### 整体架构

软件漏洞智能检测平台主要分为３个部分：前端、后端、子节点。

1. 前端主要使用**Vue框架**，在Vue之上使用**Element库**的组件展示（包括数据列表、按钮、图标、展开图等），由Axios库异步请求数据，由**Webpack**模块化开发。
2. 后端包括图１中的服务端和数据两个部分，服务端负责处理来自前端的消息，先将任务信息存入musql数据、将待测软件存入相应的目录中，然后将任务分配给子节点去执行。主要实现４个功能：fuzz任务管理、漏洞修复验证、漏洞管理、节点管理。
3. 子节点现在集成了３款模糊测试框架：afl、libfuzz、honggfuzz。可以依据任务中指定的模糊器来选择特定的模糊测试框架执行。同时，为了提升模糊测试的效率，执行相同任务的子节点之间会定时同步种子，共享最新的模糊测试信息，提升整体模糊测试的效率。

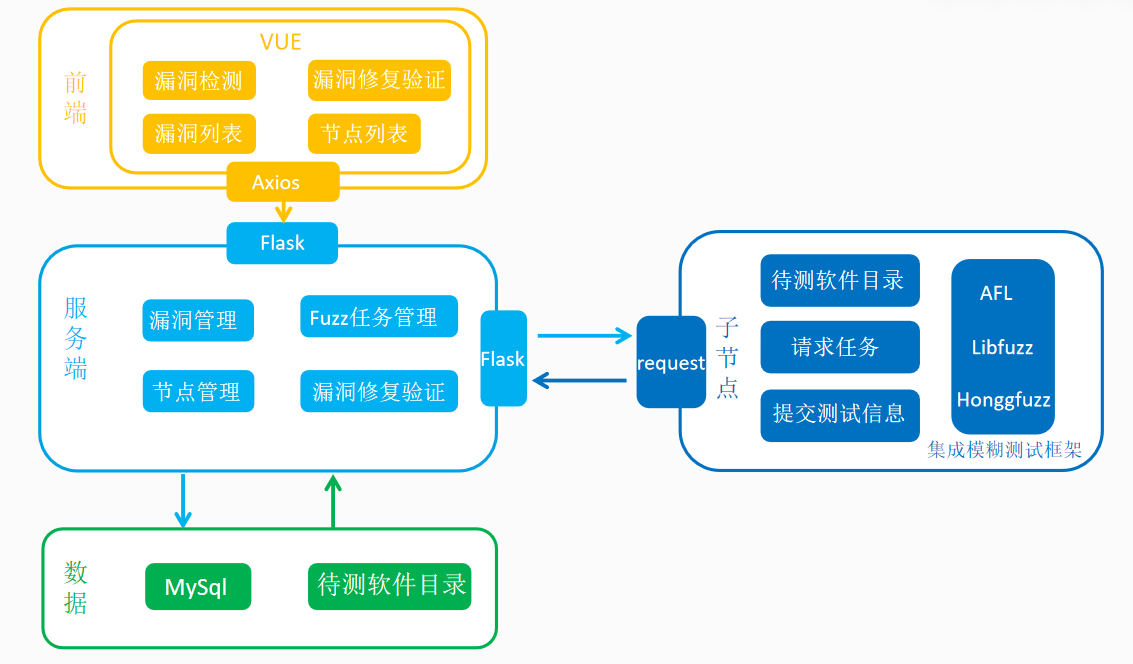


图１　系统整体架构图

### 功能模块

整个系统主要分为四个功能模块：软件漏洞检测、软件漏洞修复验证、漏洞信息管理、测试节点管理。

软件漏洞检测

# Todo 补一个模糊测试流程图

测试人员通过前端的可视化界面上传待测试软件，并且设定测试的时间和节点数量，以及调用的模糊测试框架。

后端接收到来自前端的任务信息后，会将任务信息存入mysql数据库的jobs表中，并将待测软件存入指定的目录下。

集群中的子节点在没有任务的情况下会每隔10秒向主节点请求一次任务。后端会从数据库中查询等待检测的任务，如果存在，就将任务信息和待侧软件发送给子节点进行测试。在测试过程中，子节点会将种子队列同步到主节点中，在从主节点中更新本地的种子队列。除了定时同步种子外，子节点还会将运行过程中的日志信息和漏洞信息同步到主节点中，然后后端会将漏洞信息存入mysql中，最后通过前端页面将漏洞信息和运行日志展示出来。定时同步的时间现在设置为30s一次同步。

＃Todo　补一个种子同步图解

种子同步机制介绍：

　　在节点本地会将本地的种子上传主节点，并将上传的种子记录在seed\_exist集合中，下次上传的时候避免重复上传。上传到主节点的种子，会依据nodename+seednum来命名，以区分不同节点上传的种子。

之后，当子节点来获取种子时，主节点会将不是由该节点上传的种子进行打包，发送给子节点，子节点收到压缩包之后，对其中的种子进行判别，如果已经存在本地种子池中，则丢弃，否则加入，加入时，会将种子的命名信息存入seed\_exist集合中，防止重复添加和上传。

软件漏洞修复验证

# Todo 补一个漏洞修复验证流程图

在前端上传已经修复好了的软件，并且指定测试的漏洞用例。后端会将该漏洞修复任务信息存入数据库，然后当有子节点来请求任务的时候，将任务分配给子节点进行检验。之后，后端会将子节点的检验结果传递给前端展示出来。

