鸿渐科技

鸿渐 SAST 静态代码分析工具 V5.0

用户使用手册

北京鸿渐科技有限公司

目录

1.1. 购买工具简要说明

用户在华为云商店购买产品后，需联系本公司相关人员索要相关软件工具安装包，同时会有相关技术人员远程支持工具部署，部署成功后，下载许可证界面 sn 文件并发送给我司技术人员开通 license（一个 sn 文件对应一个 license），用户上传 license 后方可使用工具。

1.2. 概述

鸿渐静态代码分析工具(简称鸿渐 SAST)是基于北大和中科院等高校优秀毕业生多年的研究成果、应用多种国际先进的代码分析技术，自主研发的源代码检测工具。在不改变组织现有研发流程的前提下，与源代码管理系统(Git、SVN 等)、漏洞管理系统(如 Jira、Bugzilla、禅道等)、持续集成工具(如 Jenkins、禅道)等无缝对接，将源代码检测流程融入企业的研发流程，实现源代码编码规则检测、运行时缺陷检测、安全漏洞检测、度量统计、编译不通过检测、逆向架构图自动生成等功能，并提供检测器自主研发接口，帮助组织快速构建源代码安全自主检测体系和能力。

1.3. 工具特点

SAST 静态代码分析工具相比于其他国外静态分析工具的主要特点如下：

① 自主研发，技术自主可控：研发团队专注于软件安全、供应链安全和漏洞自动挖掘，拥有数十项发明专利和数十篇国际顶级会议论文。

② 更多的缺陷检测类型和精度保障：支持更多的运行时缺陷的检测，相比于国外领先工具具有相似的误报率(15%左右)及更低的漏报率(相对漏报率约 10%，能够发现其他国外工具不能发现的或遗漏的缺陷)。

③ 详细的数据仪表及人员绩效分析：工具提供检测任务、项目组、缺陷、人

员等维度的数据分析仪表，例如：项目缺陷分布、重要缺陷密度、缺陷类型及等级分布、缺陷趋势、人员绩效数据等，帮助管理者快速决策关键应用及人员管理的规划。

④ 强大的语法词法分析：支持在代码编译不通过的情况下进行质量缺陷/安全漏洞检测、分析，并提示编译缺少的文件，但仍不影响整体检测，最大限度降低测试过程成本，让用户专注于漏洞和缺陷。

⑤ 精准的跟踪编译机制：SAST 静态代码分析工具采用跟踪编译形式，通过对代码进行跟踪编译，再对编译后的结果进行检测，使缺陷检测结果更加准确，误漏报更低。

⑥ 反向架构图一键查看：产品提供多种反向代码架构图，例如：函数调用图/被调用图、文件函数调用图、项目函数调用图、值跟踪图，帮助用户在脱离 IDE 的情况下准确跟踪代码上下文，确认漏洞。

⑦ 多线程检测：由于 SAST 静态代码分析工具采用 B/S 部署模式，支持多用户可同时进行检测，检测后将缺陷结果在服务器集中查看。

⑧ 更完善的缺陷确认机制：在缺陷检测完毕后，能够快速定位缺陷发生的行数，并且 SAST 静态代码分析工具提供变量的定义处、函数的定义处以及函数被调用点的自动定位等功能，并以统计图方式显示各等级缺陷。

⑨ 强大的适应性及快速大规模检测能力：能分析上千万行代码，效率达 100 万行/小时以上，并支持复杂的编译环境以及多种嵌入式的开发环境。

⑩ 国产环境适配经验丰富：产品已经参与过多个国产化适配的项目，并且获得了麒麟软件的桌面操作系统及高级服务器操作系统的 NeoCertify 认证。

1.4. 检测能力

1.4.1. 安全漏洞

SAST 静态代码分析工具支持国际上通用的 OWASP 的百余种安全漏洞，包括但不限于以下缺陷类型：

未经验证的用户输入	弱加密算法	DNS 欺骗
命令行注入	LDAP 注入	SQL 注入
使用不安全的函数	管道劫持	字节序列错误
线程安全	缓冲区溢出	资源泄漏
密码权限	拒绝服务	隐私泄漏
内存污染	非法计算	安全编程函数

1.4.2. 运行时缺陷

SAST 静态代码分析工具支持国际上 CWE、CVE 的百余种缺陷，包括但不限于以下缺陷类型：

数组越界	变量未初始化引用	空指针解引用
整数溢出	错误的隐式转换	不可达代码
内存泄漏	内存重复释放	无效跳转与条件
无效跳转	循环跳出条件异常	字符串差一错误

不正确的指针判空	强制类型转换可能丢失精度	资源使用后未释放
缓冲区溢出	内存释放后返回	使用释放后的内存
非法计算	异常迭代器操作	互斥锁问题

1.4.3. 编码规则

SAST 静态代码分析工具支持国内、国际的千余种编码规则，包括但不限于以下缺陷类型：

类别	规则名称	数量
国内	GJB 5369 2005 C 语言安全子集	138
	GJB 8114 C/C++语言编程安全子集	204
国际	MISRA C 2004 英国汽车工业软件可靠性联会 C 语言标准	123
	MISRA C++ 2008 英国汽车工业软件可靠性联会 C++语言标准	112
	MISRA C++ 2012 英国汽车工业软件可靠性联会 C++语言标准	128
CERT	CERT 出版的最新 C 编码规则集，ISO-17961 标准	45
	CERT 出版的最新 Java 编码规则集	151
软件工程化	921 C-2007 载人航天 921 工程 C 语言安全子集	151
合计		**1052**

1.4.4. 框架和库

SAST 静态代码分析工具支持 80 多种 C/C++和 Java 等语言的框架和库，主要内容如下：

Hibernate	IBatis	MyBatis
Struts	Spring Framework	Spring Boot
Axis	JEE	Restlet
Apache Wicket	Apache Velocity	Apache Hadoop
JSF	javax.websocket	OWASP ESAPI
qt4	qt5	boost

1.4.5. 编译器

SAST 静态代码分析工具支持多种内置编译器和自定义编译器，主要内容如下：

GCC	Arm_5.1.6	ARMCC
ARMCC_504u2	c2000_6.2.7	c5500_4.4.1
c6000_7.4.8	ccs	gcc-arm-none-eabi-4_7-2013q3
MinGW	msp430_4.3.3	mssdks
terboc3	tornado	vs6
vs2010	vs08	其他自定义编译器

2. 安装部署

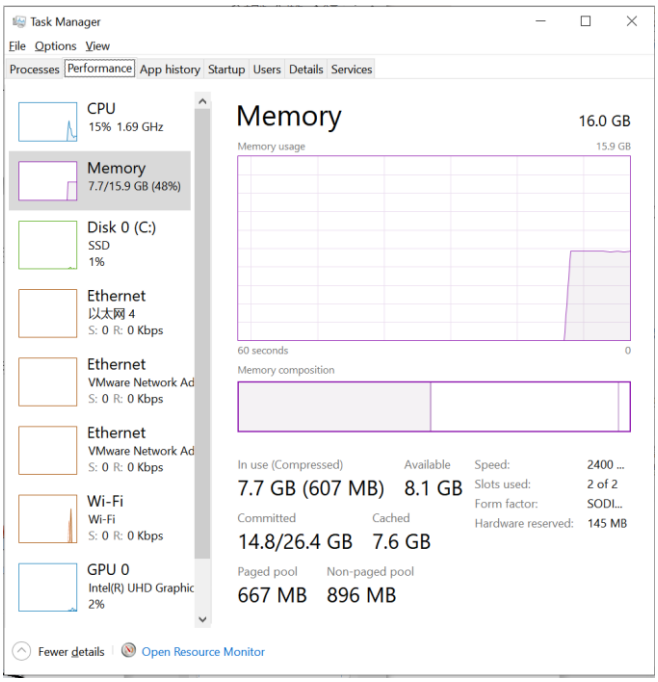
2.1. 系统环境要求

注：产品带狗，建议使用物理机安装产品。

代码规模 / 类型	10 万行以内	10 万行-50 万行	50 万行-100 万行	100 万行-1000 万行
建议物理可用内存	8G-16G 或以上	16G-32G 或以上	32G-64G 或以上	64G-128G 或以上
预计占用内存	4G-8G 或以上	8G-16G 或以上	16G-32G 或以上	32G-100G 或以上
操作系统	Windows 系列/LINUX 系列（Ubuntu、中标麒麟、RedHat）/UNIX（Solaris）			
CPU	Intel 系列/AMD 系列/龙芯系列：其中 Intel/AMD I52.5GHz 或以上，龙芯 0.8GHz-1.0GHz			
硬盘容量	100G 以上（视用户的检测代码量而定，建议预留 50G 可用空间）			

