|  |  |  |
| --- | --- | --- |
| **产品名称** | | **密级** |
| **\*\*\*\*软件设计方案** | | **公开** |
| **拟制人：（签字）** | |  |
| **日期** | **2022.7.28** | **共12页** |

**\*\*\*\*软件设计方案**

**大连理工大学**

目 录

[1．引言 4](#_Toc1339492931)

[1.1编写目的 4](#_Toc786619816)

[1.2项目背景 4](#_Toc809916580)

[1.3定义 4](#_Toc1516605374)

[1.4参考资料 4](#_Toc1103114575)

[2. 需求分析 5](#_Toc820337474)

[2.1 任务概述 5](#_Toc566911778)

[2.1.1目标 5](#_Toc1848794754)

[2.1.2运行环境 5](#_Toc752542035)

[2.1.3条件与限制 5](#_Toc1442785062)

[2.2 数据描述 5](#_Toc1650678757)

[2.2.1静态数据 5](#_Toc1764116953)

[2.2.2动态数据 5](#_Toc1354398589)

[2.2.3数据库介绍 5](#_Toc50427123)

[2.2.4数据词典 6](#_Toc1420099343)

[2.2.5数据采集 6](#_Toc476405043)

[2.3功能需求 6](#_Toc1120521685)

[2.4 性能需求 6](#_Toc1323859252)

[2.4.1数据精确度 6](#_Toc24381797)

[2.4.2时间特性 6](#_Toc1762969249)

[2.4.3适应性 6](#_Toc1392290284)

[2.5运行需求 6](#_Toc1240985476)

[2.5.1用户界面 6](#_Toc881715468)

[2.5.2硬件接口 7](#_Toc1354706376)

[2.5.3软件接口 7](#_Toc928435938)

[2.5.4故障处理 7](#_Toc606630864)

[2.6 其它需求 7](#_Toc1540058939)

[3. 概要设计 8](#_Toc150190482)

[3.1 总体设计 8](#_Toc958145749)

[3.2 接口设计 8](#_Toc1723218237)

[3.2.1外部接口 8](#_Toc1164445817)

[3.2.2内部接口 8](#_Toc822371208)

[3.3 数据结构设计 9](#_Toc388140764)

[3.4 运行设计 9](#_Toc1573984609)

[3.5 出错处理设计 9](#_Toc1255759717)

[3.6 安全保密设计 9](#_Toc84280903)

[3.7 维护设计 9](#_Toc1317413348)

[4．程序描述 10](#_Toc1209739266)

[4.1功能 10](#_Toc1860157513)

[4.2性能 10](#_Toc600387965)

[4.3输入项目 10](#_Toc1842354149)

[4.4输出项目 10](#_Toc2026959797)

[4.5算法 10](#_Toc1580215818)

[4.6程序逻辑 10](#_Toc756990677)

[4.7接口 10](#_Toc1049183511)

[4.8存储分配 10](#_Toc639043860)

[4.9限制条件 10](#_Toc844436373)

[4.10测试要点 10](#_Toc1870181635)

# 1．问题定义

## 1.1项目背景

表达式语言（Expression Language），又称EL表达式，是一种在JSP中内置的语言，可以作用于用户访问页面的上下文以及不同作用域的对象，取得对象属性值或者执行简单的运算和判断操作。OGNL全称Object-Graph Navigation Language即对象导航图语言，一种功能强大的表达式语言。OGNL可以存取对象的任意属性并且调用对象的方法，还可以遍历整个对象的结构图，实现字段类型转化。webwork2 和 Struts2.x 中使用 OGNL 代替原来的 EL 来做界面数据绑定（就是把textfield.hidden和对象层某个类的某个属性绑定在一起，修改和现实自动同步）Struts2框架因为滥用OGNL表达式，所以漏洞较多。

