思路：

找到各开源代码的文件分布功能模块

对公开的请求id检查的话，考虑路径混淆检查

检查是客户端注册时，client\_id的格式设置，是否有格式错误，在收到client\_id之后用路径混淆的方式插入一个假的client\_id，可以去检查客户端对授权服务器发送code请求时client\_id相关代码，以及授权服务器将code和client\_id绑定时相关代码

由于建于TLS之上：

1. 加密防止窃听，即使敌手截获到包也无法得知其中内容
2. 防篡改，如果第三方敌手尝试篡改传输中的数据包内容，​**TLS本身的设计会检测到篡改并导致通信失败或中断**，攻击者无法成功地将篡改后的内容送达接收方而不被发现。
3. 防劫持，能够有效防止中间人攻击
4. 认证身份，防冒充
5. 防止重放攻击，

各步通信的内容能被截取，但是截取方无法理解其内容 也无法篡改内容

虽然client\_id和code绑定，但是若仅仅根据client\_id来认证身份的话，怎么能保证 client\_id不会被伪造呢，需不需要一个对客户端的认证，去查看协议标准对这一部分的 描述，然后去查找认证的相关代码

精读论文中的漏洞，查找论文中的漏洞哪些被修复了：可以问田师姐

检测到重复的授权码，有没有撤销此前颁发的所有授权码

在重定向uri处还有许多规范需要遵守，之后工作可以对照这些规范去检查。ai提示词：详细讲解一下redirect\_uri白名单机制

不安全的路径模式检测 在通配符检测方法areWildcardsAllowed中

rootUrl如果被篡改 可能导致重定向到恶意站点

| **编号** | **授权类型** | **描述** |
| --- | --- | --- |
| P1 | 授权码模式, 简化模式 | OAuth服务器****必须****在发送重定向响应前，严格验证授权请求中的重定向URI参数与客户端注册URI完全匹配 (RFC-6749) |
| P2 | 授权码模式, 简化模式 | 授权请求中的重定向URI参数****必须****是绝对URI。OAuth服务器应使用模式匹配API执行验证 (RFC-6749, RFC-6819) |
| P3 | 授权码模式 | 授权码****必须****单次有效，兑换令牌后立即作废。若重复使用已撤销的授权码，服务器必须返回错误 (RFC-6749) |
| P4 | 授权码模式 | 授权码****必须****绑定到特定客户端。在令牌端点颁发令牌前，服务器需验证授权码确系颁发给该客户端 (RFC-6749, OAuth安全最佳实践) |
| P5 | 授权码模式 | 授权码****必须****绑定到其颁发时的客户端重定向URI。颁发令牌前需验证该关联关系 (RFC-6749) |
| P6 | 授权码模式 | 服务器****必须****在授权端点存储PKCE参数（code\_challenge和code\_challenge\_method），供令牌端点验证使用 (RFC-7636, OAuth安全最佳实践) |
| P7 | 授权码模式 | 服务器****必须****在令牌端点验证PKCE参数真实性。code\_verifier参数的转换值（如SHA256）必须与code\_challenge值一致 (RFC-7636, OAuth安全最佳实践) |
| P8 | 授权码模式, 简化模式 | 服务器****必须****通过处理state参数提供CSRF防护。该参数值必须添加到授权请求的重定向响应中 (RFC-6749, OAuth安全最佳实践) |
| P9 | 授权码模式, 简化模式 | 服务器颁发的访问令牌****应****约束到特定客户端。mTLS是标准化约束机制：从TLS栈获取客户端证书，解码并哈希后与访问令牌关联 (RFC-8705, OAuth安全最佳实践) |
| P10 | 授权码模式, 简化模式 | 服务器****不应****明文存储访问令牌，仅存储其哈希值 (RFC-6819) |