注：以上内存要求是机器的物理可用内存，可在操作系统中查看系统可用物理内存。

① windows 平台下物理可用内存查看：可以在资源管理器中查看，如图所示：



② LINUX 平台下物理可用内存查看：通过命令【free -n】查看可用内存，如图所示：

```
For more details see free(1).
defect@defect-VirtualBox:~$ free
              total        used        free      shared  buff/cache   available
Mem:        7636988     1039392     3067596       22556     3530000     6273200
Swap:       2097148           0     2097148
defect@defect-VirtualBox:~$
```

2.2. Linux 系统环境下安装

此章节介绍 LINUX 下 SAST 静态代码分析工具安装过程，包含客户端、服务端，安装环境为 Ubuntu 18.04 中文版 32 位操作系统，32G 内存。

说明：SAST 静态代码分析工具安装需要临时获取 root 管理员用户权限，请安装用户联系管理员获取临时权限，以下操作假设安装用户已经获取 root 权限。

产品安装步骤：

1) 鼠标右击 SAST 静态代码分析工具安装文件，将其解压。进入解压后的【setup】文件夹，在窗口上右键【在终端打开(T)】，打开终端窗口，运行【chmod 777 -R *】，获取安装目录执行权限

2) 在终端窗口，运行【./startconfig-linux32.sh】，运行安装程序：

① startconfig-linux32.sh 为 LINUX 32 位操作系统安装文件

② startconfig-linux64.sh 为 LINUX 64 位操作系统安装文件

③ startconfig-solaris.sh 为 Solaris UNIX 32 位操作系统安装文件

④ startconfig-solaris64.sh 为 Solaris UNIX 64 位操作系统安装文件

⑤ 安装帮助.txt，为安装帮助、编译分析项目助手文档

⑥ 将激活码 sn.txt 放置到安装目录下即可完成安装

2.3. Windows 系统环境下安装

此章节介绍 Windows 下 SAST 静态代码分析工具安装过程，包含客户端、服务端，安装环境为 Win7 64 位操作系统，32G 内存。

2.3.1. 安装 JDK8

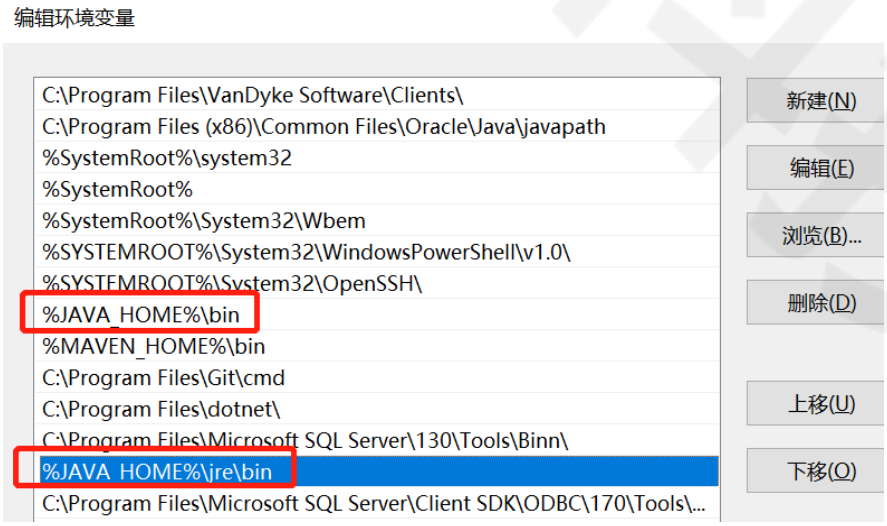
在“工具安装包”中找到 jdk-8u152-windows-x64.exe 并运行，按照默认配置安装即可。注：此配置是按照将 jdk8 设置为计算机全局设置来操作运行的，如果电脑中包含其他版本 jdk，为防止 jdk 冲突，可向相应技术人员获取运行时 jdk 的配置方式。

安装完成后配置环境变量：

① 我的电脑右键->属性->高级系统设置->环境变量->系统变量

② 新建：变量名：JAVA_HOME，变量名：C:\Program Files\Java\jdk1.8.0_261（按照实际 jdk 安装目录为准）

③ 编辑 Path：增加;%JAVA_HOME%\bin; %JAVA_HOME%\jre\bin



④ 保存后，打开 cnd 窗口，输入：java –version，出现版本信息，说明配置成功（部分系统可能需要重启生效）

```
C:\Users\l   i>java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
```

2.3.2. 工具部署启动

工具部署启动的准备工作：

① 创建项目主目录，随意位置，注意路径中不要包含中文和空格。

② 标准示例：D:\sast

③ 安装谷歌浏览器，本系统与谷歌浏览器兼容的最好。

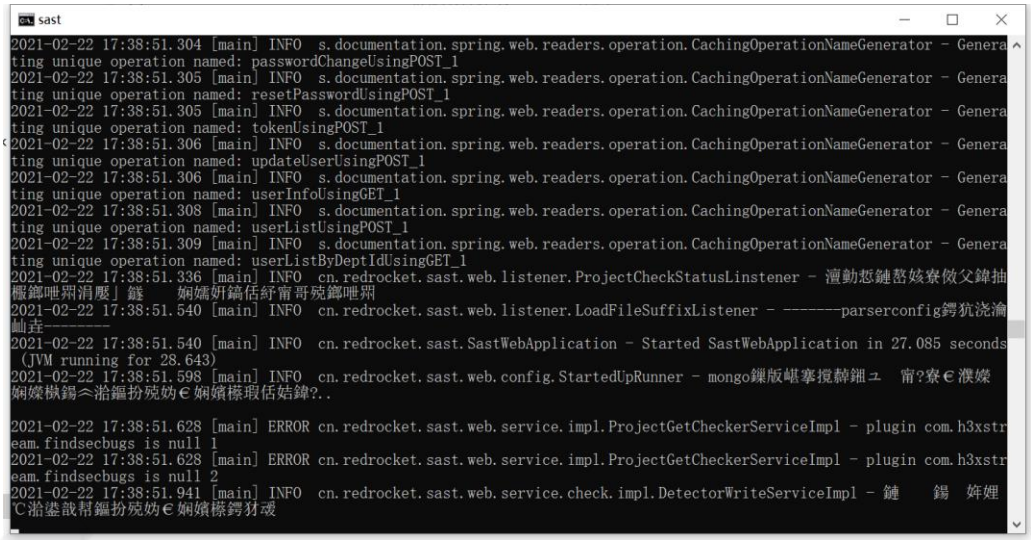
2.3.3. Server 配置变更

拷贝 server 到主目录下，进入 server，修改 application.yml 配置文件，将第 80 行的路径改为本机当前路径：

```
74  #目定义配置
75  sast:
76    analysis:
77      version: sast-1.0.0
78      url: http://sast.natapp1.cc
79    base:
80      path: E:/release/sast/parallel/server
81    project:
82      repo: ${sast.base.path}/data/sast-repo
```

