API Documentation

December 12, 2005

Contents

1 Module malcontent

This package is an implementation of an exploit detection analyzer for the Orchid system and several rules for detecting specific types of exploits.

1.1 Class ContentRule

malcontent.Rule
└
ContentRule

This rule type matches regular expressions against the raw content of the pages.

1.1.1 Methods

__init__(*self*, *reMap*, *reFlags*=66)

Creates a new ContentRule.

Parameters

`reMap`: a map from regular expression strings to tuples of the form (exploitName, level) where exploit name is the exploit name and level is the badness level like Rule.EVIL

__call__(*self*, *site*)

Applies the rule to the given site

Overrides: malcontent.Rule.__call__

1.1.2 Class Variables

Name	Description
Inherited from Rule: EVIL *(p. 5)*, GOOD *(p. 5)*, MAY_BE_EVIL *(p. 5)*	

1.2 Class ExternalIframeRule

malcontent.Rule
└
ExternalIframeRule

This rule type identifies IFRAME elements which load content from external servers.

1.2.1 Methods

__init__(*self*)

Creates a new ExternalIframeRule

__call__(*self*, *site*)

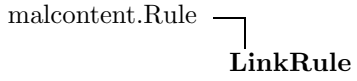
Applies this rule to the given site

Overrides: malcontent.Rule.__call__

1.2.2 Class Variables

Name	Description
Inherited from Rule: EVIL *(p. 5)*, GOOD *(p. 5)*, MAY_BE_EVIL *(p. 5)*	

1.3 Class LinkRule



This rule applies regular expressions to certain link types.

1.3.1 Methods

__init__(*self*, *reMap*, *reFlags*=66)

Creates a new LinkRule

Parameters

`reMap`: a map which maps regular expression strings to tuples of the form (exploitName, linkTypes, level) where exploitName is the name of the exploit, linkTypes is a list of link types on which to match the regular expression (see orchid.OrchidExtractor for more detail), and level is the level of maliciousness for example: Rule.EVIL

__call__(*self*, *site*)

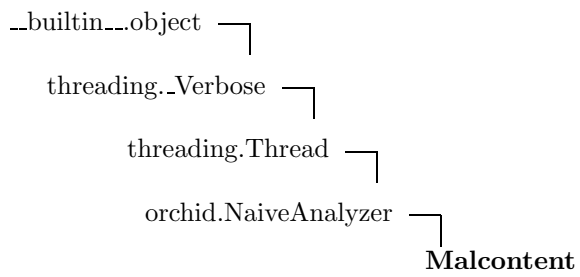
Applies the rule to the given site.

Overrides: malcontent.Rule.__call__

1.3.2 Class Variables

Name	Description
Inherited from Rule: EVIL *(p. 5)*, GOOD *(p. 5)*, MAY_BE_EVIL *(p. 5)*	

1.4 Class Malcontent



This is a concrete analyzer which used together with the Orchid crawler to detect malicious web pages based on a given set of rules.

1.4.1 Methods

__init__(*self*, *linksToFetchAndCond*, *siteQueueAndCond*, *db*, *rules*)

Creates a new malicious content analyzer.

Parameters

`rules:` a list of Rule objects to be applied against crawled sites.

Overrides: orchid.NaiveAnalyzer.__init__

addSiteToFetchQueue(*self*, *lfs*)

Add the sites we extracted in analyzeSite to the "to fetch" queue.

Overrides: orchid.NaiveAnalyzer.addSiteToFetchQueue

analyzeSite(*self*, *db*, *site*)

Applies all the available rules to the given site and extracts the links that we intend to crawl. Currently we follow regular ('<a...'), frame, iframe and script links.

Overrides: orchid.NaiveAnalyzer.analyzeSite

report(*self*)

Logs the results of the crawl.

Overrides: orchid.NaiveAnalyzer.report

selectNextUrl(*self*)

Select the next url to crawl to. This is done by selecting a random domain and then taking one page from it's queue.

Overrides: orchid.NaiveAnalyzer.selectNextUrl

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__, __str__

Inherited from NaiveAnalyzer: getNumSitesProcessed, reorganizeByDomain, run, setStopCondition
Inherited from Thread: __repr__, getName, isAlive, isDaemon, join, setDaemon, setName, start

1.5 Class Rule

Known Subclasses: ContentRule, ExternalIframeRule, LinkRule

An abstract class representing rules for detecting malicious content.

1.5.1 Methods

__call__(*self*, *site*)

Applies the rule to the given site. Should be overriden by real rules.

1.5.2 Class Variables

Name	Description
EVIL	**Value:** `2` *(type=**int**)*
GOOD	**Value:** `0` *(type=**int**)*
MAY_BE_EVIL	**Value:** `1` *(type=**int**)*

2 Module orchid

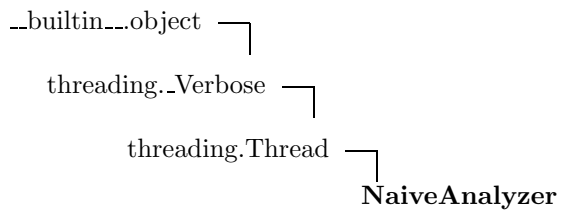
This package is a multi-threaded generic web crawler. For more detail about the package please see the attached paper.

