

|  |  |  |
| --- | --- | --- |
| MSc Advanced Software Engineering  Research Project | | |
|  | | |
| Assignment Type: | Software Design Document | |
| Submission Date: | 03/15/2017 | |
|  | | |
| Student Id | | Student Name |
| 2015101 | | K.A.S.C.Kathriarachchi |

|  |
| --- |
|  |
| Software Design Document |
| Blocking brute force attacks using binomial ladder filter in .net applications |

Student Id - 2015101

Name - K A S C Kathriarachchi

Supervisor - Mr Cassim Farook

**Introduction**

The purpose of this software design document is to provide a low-level description of the brute force attack blocking application, providing insight into the structure and design of each component. Topics covered include the following:

1. Class hierarchies and interactions
2. Data design
3. Design constraints and restrictions
4. Restrictions, Limitations, and Constraints.

**Purpose**

The aim of this document is to specify complete description of the library package for blocking brute force attacks using binomial ladder filter in .net applications. It is basis for requirement that I gathered from research. Through this document, the workload needed for development, validation and verification will ease. To be specific, this document is going to describe functionality, performance, attributes and the design constraints of the system which is going to be developed.

**Scope**

Special design to identify brute force attacks and block those from accessing the application. This will not effect to original users and they can work without locking or entering captcha to their accounts.

**Overview**

We are going to focus on describing the system in terms of product perspective, product functions, user characteristics, assumptions and dependencies on the following section of this document. Next, we will address specific requirements of the system, which will enclose external interface requirements, requirements of the system and other requirements.

**Overall Description**

This section gives background information about specific requirements of the library package for blocking brute force attacks using binomial ladder filter in .net application project to be developed in brief. Although we will not describe every requirement in detail, this section will describe the factors that affect the final product.

**Product Perspective**

This software product is eventually intended for the .net web application developers. Product will be deployed as library package and can be accessed by directly and through nugget package manager. However, this library package will be only a part of a large .net applications.

To use this you have to have .net environment 4.0 or higher. This will be deployed as free package to download for everyone. This will be coming as version with bug fixes and latest improvements.

From the user point of view, user will have to use integrated function to authenticate and unauthenticated. For that user needs to put inputs that required for plugin and those are mentioned below

All of the data processing and mining work will be done inside of the package and those criteria are not going to publish. And those criteria’s may be change with versioning of the package in future.

**Product Functions**

This new product, block brute force attacks using binomial ladder filter in .net application, having number of features which will allow .net web application developers to use functionalities which have been explained above. Required functionalities of the product can be summarized in two categories; Identifying brute force attacks and blocking brute force attacks. Overall description of the requirements can be found below;

**Identifying brute force attacks**

To identify brute force attacks system need inputs those are mentioned in functional inputs section. This identifying happens in series of calculation and mining work.

**Blocking brute force attacks**

If pervious category identifies this request as brute force attack then plugin needs to block this request and future request from that requester.

**SOFTWARE CONTEXT**

Brute force attack blocking application will be offered through the nugget package manager for free of charge. Future development plans will be based on the features (if any) that do not make it in the initial release of the application. If all of these features are included, there are several experimental features that will potentially be incorporated. These features are not covered in this document.

**INTENDED AUDIENCE AND READING SUGGESTIONS**

While the software requirement specification (SRS) document is written for a more general audience, this document is intended for individuals directly involved in the development of Brute force attack blocking application. This includes software developers, project consultants, and team managers. This document need not be read sequentially; users are encouraged to jump to any section they find relevant. Below is a brief overview of each part of the document.

* Part 1 (Architectural and Component-Level Design With class diagram and Decision tree)

This section describes the Brute force attack blocking application system class by class, class hierarchies, performance/design constraints, process details, and algorithmic models.

* Part 2 (Data Design)

Readers interested in how Brute force attack blocking application organizes and handles data should consult this section, which covers data structures and flow patterns utilized by the system.

* Part 3 (Restrictions, Limitations, and Constraints)

This section discusses the general constraints imposed upon the project

* Part 4 (Testing)

Readers interested in the software testing process should consult this section, which offers a list of test cases, expected responses, and other pertinent information.

**ARCHITECTURAL & COMPONENT-LEVEL DESIGN WITH CLASS DIAGRAM AND DECITION TREE**

**SYSTEM STRUCTURE**

The Brute force attack blocking application is a server-side .Net library package and stored the data in the memory. This can be plugged into any .Net base authentication platform from server side. Brute force attack blocking application monitors the authentication process and mining the request as fake or original. There’s no any graphical component in the application it’s just a pluggable to any other .net back-ends.

**LoginAttempt Class**

The LoginAttempt class is meant to represent Brute force attack blocking application header component class for its data structure. With this information it takes all the filtering work.

* UsernameOrAccountId

A unique ID that identifies the account that the client was attempting to login to.