2.3.4. 启动

在 server 目录下双击“start.bat”文件（或中文的“一键启动.bat”文件），弹出的窗口不要关闭。窗口中出现下面字样说明启动成功：



2.3.5. license 自助获取

④ 获取 license：访问 SAST 应用页面，通过登录页“许可证设置”导出 sn.txt，依据此文件向支持人员获取 license。

⑤ 导入 license：获取 license 后，同样通过 SAST 应用页面的“许可证设置”导入 license。

2.4. 其他系统环境下安装

其他系统环境，例如银河麒麟、中标麒麟等环境，可凭借环境具体配置（内核、CPU 等）联系技术支持人员获取对应的安装操作手册。

3. 工具使用

3.1. 访问

确保服务是开启的，若需重新开启服务，则根据“2.3.4 启动”章节内容进行操作。

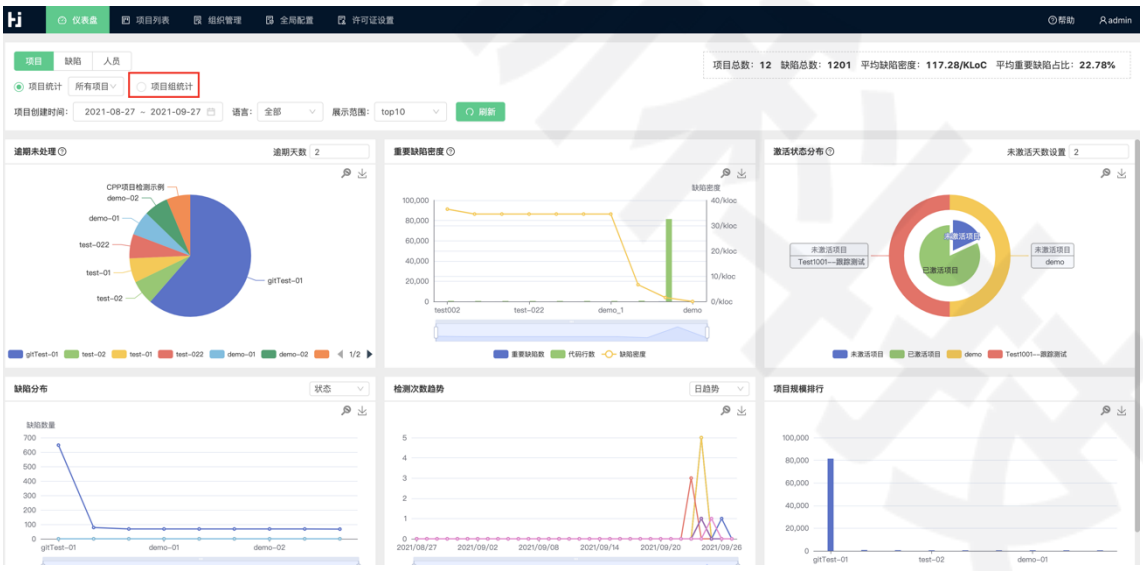
访问服务端：打开浏览器后，在地址栏中输入以下地址及端口号，例：【127.0.0.1:8000/】即可打开 B/S 主界面，如下图所示。默认登录用户信息：用户

名： admin，密码：向指定技术人员获取。

3.2. 数据仪表

3.2.1. 项目数据

点击【仪表盘】一级菜单，默认界面是项目维度的数据分析汇总，可展示项目相关的数据分析图：逾期未处理、重要缺陷密度、激活状态分布、缺陷分布、检测次数趋势、项目规模排行，并且能够以项目组维度（项目组概念详见 3.4 章节）展示数据分析，如下图所示：