2.1 Functions

extractServerName(*stringUrl*)

Extracts the domain name from a string URL and returns it.

2.2 Class NaiveAnalyzer



Known Subclasses: Malcontent

This is an interface for the analyzers. To use the Ugrah crawler subclass this class with you logic of analyzing the gathered data and choosing which links to crawl.

2.2.1 Methods

__init__(*self*, *linksToFetchAndCond*, *siteQueueAndCond*, *db*)

Creates a new analyzer. There can be as many analyzers as you like, depending on the type of processing of data you wish to do.

Parameters

`linksToFetchAndCond:`	A tuple of the form (map, Condition) where the map is the map of domains to links to be fetched (which is updated by the analyzer) and the condition is a `threading.Condition` object which is used to synchronize access to the list.
`siteQueueAndCond:`	A tuple (siteQueue, siteCond). The siteQueue is a queue unto which new sites (with their links) are inserted. The siteCond is a Condition object used to lock the queue and signal the analyzer to wake up.
`db:`	a tuple of the form (siteDb, siteDbLock) where the linkDb is the global link database, siteDb is the global site database and the lock is used to synchronize access to both of these databases.

Overrides: threading.Thread.__init__

addSiteToFetchQueue(*self*, *lfs*)

Adds links to the fetch queue. A real analyzer should override this method.

analyzeSite(*self*, *db*, *site*)

Processes the site and adds it to the db. Any real analyzer should override this method with it's own logic.

getNumSitesProcessed(*self*)

Returns the number of sites this analyzer has processed

reorganizeByDomain(*self*, *listOfLinks*)

Returns a map which maps domain names to links inside the domain.

report(*self*)

A real analyzer should override this method. Outputs the results of the analysis so far.

run(*self*)

Performs the main function of the analyzer. In this case, just adds all the hyperlinks to the toFetch queue.

Overrides: threading.Thread.run

selectNextUrl(*self*)

Chooses the next url to crawl to. This implementation will select a random domain and then crawl to the first link in that domain's queue.

setStopCondition(*self*, *val*)

Sets the stop condition to the specified value. Should be True to stop the analyzer thread.

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__, __str__
Inherited from Thread: __repr__, getName, isAlive, isDaemon, join, setDaemon, setName, start

2.3 Class Orchid

The main class of the crawler. Use this to start the crawling process.

2.3.1 Methods

__init__(*self*, *seed*, *fetcherThreads*, *maxUrlsToCrawl*, *timeOut*, *delay*, *analyzer*=<`class 'orchid.NaiveAnalyzer'`>, *args*=`[]`)

Creates a new crawler.

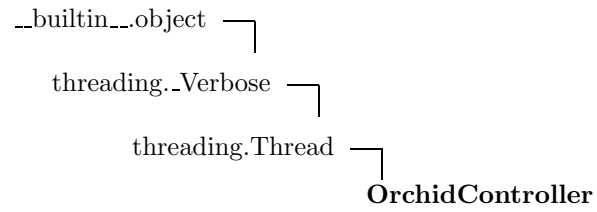
Parameters

`seed:`	A map of domain names to urls in that domain from which to start crawling.
`fetcherThreads:`	The number of fetcher threads to use.
`maxUrlsToCrawl:`	How many pages to crawl.
`timeOut:`	The socket timeout for loading a page.
`delay:`	The delay between crawls.
`analyzer:`	The class of the analyzer to use.

crawl(*self*)

Performs the crawling operation.

2.4 Class OrchidController



This class is responsible for controlling the fetchers and distributing the work load.

2.4.1 Methods

__init__(*self*, *linksToFetch*, *siteQueueAndCond*, *analyzer*, *numFetchers*=1, *maxFetches*=100, *socketTimeOut*=20, *delay*=5)

Creates a new controller. Typically you will need only one.

Parameters

`linksToFetch:`	A tuple of the form (Dict, Condition) where the dict is the map of domains to links to be fetched (which is updated by the analyzer) and the condition is a `threading.Condition` object which is used to synchronize access to the list.
`siteQueueAndCond:`	a tuple of the form (list, cond) where list is the site queue into which fetchers insert new sites that have been fetched. cond is a Condition object which is used to lock the queue.
`analyzer:`	The analyzer which we are using for analyzing crawled data.
`numFetchers:`	The number of active threads used for crawling.
`maxFetches:`	Maximum number of pages to crawl.
`socketTimeOut:`	The timeout to use for opening urls. WARNING: the timeout is set using socket.setdefaulttimeout(..) which affects ALL the sockets in the multiverse.
`delay:`	The delay in seconds between assignments of urls to fetchers.