## 1.2问题描述

OGNL具有三要素: 表达式、ROOT对象、上下文环境（MAP结构）。处理OGNL的最顶层对象是一个Map对象，通常称这个Map对象为context map或者context，OGNL的root就在这个context map中，在表达式中可以直接引用root对象的属性。OGNL是一种功能强大。的语言，但在Apache Struts中将用户提供的输入视为OGNL会影响安全性。

# 需求分析

## 2.1 用户需求分析

用户输入检测语句，注入语句检测模型返回语句是否可能构成注入，并对注入语句进行风险评级。

## 2.2 功能需求分析

#### 2.2.1功能划分

#### 2.2.2功能描述

## 2.3 非功能需求

#### 2.3.1数据精确度

#### 2.3.2性能需求

【如响应时间、更新处理时间、数据转换与传输时间、运行时间等。】

## 2.4运行需求

#### 2.4.1用户接口

命令 输入输出，UI等

# 概要设计

## 3.1 总体设计

#### 3.1.1处理流程

流程图如下图3-1

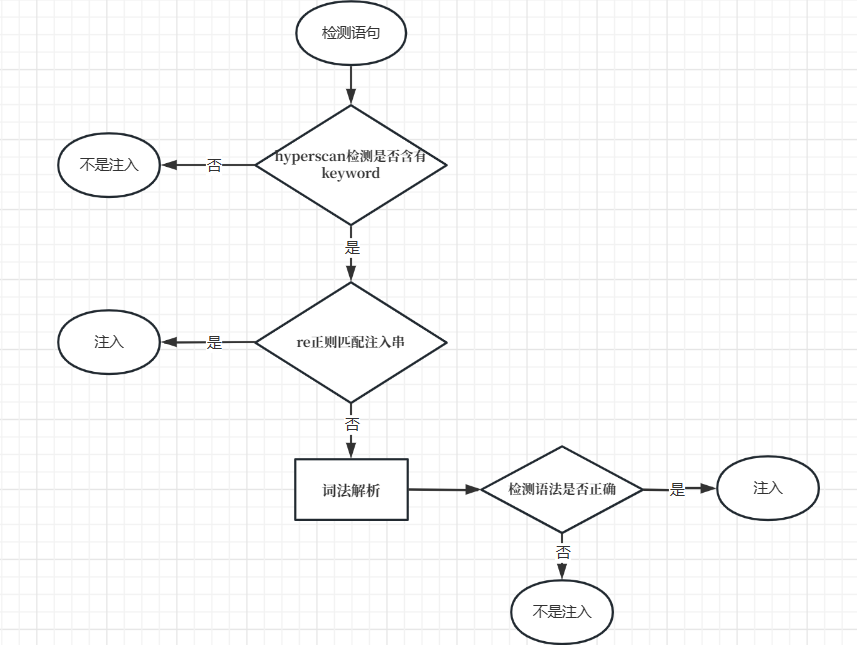


图3-1

#### 3.1.2总体结构和模块外部设计

#### 3.1.3功能分配

Hyperscan：Hyperscan是一种高性能、多模式的正则表达式匹配引擎，能够在非常短的时间内快速地匹配输入数据中的多个正则表达式。在模型的第一阶段运用Hyperscan进行白噪音筛选，尽可能的筛选掉不可能构成注入的数据，避免大量的白噪音数据进入模型导致模型运行效率降低，降低白噪音对检测结果的影响。

Re正则匹配：我们将常见的注入语句中，影响注入权重较高的部分，提取出来作为注入成分库。在模型的第二阶段使用Re正则匹配，如果我们的检测语句与注入成分库中的注入片段成功匹配，确认检测语句可能完成注入或有注入倾向，判定该语句有注入风险，为注入语句。