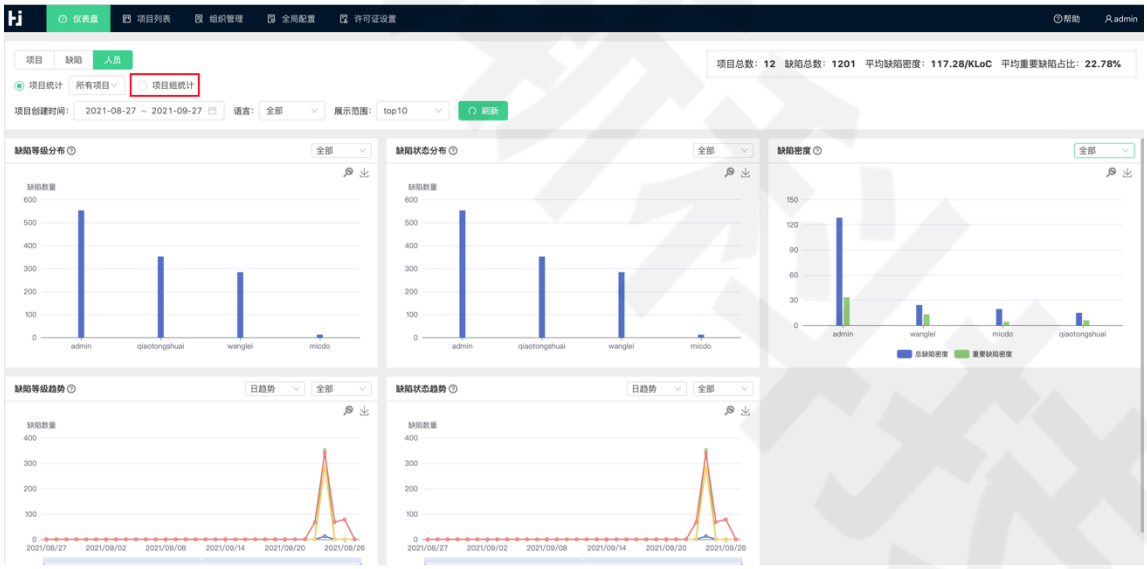


3.2.2. 缺陷数据

点击【仪表盘】下的【缺陷】二级菜单，可展示缺陷维度的数据分析，同样也能够以项目组为维度展示缺陷数据分析，如图所示：

3.2.3. 人员绩效数据

点击【仪表盘】下的【人员】二级菜单，可展示人员绩效维度的数据分析，同样也能够以项目组为维度展示人员绩效的数据分析，如图所示：

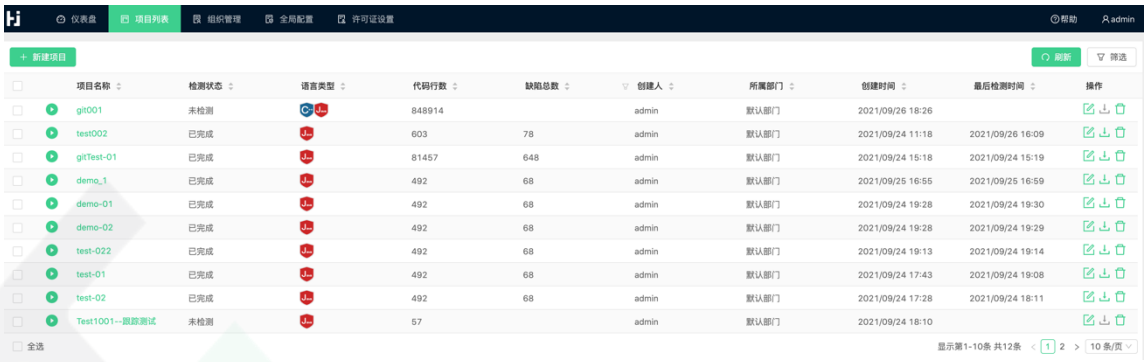


3.3. 代码检测流程

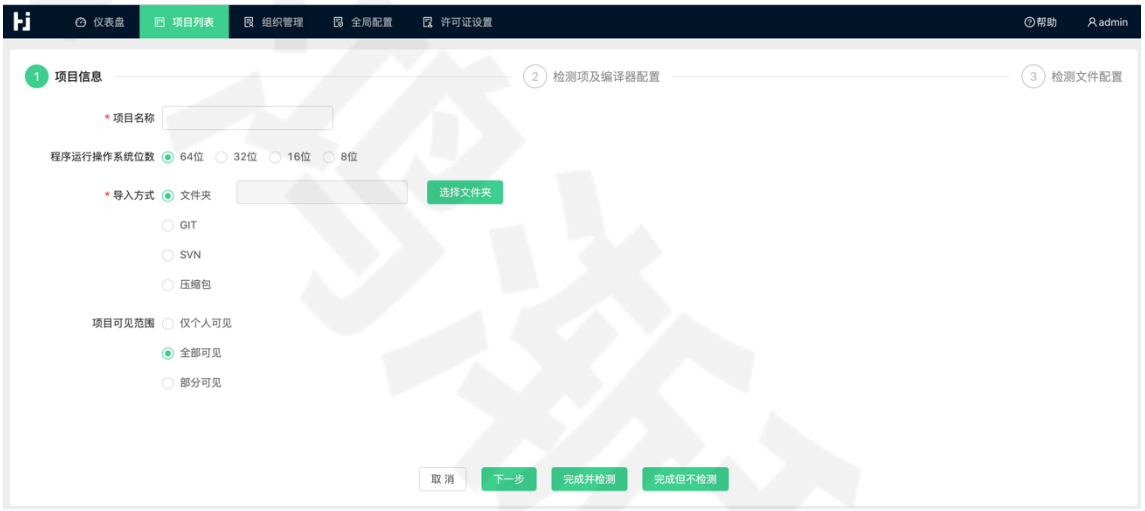
3.3.1. 新建检测项目

点击【项目列表】一级菜单下的【项目队列】，可展示当前所有检测项目的基

本信息，如下图所示：



点击左上方【新建项目】，打开新建项目界面，在该界面中可以新建项目，如图所示：



1．基本信息：

1) 新建项目：点左侧菜单【新建项目】，打开项目导入界面

2) 项目名称：输入导入的项目名称

3) 基本信息：选择基本信息，设置项目的基本信息（默认选择）

4) 项目类型：选择导入的项目类型，支持 C、C++、JAVA、HTML、Kotlin、JavaScript、Python、Scala 等

5) 选择导入方式：支持 Git、SVN、文件夹、压缩包四种导入方式

i. 文件夹：如图，可以选择将整个文件夹作为项目导入，如图所示：

ii. 压缩包：可以选择将整个压缩包作为项目导入

iii. Git：输入 git 地址、用户名、密码，然后单击【浏览】，选择导入的项目。

iv. SVN：输入 SVN 地址、用户名、密码，然后单击【浏览】，选择导入的项目，填写方式同上。

2．检测项及编译器配置

在项目基础信息填写步骤中点击【下一步】，出现以下配置：

1) 数字 1 区域：可在该界面对缺陷类别、编码规则、安全漏洞、度量项进行配置：

① 配置检测规则：点击相应规则名称后的 ⓘ 按钮，出现检测规则详情，可酌情选择为当前项目配置哪些检测规则，忽略哪些。如图：

2) 数字 2 区域：编译器配置。系统内置了多种编译器，可根据需要进行选择，一般默认即可。若自定义配置，则步骤如下：

① 添加库配置：根据项目需要配置编译器：

- 步骤 1：在上图数字 2 区域的库配置处，点击【添加】按钮，根据如下图所示内容进行配置：

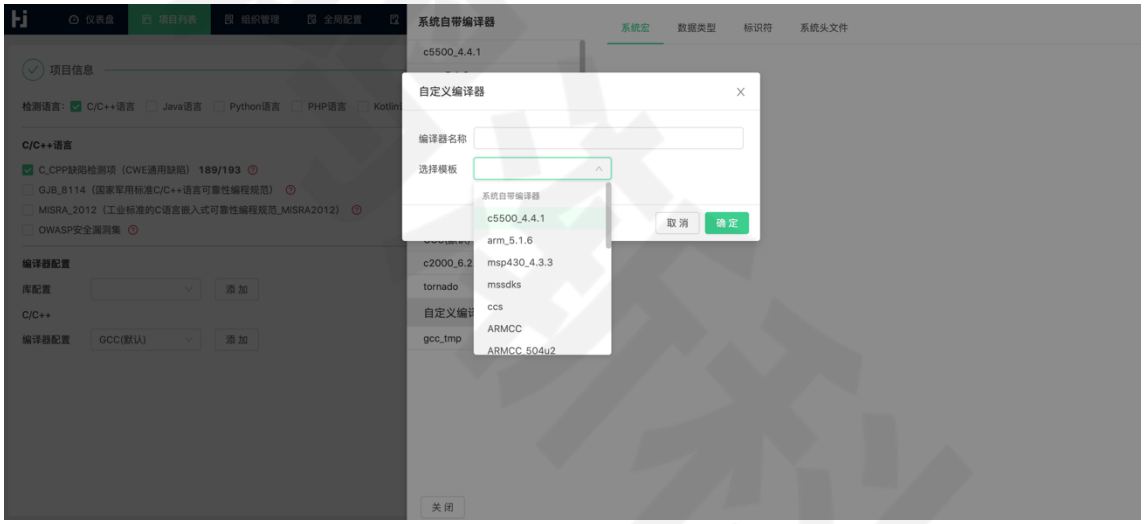
- 步骤 2：选择“用户”
- 步骤 3：点击“新建库”
- 步骤 4：输入库名称
- 步骤 5：选择库配置

- 步骤 6：选择压缩包方式导入库文件
- 步骤 7：点击【确认】

② 添加编译器配置

编译器配置：

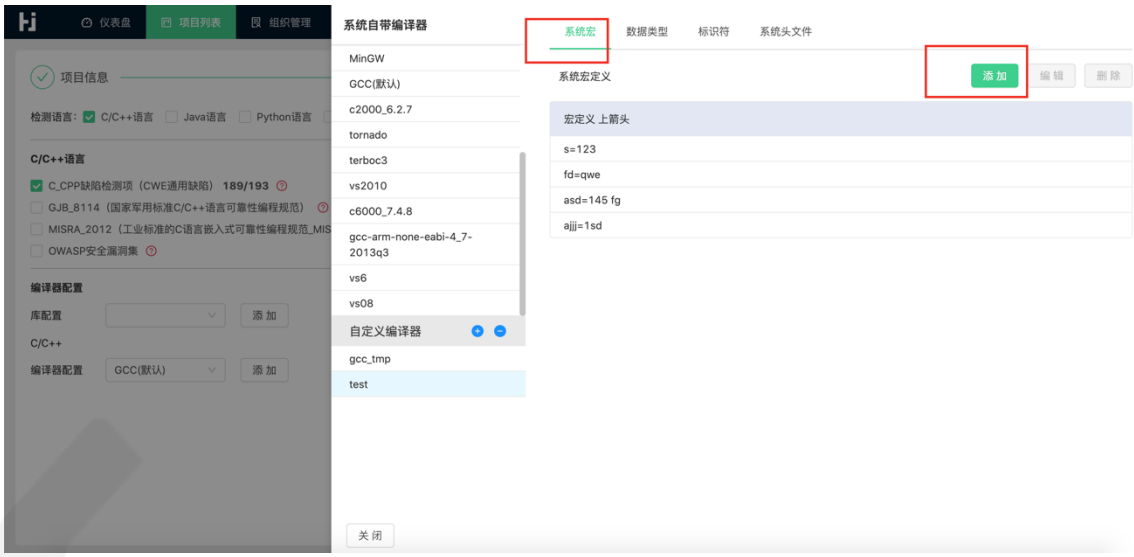
- 步骤 1：在在上图数字 2 区域的编译器配置处，点击【添加】
- 步骤 2：点击自定义编译器处的【添加】按钮，根据如下图所示内容进行配置：