Overrides: threading.Thread.__init__

getFetcherThreadUtilization(*self*)

Returns a list of number of urls each fetcher thread handler.

getNumFetchersUsed(*self*)

Returns the number of fetchers that handled at least one url.

run(*self*)

Runs this controller thread. The controller will use it's fetchers to fill the links and sites databases.

Overrides: threading.Thread.run

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__, __str__
Inherited from Thread: __repr__, getName, isAlive, isDaemon, join, setDaemon, setName, start

2.5 Class OrchidExtractor

A class responsible for parsing and analyzing html content and extracting various forms of links from it. The goal is to provide the user with a datastructure representing the parsed HTML and a set of links that appear on the given page. Supported links type are:

1. simple link type links (done)
2. relative links

3. BASE tag (done)
4. LINK tag (done)
5. FRAMESET links (done)
6. IMG links (done)
7. client side image maps (done)
8. server side image maps (unsupported)
9. Object links
10. JavaScript links
11. Meta refresh tags
12. iframes
13. form links

2.5.1 Methods

__init__(*self*)

Creates a new link extractor. Should be followed by a call to setSite

extract(*self*)

Extracts all the links in the page according to the patterns specified in LINK_PATTERNS. The links are stored in a map (link type -> url list) called links (accessible by 'extractor.links' where extractor is an instance of HtmlLinkExtractor)

getLinks(*self*)

Returns a map from link type to a list of links of that type that appeared in the page.

getParsedContent(*self*)

Returns the BeautifulSoup datastructure of the HTML of the site that was set using setSite .

getRawContent(*self*)

setSite(*self*, *stringUrl*, *content*)

Sets the current site url and content for the extractor.

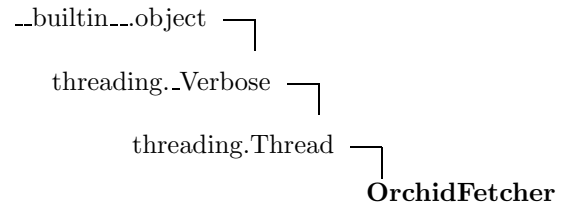
Parameters

`stringUrl`: The url of the site being analyzed.
`content`: The html content of the site.

2.5.2 Class Variables

Name	Description
LINK_PATTERNS	**Value:** `[('regular', <_sre.SRE_Pattern object at 0xb7a4-9bf0>, {'href': <_sre.SRE_Patt...` *(type=`list`)*

2.6 Class OrchidFetcher



This class is responsible for fetching url contents, processing them with UgrahExtractor and updating the site and link database.

2.6.1 Methods

__init__(*self*, *siteQueue*, *siteQueueCond*, *fetcherCondition*, *stopConditionLock*)

Creates a new fetcher thread (not started) with the following

Parameters

`siteQueue:`	the site queue from which the analyzer takes sites to analyze.
`siteQueueCond:`	A Condition object used to lock the siteQueue.
`fetcherCondition:`	a threading.Condition object which is used for communication between the fetcher and the controller: whenever a fetcher finishes working on it's assignment it calls fetcherCondition.wait() and waits until the controller assigns a new url for it to fetch.
`stopConditionLock:`	a threading.Lock object which is used to lock the internal stop condition variable. A thread that wishes to change this variable should lock it first.

Overrides: threading.Thread.__init__

getUrlsCounter(*self*)

Returns the number of URLs this fetcher has handled. Should be called only AFTER the thread is dead.

isFree(*self*)

Returns True if the fetcher hasn't been assigned a URL yet.

run(*self*)

Performs the main function of the fetcher which is to fetch the contents of the url specified by setCurrentStringUrl. This method loops until the stop condition is set.

Overrides: threading.Thread.run

setStopCondition(*self*, *val*)

Can receive either True or False. Set to Ture when the fetcher should stop working. WARNING: It's *necessary* to acquire the lock which was passed to the constructor as stopConditionLock before calling this method.

setUrl(*self*, *stringUrl*)

Sets the url that the fetcher should work on. It's *necessary* to acquire the condition instance which was passed to the constructor as fetcherCondition before calling this method and call notify afterwards

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__, __str__
Inherited from Thread: __repr__, getName, isAlive, isDaemon, join, setDaemon, setName, start

2.7 Class Site

A class for representing the information that is collected for a specific site.

2.7.1 Methods

__init__(*self*, *stringUrl*, *links*, *parsedContent*, *rawContent*)

Creates a new site.

Parameters

`stringUrl:`	the url of the site
`links:`	a map from link types to links which appear on this page
`parsedContent:`	BeautifulSoup instance which contains the parsed content of the page.
`rawContent:`	The raw content of the page as a string.

2.8 Class UrlHandler

A class responsible for parsing a url and retrieving it's contents.

2.8.1 Methods

__init__(*self*)

A constructor for the url handler. Should be followed by calls to setCurrentUrl and getSite.

getSite(*self*)

Returns the url object which was opened by setCurrentUrl. The returned object acts just like a file object.

processUrl(*self*, *stringUrl*)

Sets the url that the parser is working on. Raises an exception if we can't open it.

Index