

前身：

请求提取：

首先，将源代码提交到CI服务器，在该服务器上运行一系列用于自动测试和代码检查的工具。此处部署的静态分析引擎用于提取可能启动的调用，并生成一个清单文件来描述这些调用。微服务使用有限数量的协议进行通信，如HTTP、gRPC等。因此，为了缩小状态空间并加速静态分析，Jarvis首先扫描控制流中的代码，并识别进行网络API调用（如请求）的语句。get（）。接下来，它使用这些作为起点，在控制流上执行反向污染传播，以获得与每个请求相关的程序片。最后，Jarvise通过语义分析从切片中提取有用的属性，如asU RL、method等，并使用通配符表示依赖于上游请求输入的不确定字段

策略生成：

1. 对于多版本：在访问控制策略中用标记来区分
2. Jarvision还通过实际的服务注册信息确定哪些微服务提供了它调用的接口，即找到被调用方
3. 服务间流量管理规则也会影响细粒度授权
4. Jarvis还从基础设施的策略数据库中提取此类规则。对于上面提到的所有数据源，Permission Engine最终生成访问控制策略，将其聚合以进行管理和后续更新，并将其部署到Policy DB中