- 步骤 3：输入编译器名称
- 步骤 4：下拉选择模板
- 步骤 5：点击【确定】

宏定义添加：

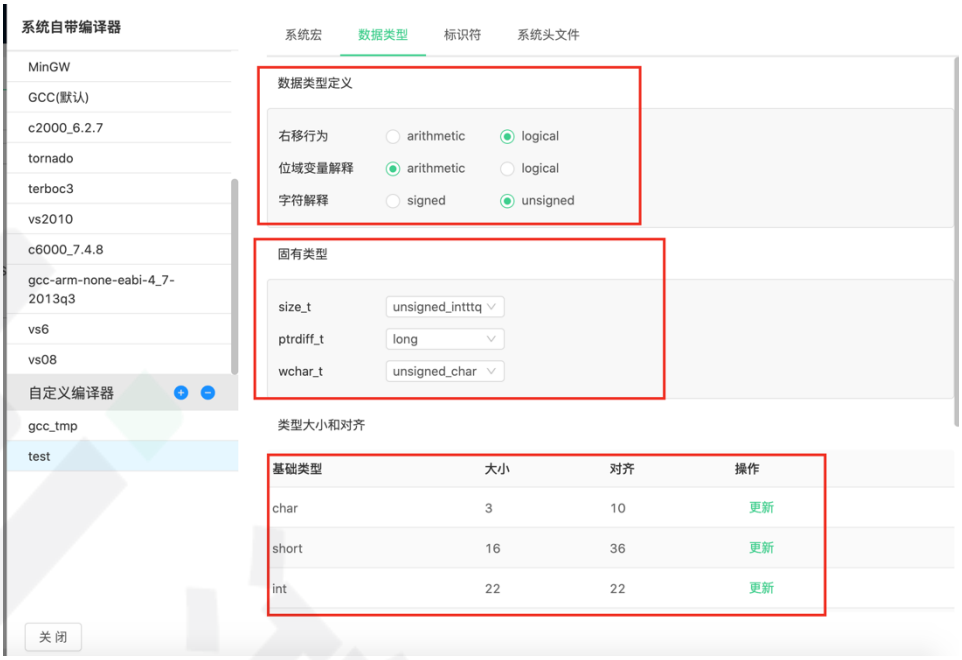
- 步骤 6：点击【添加】系统宏定义：



➢ 步骤 7：按照提示格式输入宏定义：

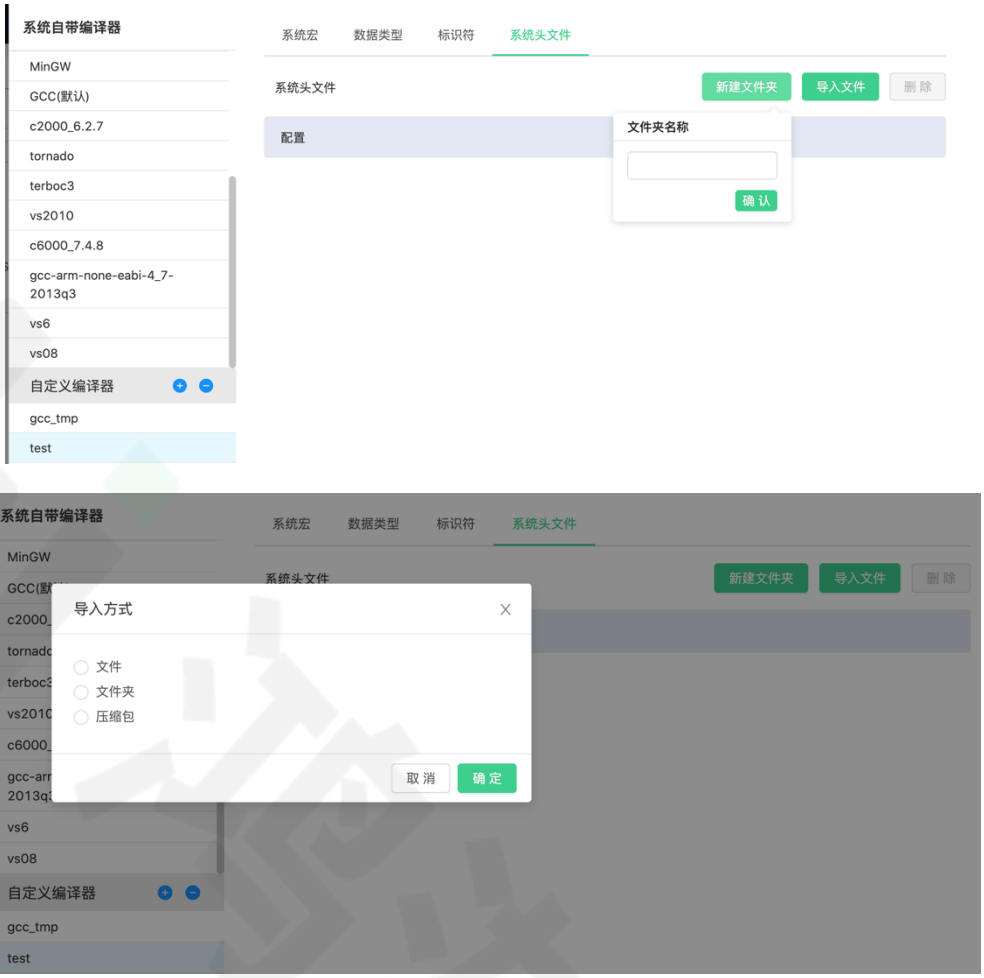
➢ 步骤 8：点击【确认】按钮

编译器数据类型：进行配置，包括对数据类型的定义、固有类型的格式、类型大小和对齐方式的定义。



对编译器的标识符：进行配置，包括内部标识符长度和外部标识符长度，配置完点击【更新】按钮：

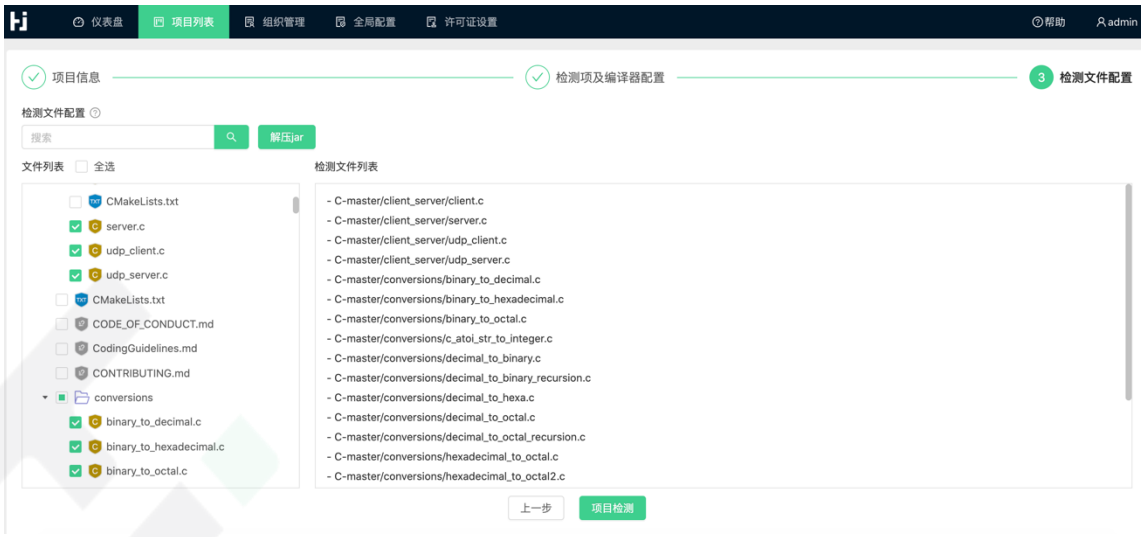
编译器的系统头文件：进行配置，点击【新建文件夹】，然后点击【导入文件】；择导入方式，可选“文件”导入、“文件夹”导入、“压缩包”导入三种方式导入头文件；点击【确定】按钮：



