

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

|  |
| --- |
|  |
| Software Requirement Specification |
| 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**

This document is a Software Requirement Specification (SRS) for the library package for blocking brute force attacks using binomial ladder filter in .net application. This is the initial draft for the SRS and it will be used for the extensions.

This is a library package which provides facilities to web based .net applications to protect from brute force attacks. As a library package this can plug into any .net web based application and improve the security of the application.

The purpose of this project is provide an easy-to-use protection library package that will save programmers time.

**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.

**Research Problem**

Password-guessing attacks are considered to be the simplest to execute by an attacker. It doesn’t require a vulnerability or an exploit, and can be performed by any bad actor. The underlying problem, however, is a bit more complicated.

First, people tend to be really bad at picking strong passwords and the majority don’t use a password manager.

Second, brute force attacks can have other unwanted side effects, such as causing website availability issues due to denial-of-service errors. The repeated attempts can create unnecessary load on the web server’s local resources by passing too many requests at a time. As an active defender, they can also be problematic due to the noise that comes from all the logs. This makes it hard to manage and spot other attacks among the brute-force attempts.

Their few methods available to prevent this, but none of are not user friendly. My research is going to make this protection as very user friendly and easily attachable to other .net applications. For that I am going to use binomial ladder filters and going to do internal mining work and calculation work to understand request is type of attack or not. If request is attack, I am going to block that without harming to the application.

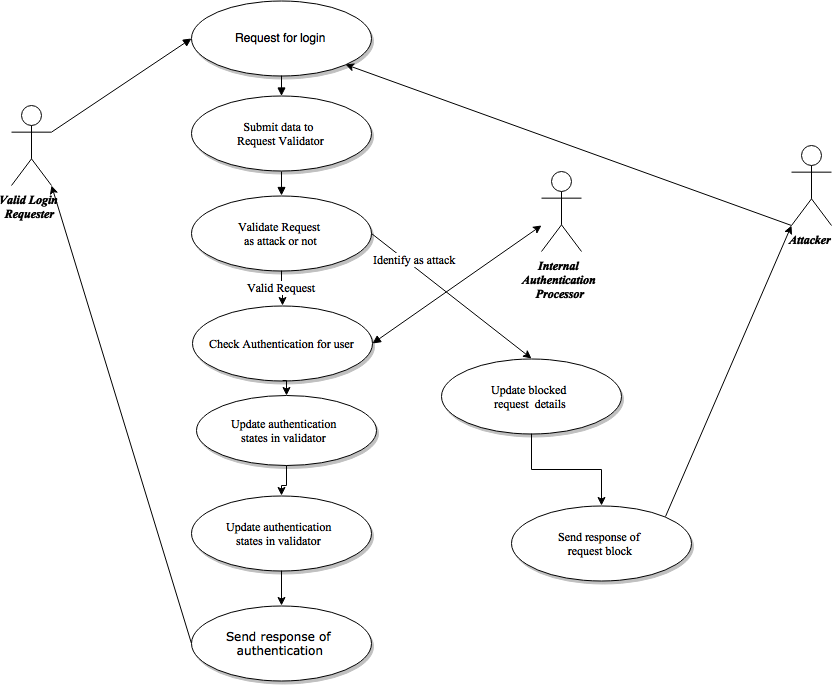
**Scope**

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.

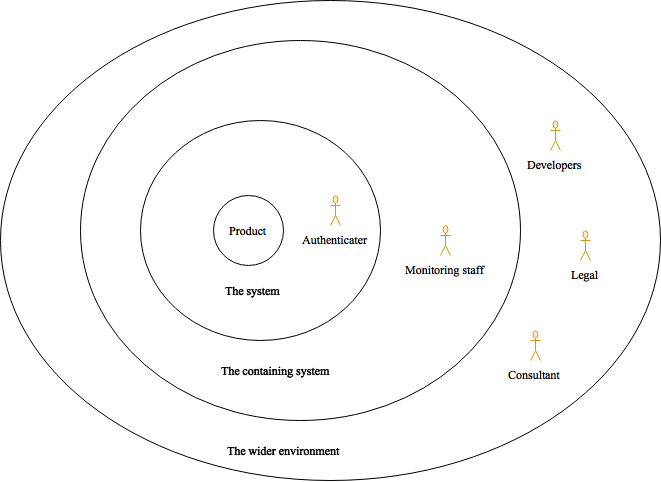
**Overview**

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.

**Use case Diagram**

****

**Onion Model**

****

**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.

# **Functional Inputs**

## **Unique Id That identifies the account**

This will be a email or account id, System needs to identify account from unique identity. For that users can use email of account id for this input. This is mandatory input

1. **IP address of the client that was attempting to login**

To identify where we receive login attempt we need to have ip address of the login attempt receiving place. To identify user frequent of login place we need this. From this we can identify how many logins going on one place.

1. **Time of the attempt**

This store the time of the login attempt. System needs this to check the frequent of login attempts.

1. **API/Protocol over which the attempt was sent**

To identify from where this login attempt come we need API/Protocol. This enables to identify login attempt from what type of environment

1. **Outcome of the login attempt**

This store the outcome of the login attempt, that means attempt was success or failed. System needs this for future process if login attempt get failed or success. If specially it failed needs to save failed login information for future process

1. **Incorrect password**

This store the password that failed on the login. System needs this to compare passwords that trying to login to the system.

**Functional Requirements**

* **Input Requirement**

The system shall provide an input interface to input request parameters before authenticating

1. Unique Id can be any type, because it will be identification for that account and its mandatory
2. IP address accept type of string and its mandatory field
3. Time of attempt will accept date time format data in its mandatory.
4. Api/protocol input accept data type string and its not a mandatory field
5. Outcome of the login type will by type of enum and it contain login attempt success or failed
6. Incorrect password is data type of string and its not a mandatory field

The system shall provide and input after the authenticating, that input contain outcome of the login and incorrect password. Incorrect password is entered only if login attempt is failed

* **Output Requirement**

The system shall provide an output of the request parameters before authenticating.

1. System should provide output type of enum and that will be attack or not.
2. If type identify as attack system has to send the request block call
3. If identified as not attack system has to provide next input after the authentication

**Non-functional Requirements**

In this section, last group of the requirements which is non-functional requirements will be explained in detail. Non-functional requirements include performance requirements.

* **Performance Requirements**

1. Since this software is going service to the web based applications, it must response fast, for that all the functionalities must be optimized.
2. Final output must be light weight and it must run in all the platforms within the framework area
3. This package must be work in a multiuser environment