测试节点管理

　　子节点在向主节点请求任务的时候，后端会更新子节点的活跃时间，以此来管理集群中存活的节点。测试人员也可以通过空闲的节点数量以及待测任务的重要性来分配测试资源。

###### 漏洞信息管理

　　管理各个待测软件中所存在的漏洞信息。用于前端的漏洞信息展示以及漏洞修复验证中的漏洞选择。

### 模糊测试框架介绍

子节点中集成的模糊测试框架现在主要包括三种：afl、libfuzz、honggfuzz。均是采用最先进的基于覆盖率指导的模糊测试框架。其差别如下图２所示：

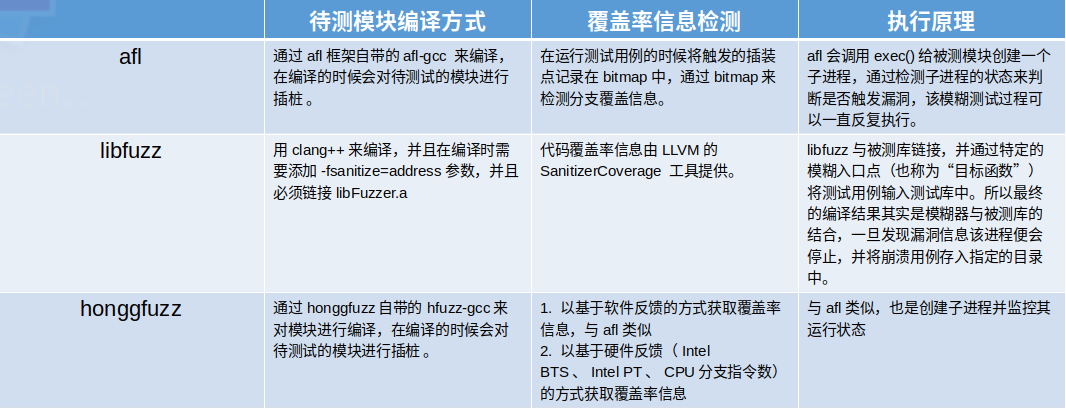


图２　各模糊测试框架对比

### 数据库介绍

数据库采用mysql，　数据库名为fuzz，一共包含四个表：jobs、crashes、nodes、reproduce。分别记录创建的模糊测试任务、发现的漏洞信息记录、可用的节点资源、创建的漏洞修复验证任务。

１. 创建任务表

create table jobs(

id int(20) primary key NOT NULL AUTO\_INCREMENT,

name char(30),

fuzzer char(15),

botnum int,

surplusnum int COMMENT '任务剩余数',

completenum int COMMENT '任务完成度',

time int,

exec char(20),

jobpath varchar(100),

crashnum int,

createtime datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间'

);

/\* 不同的节点获取任务的时候会将surplusnum-1，当surplusnum的值为０的时候，就表示任务已经分配完毕，当每一个节点执行任务完毕之后，会给主节点发送消息，completenum＋１，当completenum == botnum的时候，表示任务已经完成。

　　一个模糊测试任务总的测试时间是botnum＊time。

\*/

２. 创建漏洞库表

create table crashes(

id int(20) primary key NOT NULL AUTO\_INCREMENT,

name char(60) ,

jobid int,

crashpath varchar(100),

isfix int,

findtime datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间'

);

/\*name: 任务名\_节点名\_crash\_num # num是当前crash目录下的漏洞个数+1,避免文件名冲突。

将name中的”crash”替换为"info"就是该漏洞对应的漏洞详细信息的文件名

isfix: 1已经修复，0未修复，default 0。

\*/

３ 创建节点表

create table nodes(

id int primary key NOT NULL AUTO\_INCREMENT,

name char(20),

ipaddr char(20),

men int, // GB

corenum int,

jobid int(20),

fetchtime datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '节点提取任务时间',

last\_modify\_time datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '最后活跃时间',

createtime datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间'

);

/\* jobid 0代表空闲; 非0代表fuzz任务id,繁忙; -1表示正在执行漏洞修复验证任务。

\*/

４. 创建漏洞复现任务表

create table reproduce(

id int primary key NOT NULL AUTO\_INCREMENT,

name char(20),

fuzzer char(15),

crashname char(60),

exec char(20),

jobpath varchar(100),

isfetch int,

completed int,

isfixed int,

createtime datetime null DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间'

);

/\*

isfetch 1已有节点获取任务，0还没有节点获取任务

completed 是否已经验证完成，0不是，1已完成

isfixed 漏洞是否已经修复，0未修复，1已修复

\*/

### 注意

1. 在每个节点上启动的时候需要设置afl的运行环境。如下：

su root

echo core >/proc/sys/kernel/core\_pattern

cd /sys/devices/system/cpu

echo performance | tee cpu\*/cpufreq/scaling\_governor

### 待改进

1. 一个节点多次运行同一个任务的时候会存在问题？
2. 前端界面中的模糊测试任务列表无法分页显示？

解：分页没有问题，数据库中的任务多次执行导致botnum　< completenum的缘故。

1. 前端界面修整、美化？
   1. 漏洞修复验证时无法自动同步数据信息，需要点击其他栏目才能显示最新的数据，能否每隔一秒自动刷新？
   2. 类似师兄的运行界面图
   3. 添加一个搜索栏目
   4. 任务创建的时候需要添加条件限制
   5. 换页符号箭头呈现白色
2. Honggfuzz日志信息量太大了，前端可能无法显示？
3. Window系统打开前端界面的时候会无法上传待测软件？
4. 主节点的漏洞去重？
5. 子节点在运行过程中死掉了，模糊测试任务永远都无法完成了？
6. 后端删除死亡节点？
7. 当节点资源不足且都在运行fuzz时，漏洞复现需要有等很久。