数字 1、数字 2 区域中的所有配置完成，点击【下一步】。

3．检测文件配置

可设置哪些项目需要检测，哪些项目不需要检测：工具根据检测语言自动识别了需要检测的文件，如果需要变更，更改文件的选中框状态即可，如图所示：



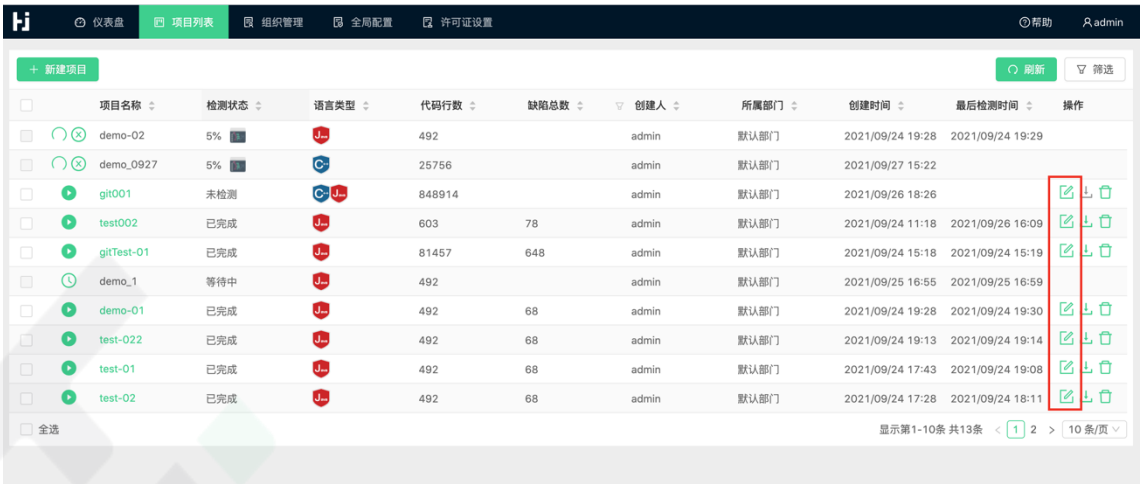
4．创建项目完成：选择完文件、配置设置完成后，点击【项目检测】完成项目的创建。

3.3.2. 检测队列

多并发检测情况下，如果当前检测项目数超过了设定的并发数量，则后来检测项目排队，并且可以调整排队顺序，如图所示，在【项目列表】下的【检测队列】菜单：

3.3.3. 检测项目编辑

在项目列表页面，点击编辑按钮 ，按照需求编辑响应的参数：

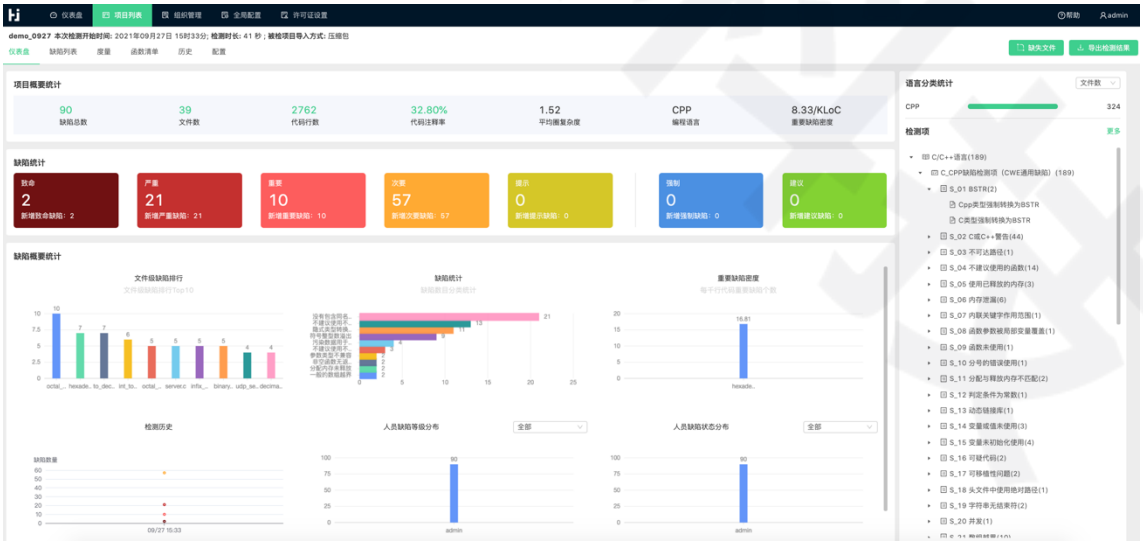


编辑内容如下图所示，并且在【检测器及编译器配置】中，可以补充缺失文件，也可以直接通过项目信息的右上方按钮【缺失文件】进入此功能。

（缺失文件：工具在编译、解析代码过程中遇到的造成编译失败的公共库文件或自研文件。通常工具会根据代码自动补全技术进行智能补全公共库文件，但其他文件需要手动补全，以使检测过程更完美，检测结果更准确）

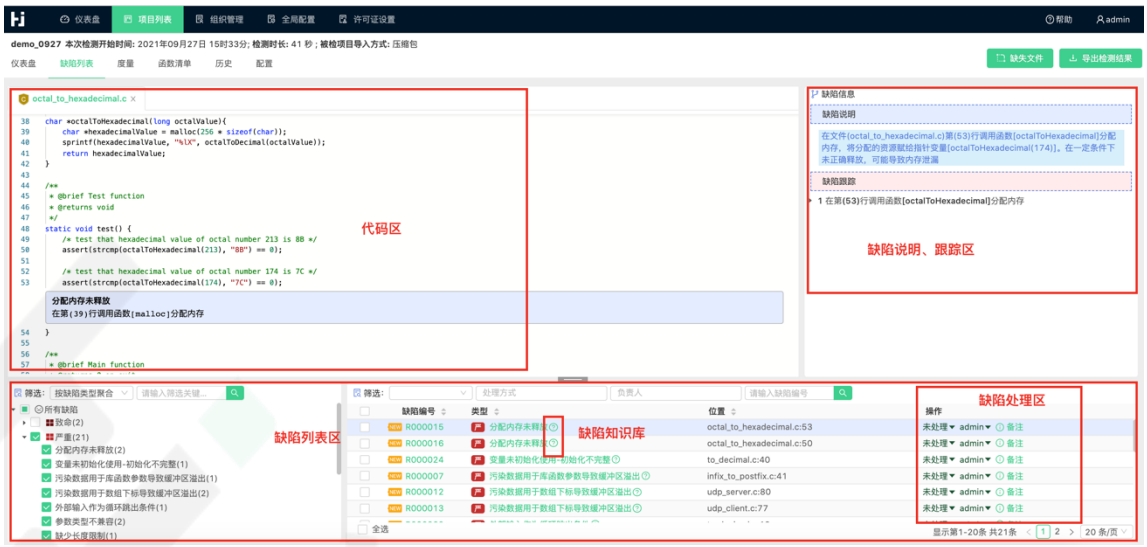
3.3.4. 缺陷结果查看

1．缺陷查看主界面：在项目菜单中选择我们刚才测试的项目“demo_0927”，打开项目总览主界面，如下所示：



2．进入缺陷列表页签，即可查看所有缺陷列表，点击缺陷编号，即可在中间

主区域内查看相关问题。如图所示：



以上各区的功能解释如下：

1) 缺陷列表区：根据查询条件查询后，缺陷的查询结果会在【缺陷树】及右方缺陷列表中根据查询结果列表展示。

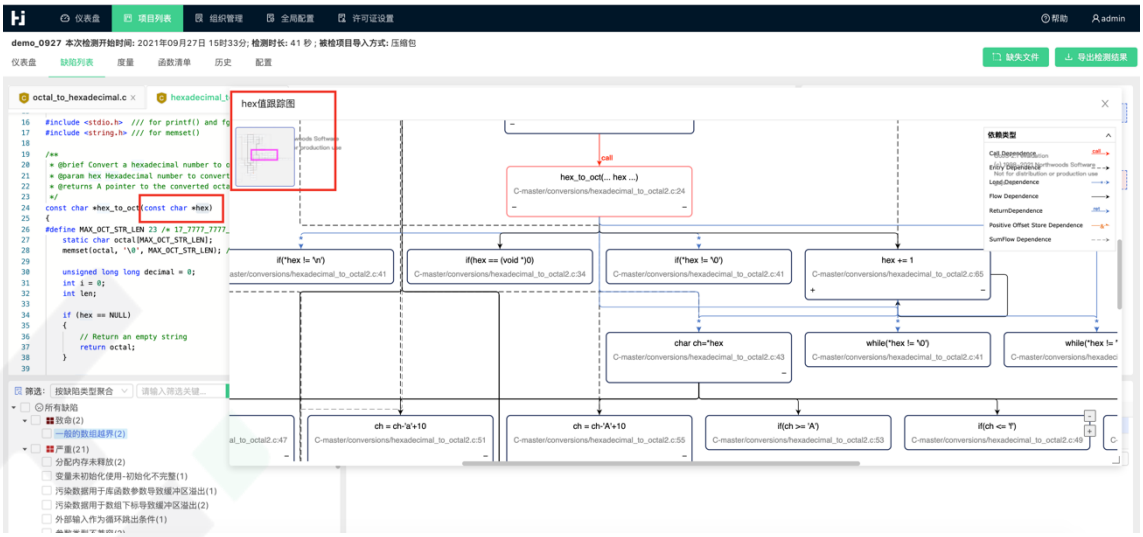
2) 缺陷说明及跟踪区：在【缺陷列表】中，单击一条缺陷，缺陷详细信息区中会显示该条缺陷的详细信息，还可以列表显示缺陷的【引发位置、缺陷的发生位置、缺陷所在文件、缺陷发生行数】，在列表中单击，【缺陷代码主工作区】中的代码，会自动定位到发生问题的代码行，并底色标亮。