* AddressOfClientInitiatingRequest

The IP address of the client that was attempting to login to. This is

* EncryptedIncorrectPassword

When a login attempt is sent with an incorrect password, that incorrect password is encrypted and saved. .

* Phase2HashOfIncorrectPassword

The phase2 hash encrypted the incorrect password, which is available for future analysis. We can use this to determine if a client is sending an incorrect password that was recently attempted for this account. This allows us to detect a common-benign behavior: an automated client that is trying the same incorrect password over and over again.

* TimeOfAttemptUtc

The time of the attempt.

* Api

The API/protocol over which the attempt was sent. Can be used to differentiate attempts via web browsers (which support cookies, javascript, CAPTCHAS, and other goodness) from less flexible clients.

* HashOfCookieProvidedByBrowser

The hash of a client-identifying cookie provided by the client/browser. To allow flexibility in the hash function and save the user of this class from having to choose a hash function, the hash is automatically performed by using the setter CookieProvidedByClient.

* DeviceCookieHadPriorSuccessfulLoginForThisAccount

Will be set to true if, when the login attempt is being processed, it is determined that the client that initiated the login attempt provided a cookie that was also provided during a previous successful login attempt.

* Outcome

The outcome of the LoginAttempt, which default to Undetermined and will be set during analysis. This value does need not be set by the creator of the LoginAttempt record.

* PasswordsHeightOnBinomialLadder

The popularity of the password provided among the set of past failed guesses observed by the system.

**IpHistory Class**

This store the history of the ip addresses that use to login and it also scores the points for each ip address

* Address

The IP address being tracked.

* RecentPotentialTypos

A set of recent login attempts that have failed due to incorrect passwords that are kept around so that, when the correct password is provided, we can see if those passwords were typos and adjust the block score to reduce past penalities given to failed logins that were typos.

* CurrentBlockScore

The current block score for this IP, in the form of a number that decays with time.

**RemoteHost Class**

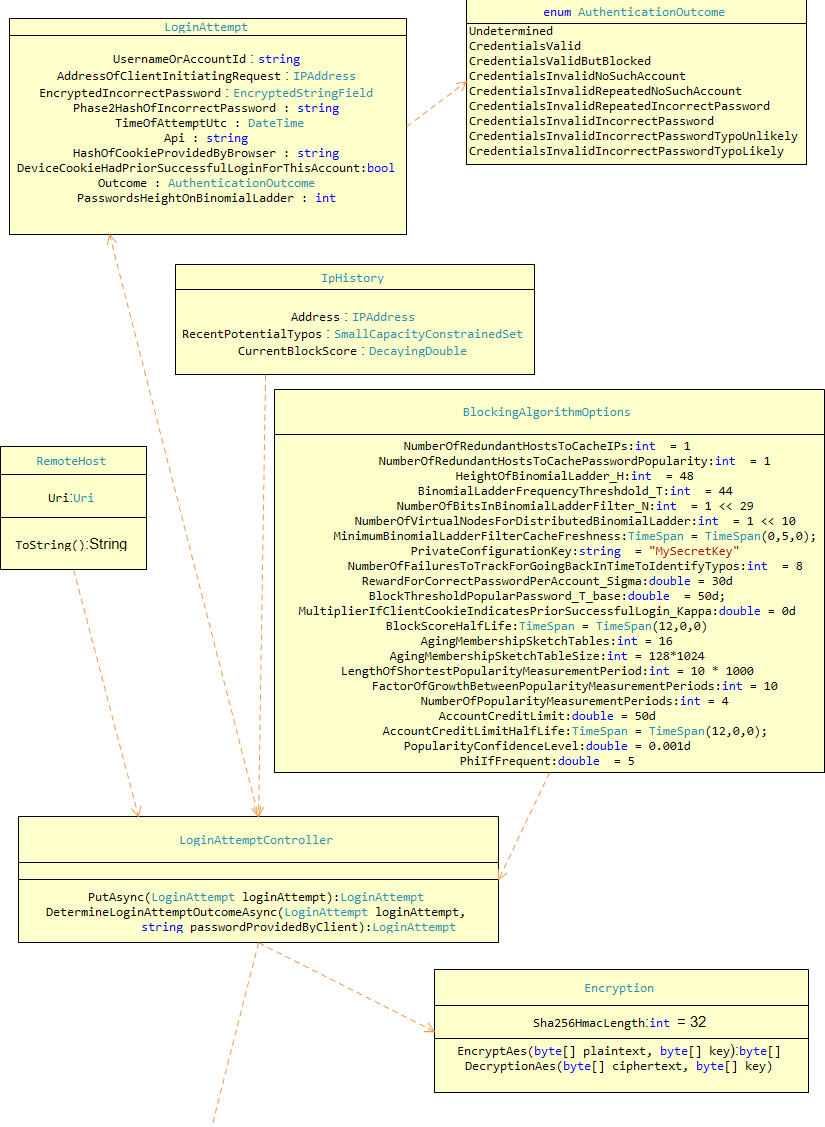
This class identifies remote hosts to be used as part of a distributed system for