词法分析：将输入语句转换为token流，token流用于语法解析。

语法分析：将词法分析阶段产生的token流，基于我们定义的语法文件进行语法分析。

## **3.2 接口设计**

#### 3.2.1外部接口

【包括用户界面、软件接口与硬件接口。】

#### 3.2.2内部接口

【模块之间的接口。】

## 3.3 数据结构设计

#### 3.3.1逻辑结构设计

#### 3.3.2物理结构设计

#### 3.3.3数据结构与程序的关系

## 3.4 运行设计

#### 3.4.1运行模块的组合

#### 3.4.2运行控制

## 3.5 异常出错处理设计

#### 3.5.1出错输出信息

#### 3.5.2出错处理对策

【如设置后备、性能降级、恢复及再启动等。】

# 问题汇总

## 4.1 特殊用例

#### 4.1.1 长亭网页版demo检测不稳定的语句

有时候是nomal 有时候是high

%{@java.lang.Class}

%{@java.lang.ClassLoader}

#### 4.1.2 ognl.OgnlContext相关检测

%{@ognl.OgnlContext@DEFAULT\_MEMBER\_ACCESS, @a@a()} high

%{@ognl.OgnlContext} high

%{#a=@ognl.OgnlContext@DEFAULT\_MEMBER\_ACCESS,@a@a()} normal

%{#\_memberAccess=@ognl.OgnlContext,@sun.misc.Unsafe@getUnsafe()} normal

#### 4.1.3 ProcessBuilder相关检测

%{@java.lang.ProcessBuilder@command(List<String> command)} high

%{@a()@java.lang.ProcessBuilder} high

%{@ProcessBuilder@a()} high

%{@ProcessBuilder@a} high

%{@a@ProcessBuilder} high

%{@a()@ProcessBuilder} normal

## 4.2 数据统计

#### 4.2.1 注入相关的java类

java.lang.ProcessBuilder

Java.lang.RuntimePermission

Java.lang.ClassLoader

Java.lang.System

java.lang.ThreadGroup

Java.lang.Compiler

java.lang.Thread

java.lang.Class

java.lang.ThreadLocal

java.lang.Package

java.lang.Runtime

java.lang.SecurityManager

Java.lang.Object

java.lang.InheritableThreadLocal

java.lang.Shutdown

java.lang.Process

sun.misc.Unsafe

Java.io.File

java.io.BufferedReader

java.io.DataInputStream

javax.script.ScriptEngineManage

javax.persistence.EntityManager

javax.servlet.ServletContext

Java.io.FileDescriptor

Java.io.FileInputStream

Java.io.FileOutputStream

Java.io.FilePermission

Java.io.FileReader

Java.io.FileWriter

java.util.Collections

java.util.Scanner

#### 4.2.2 注入成分库

java.lang.ProcessBuilder

Java.lang.Runtime

Java.lang.ClassLoader

Java.lang.System

javax.script.ScriptEngineManager

java.lang.Shutdown

java.lang.Class

Java.io.FileDescriptor

Java.io.FileInputStream

Java.io.FileOutputStream

Java.io.FilePermission

Java.io.FileReader

Java.io.FileWriter

Java.io.File

ognl.OgnlContext

DEFAULT\_MEMBER\_ACCESS

# 5.附录

【列出文当中所用到的专门术语的定义和缩写词的原文。】

## 5.1术语及缩略语

Root对象:OGNL的Root对象可以理解为OGNL的操作对象. 当指定了一个表达式的时候, 需要指定这个表达式针对的是哪个具体的对象. 而这个具体的对象就是Root对象, 这就意味着, 如果有一个OGNL表达式, 则需要针对Root对象来进行OGNL表达式的计算并且返回结果.

上下文环境:在OGNL的内部, 所有的操作都会在一个特定的数据环境中运行. 这个数据环境就是上下文环境(Context).OGNL的上下文环境是一个Map结构, 称之为OgnlContext.Root对象也会被添加到上下文环境当中去, 简而言之, 上下文就是一个MAP结构, 它实现了java.utils.Map的接口.

OGNL:全称Object-Graph Navigation Language即对象导航图语言，一种功能强大的表达式语言.

## 5.2参考资料

1. RFC等标准文档
2. 技术白皮书等