3) 缺陷判断区，可以对缺陷进行判断，类别包含【缺陷、误报、待定、故意为之】，同时能对【缺陷等级】进行判定。

4) 缺陷知识库，在缺陷列表的某条缺陷上，单击图标，显示缺陷知识库：缺陷的详细描述、修复建议、代码示例等。

3．代码架构图

在代码文件中选中函数或者变量并右键，可查看函数调用关系图、值跟踪图等代码架构图，帮助用户在脱离 IDE 的环境下准确阅读代码上下文，确认问题。例如值跟踪图，可以展示反映污点变量在程序中经过的上下文处理，如下图所示：



3.3.5. 代码度量

在检测项目的结果页进入【度量页签】，可查看度量列表、度量结果导出，如图所示：

1) 度量列表：根据度量查询及度量项选择，列表区显示查询的度量结果信息。

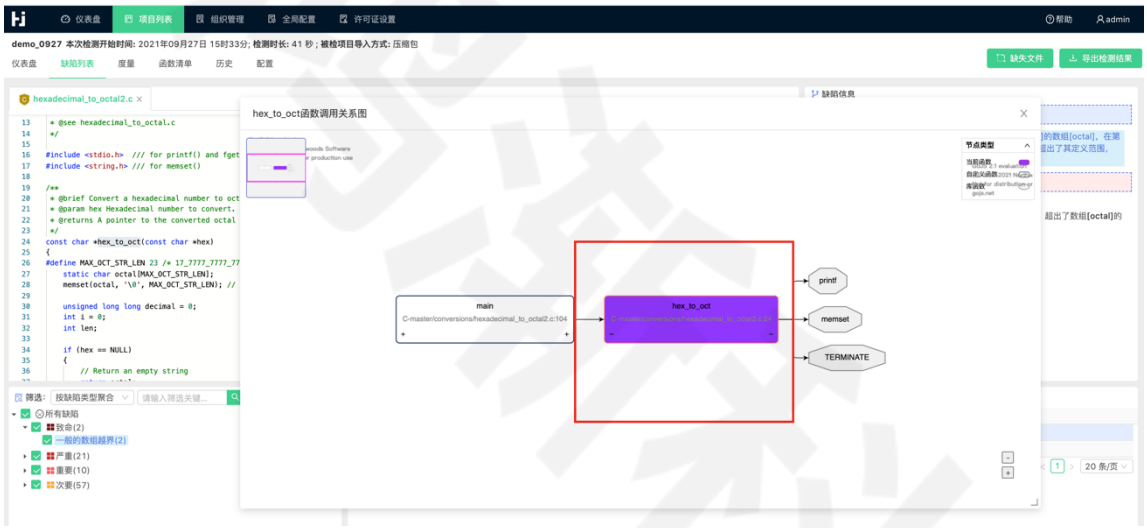
2) 度量导出：将度量结果导出成报告，支持【Excel】格式导出。

3.3.6. 函数清单

在检测项目的结果页进入【函数清单】页签，展示当前项目中的所有函数清单及函数实现代码，并且可配置函数类型，选中之后，当前函数在“3.3.4”章节内容

中的“代码架构图”中会高亮显示。

例如，将“adaline_predict”函数标准为特殊函数之后，再生成函数调用图时，此函数会标紫显示（特殊函数标识和高亮颜色可在 3.5.5 章节中具体配置），如下图所示：



3.3.7. 检测历史

在检测项目的结果页进入【历史】页签，可查看项目的历次检测结果，并且能够看到数据对比分析，如下图所示：

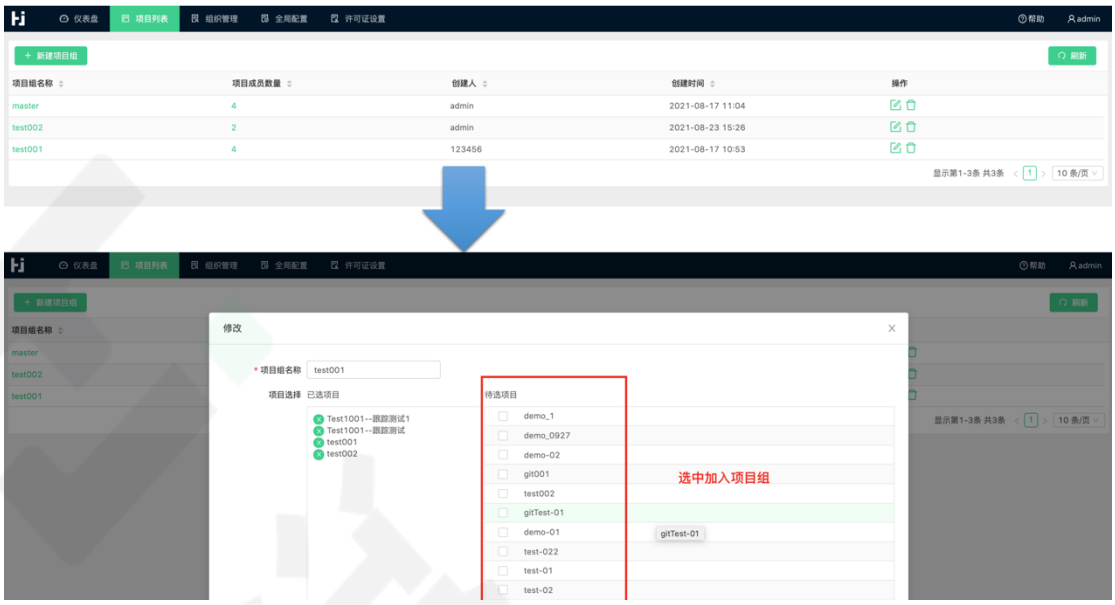
3.3.8. 报告导出

在检测项目的任意结果页签，点击右上方【导出检测结果】，可根据需求自定义导出报告的内容，格式支持 Word 和 Excel。如下图所示：

3.4. 项目组

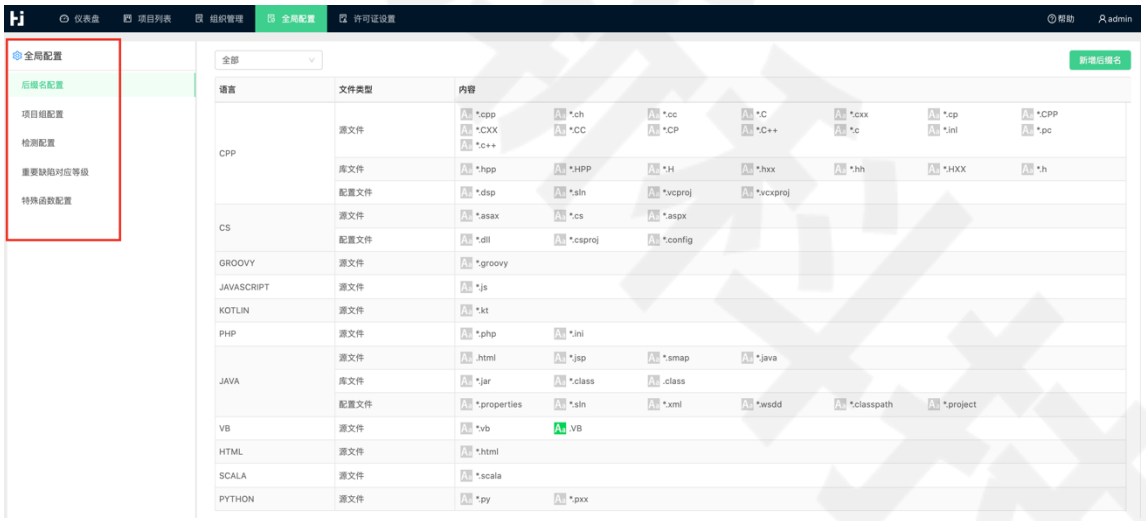
点击一级菜单【项目列表】下的【项目组】二级菜单，可展示项目组的管理页面。项目组的概念：将多个检测项目（代码项目）统一归入一个项目组进行管理，也类似产品—代码项目的概念。