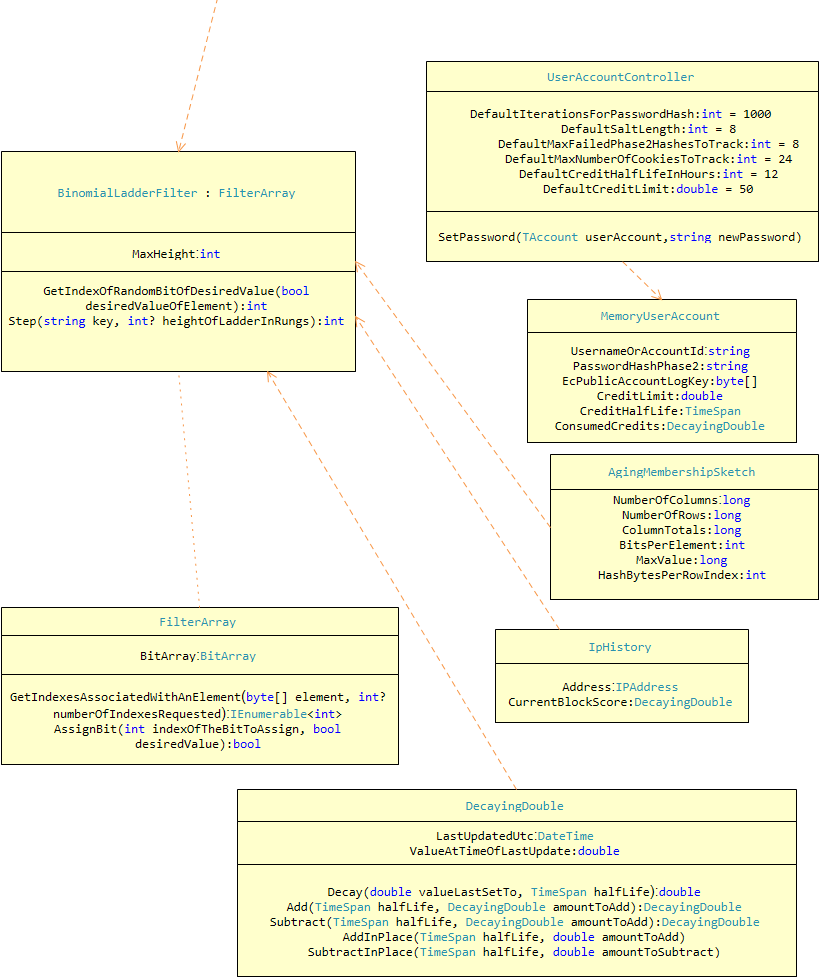
Balancing the load of login requests (and storing data associated with past requests) across systems.

