops parsing. removeScripts method sets the start elements and end elements elements with name script to null elements, so that these elements are ignored. operator method returns the reference to the "index"th element, so that it can be used as an array. addElement method adds the given element to the end of elements. DumpHtml method creates an HTML code using the current elements and returns the code. DumpText method returns a string containing the text part of the parsed HTML code. readElement method reads from the "readBuffer", creates elements and store the elements to the "elements" attribute. Returns the number of characters read.

3. DatabaseCommunicator Sub-Module

All the database operations are done by this class. Other classes use this class's methods to communicate with related tables in database. Methods use MySQL C APIs to access, query and get results to/from MySQL database. readDatabaseInfo method reads database information such as database name, username, password and hostname from the "Database.conf" file. This information is used by connect method in order to connect database.

4. WebConnector Sub-Module

Socket Class

Socket class provides basic socket functions such connect, listen, bind, accept, read, write etc. with a user friendly interface.

HttpRequest (Request) Class

This class reads requests of clients' and obtains the information necessary for the verification process. It parses and rewrites the http header. The parsed header information is kept in the *header* attribute which is of type httpRequestHeader.

HttpResponse(Response) Class

This class reads responses to clients' requests and obtains the information necessary for the verification process.

ConnectionHandler Class

It uses both HttpRequest & HttpResponse to obtain related information when a connection establishes to the server then it forwards data to Verifier to determine whether the request is ought to be allowed or denied. After processing it Verifier set allow field of the Requests and ConnectionHandler checks this field. If allowed set true, then it open a connection to the destination of the request, send the requests and read the response for the request. Response is processed and verified by Verifier and again Response allow attributes sets by Verifier and ConnectionHandler sends the response to the clients if allow is true, otherwise sends a block message.