设置项目组时，可先新建项目组，再点击项目组的【编辑】菜单，将检测项目配置进项目组，如图所示：



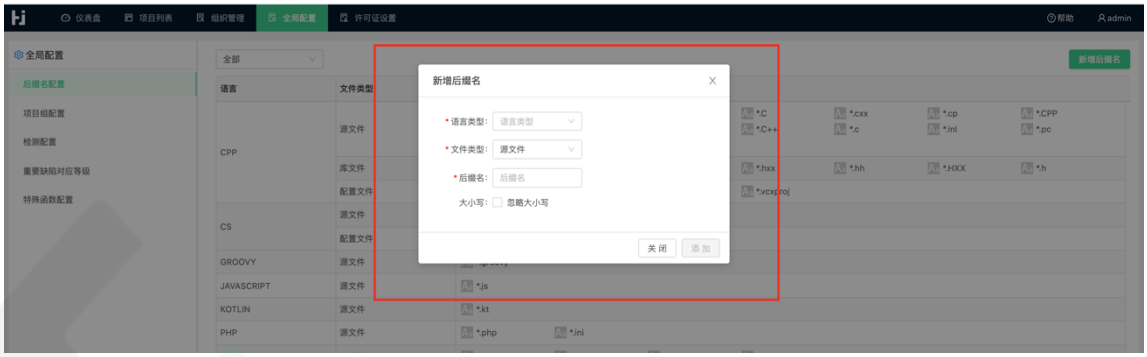
3.5. 全局配置

点击一级菜单【全局配置】，所有的可配置项均在页面左侧二级菜单的功能中。如下图所示：



3.5.1. 后缀名配置

后缀名配置可以新增非通用文件后缀名与开发语言之间的映射关系，帮助工具自动识别开发语言。新增后缀名如下图所示：



3.5.2. 项目组配置

可以配置当前工具是否开启项目组的概念和对应功能菜单，如下图所示：

3.5.3. 检测配置

在【检测配置】中，有 3 项配置：

1) “标注为误报的缺陷，下次是否显示”
2) 代码架构图中，值跟踪图的默认展示层级
3) 代码架构图中，项目函数调用图的默认展示层级

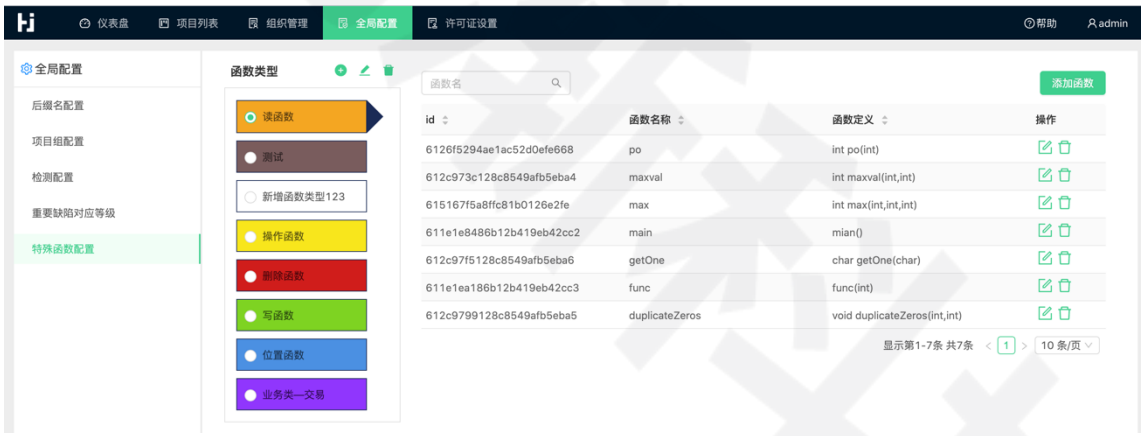
如下图所示：

3.5.4. 重要缺陷等级配置

在数据仪表盘中，有“重要缺陷密度”的数据分析，那么重要缺陷包含了那些级别的问题，可以在此配置，如图所示：

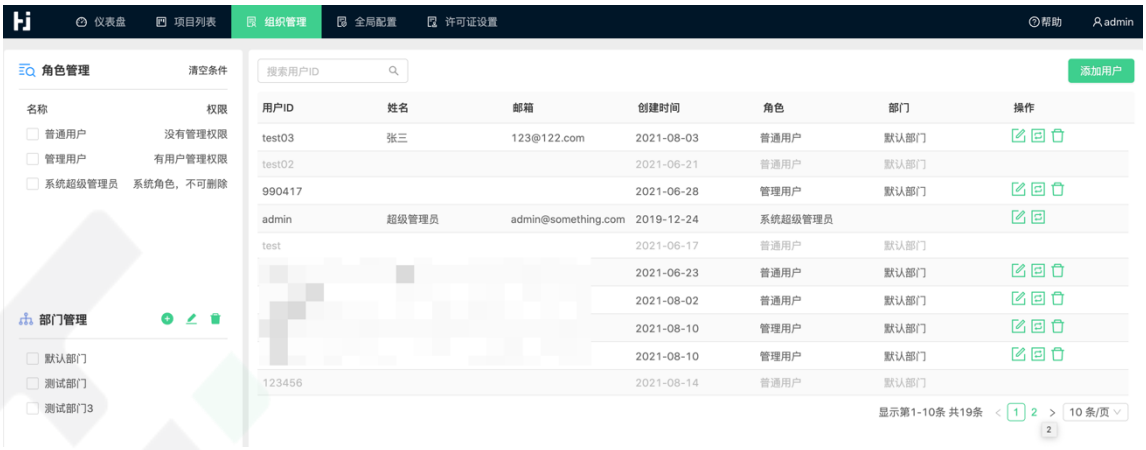
3.5.5. 特殊函数配置

在【特殊函数配置】中，可以自定义配置重要函数的属性，添加重要函数功能如下图所示：



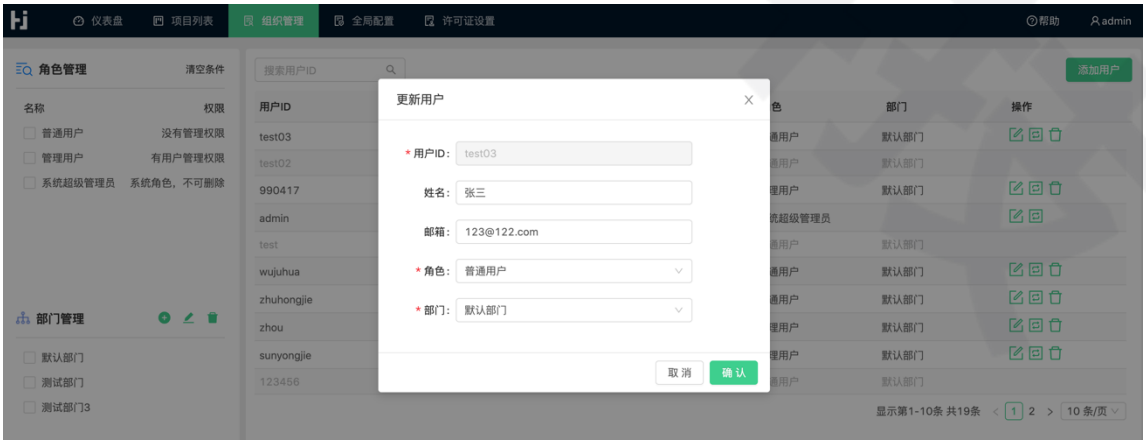
3.6. 权限管理

1．创建新用户：进入首页面，点击菜单栏中的【组织管理—用户管理】，跳转到用户及权限管理页面，如图所示：



① 新建用户：点击图中的【新建】按钮，在【用户信息】栏中可输入新用户的基本信息，若不输入密码，则默认为 123，点击【保存】，即可创建用户。

② 修改用户信息：点击图标 ，修改用户信息，如图所示：



2．角色管理：使用管理员登录系统后，进入【组织管理—用户管理】页面，

在左侧页签，即可添加、修改、删除用户角色组：

3. 部门管理：使用管理员登录系统后，进入【组织管理—用户管理】页面，在左侧下方页签，即可添加、修改、删除部门：