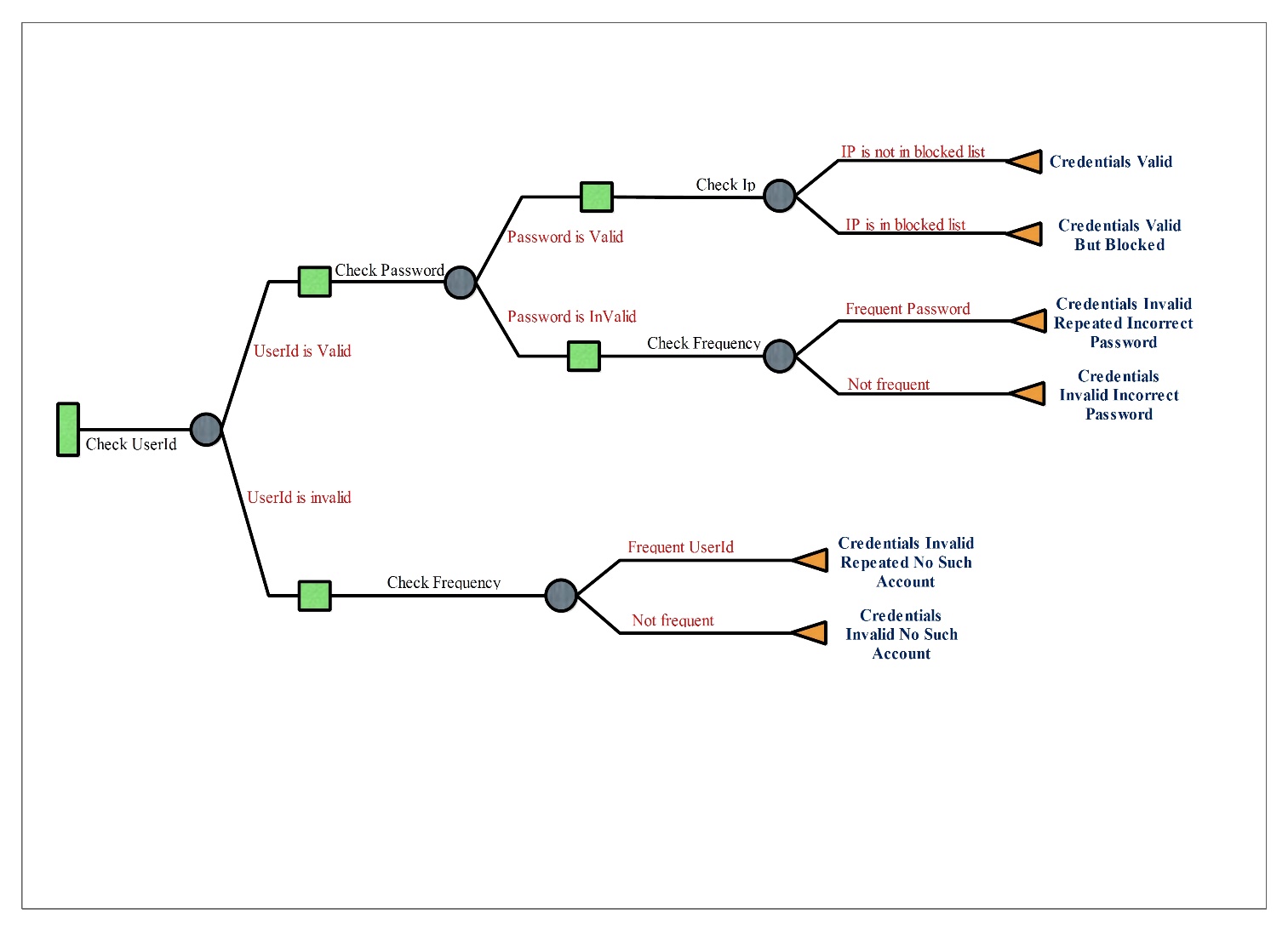
* URI

This store the remote host request path

**Class Diagram**

****

****

**Decision Tree**

**Data Design**

Data will reside locally in memory and will be organized based on the classes defined in this document. The classes that handle the data will be isolated and will be accessed by way of a access system. The data objects created on the local device will only exist for the duration of time that the application is running, and will subsequently be destroyed.

**RESTRICTIONS, LIMITATIONS, AND CONSTRAINTS**

As time is a limiting factor, Implementation of this application package will limited to support of web applications and can be integrated with .net authentication libraries.

Another limitation of the software is the lack of an analytics generation from the library package. Currently this package will identify and block the suspicious activities and not giving any analytics output to administrators

A constraint that is frequently mentioned in this document as well as the SRS is the is for blocking bruit force attacks and use with existing authentication methodology. Currently the data is hold as in memory data and if there is future requirement, we can go for sql db data storage

As the application frequently block the suspicious attacks and make memory, this will increase temporary memory balance and we may want good temporary memory (RAM) server. Currently, we do not anticipate this being a problem, because this reduces temporary memory by blocking unwanted attacks and 10% of that memory is enough to handle that. However, if we going to use sql db in future this also can avoid as constrains, but this is super-fast than sql.

Another constraint imposed on the user of this plugin has to have username and password base login system and this will not support to other social base login methodology. This is used a unique identifier for each user and identify login is valid or not.

**TESTING**

Each component will be tested individually to make sure the functions and constructors are operating correctly. Then, once the program is assembled, it will be tested as a whole to ensure all of the components work together correctly.

The types of tests to be conducted are specified below, including as much detail as is possible at this stage. Emphasis here is on black-box and white-box testing.

**WHITE BOX TESTING:**

While each class is being implemented, the developer assigned to that class will test to make sure each function is working. The developer is fully responsible for debugging his/her own code because the overhead of sharing code between developers has been deemed too costly. However, all code will be accessible through the provided version control system, so this rule may be violated if needed.

**BLACK BOX TESTING:**

Black Box Testing comprises a majority of the testing process. This will be done after all the components are assembled, and will consist of running through all possible situations that may occur in the use of the application.

**PERFORMANCE BOUNDS**

Due to fact that the application is very demanding with respect to resources, execution time for all actions should be negligible. Also, since data is filtering into a methodology and save those for future mining task, that also take small timing. In relation to the server component, it must uphold acceptable performance ability when negotiating the passing of information between authentication process and filtering. The only expected issue involving the server (in terms of performance) involves the amount of temporary memory allocation of the server. This would be easily throttled by upgrading the temporary memory of the server. However, this will not be an issue for this project, as long as this reduce unwanted memory used by attackers.