



网络安全创新大会
Cyber Security Innovation Summit

Security By Default: MyBatis框架下SQL注入解决方案



汪昱

快手高级安全工程师

- 背景与现状
- 初阶安全实践
- 最佳安全实践：Security By Default

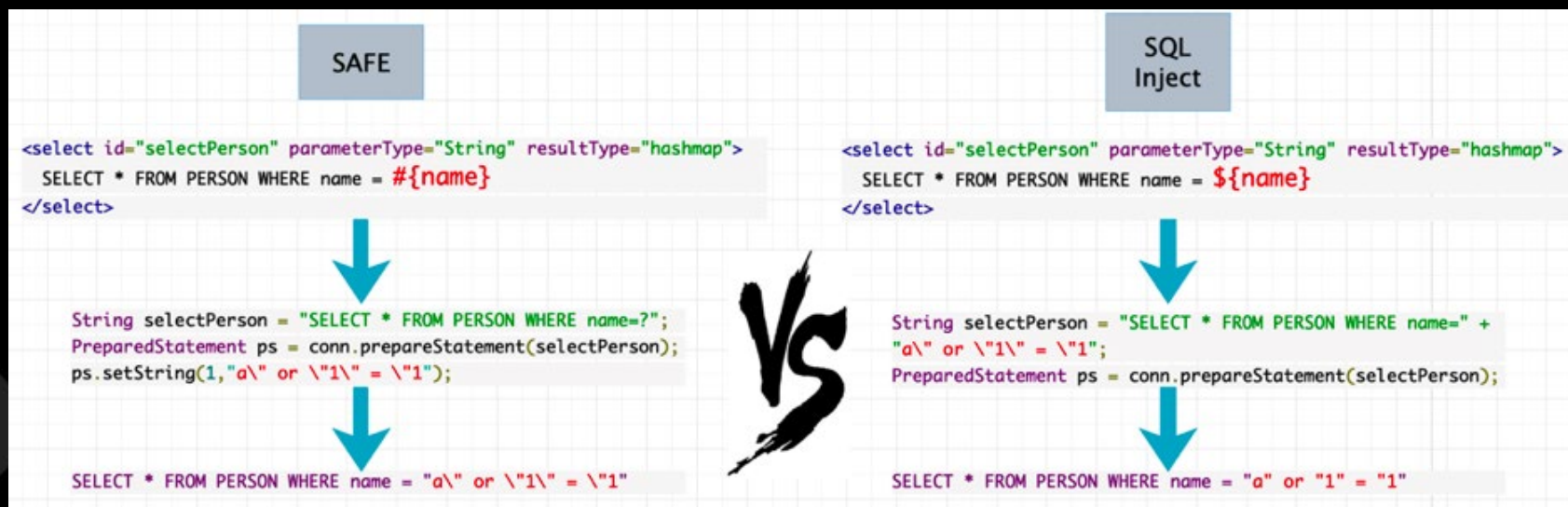


- 据“非官方”甲方安全数据统计，SQL注入仍然占据不少公司高危安全漏洞Top 3之列，在SQL注入中，order by注入比例最高
- 2022年都要结束了，为什么SQL注入还是痛点问题之一？
→ RD有犯错的机会



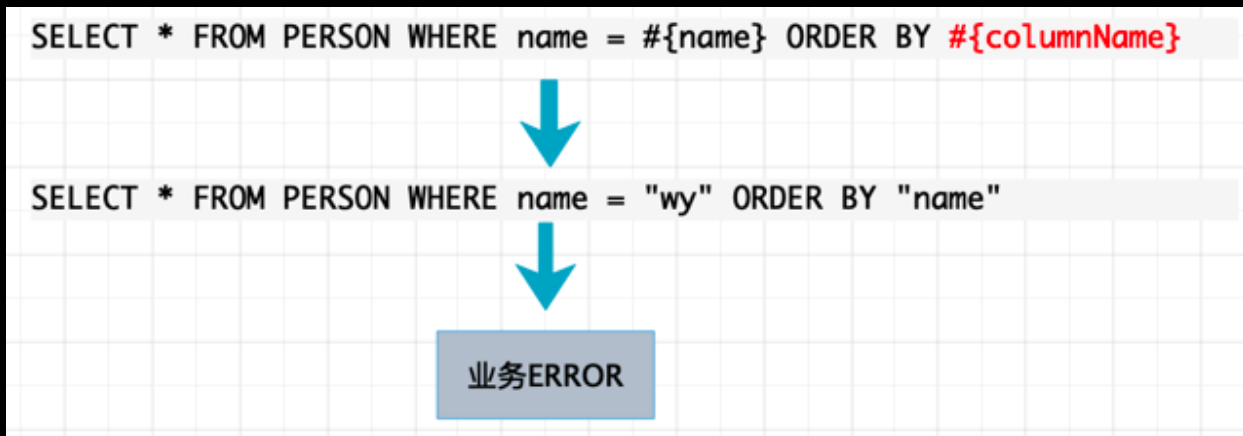
- MyBatis是一款在国内外使用度很高的ORM框架
- 京东、阿里、美团等以Java为业务主要开发语言的公司均有使用
- MyBatis框架下，SQL注入漏洞依旧时常发生

- 众所周知，SQL拼接是SQL注入的罪魁祸首
- 众所周知，预编译是SQL注入的安全编码方案
- 众所周知，MyBatis下的SQL动态传参有2种方法：
 - ✓ `{param}`是预编译方式
 - ✗ `${param}`是直接拼接方式



#{param}不是万能的

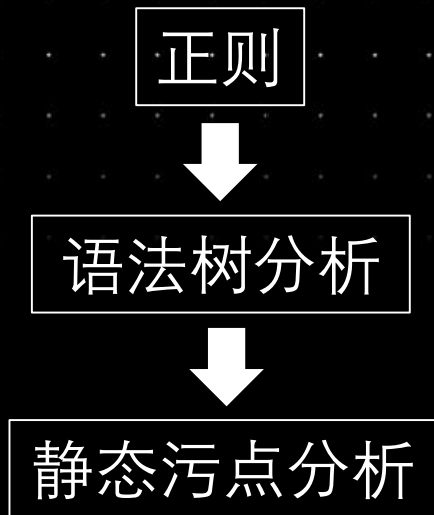
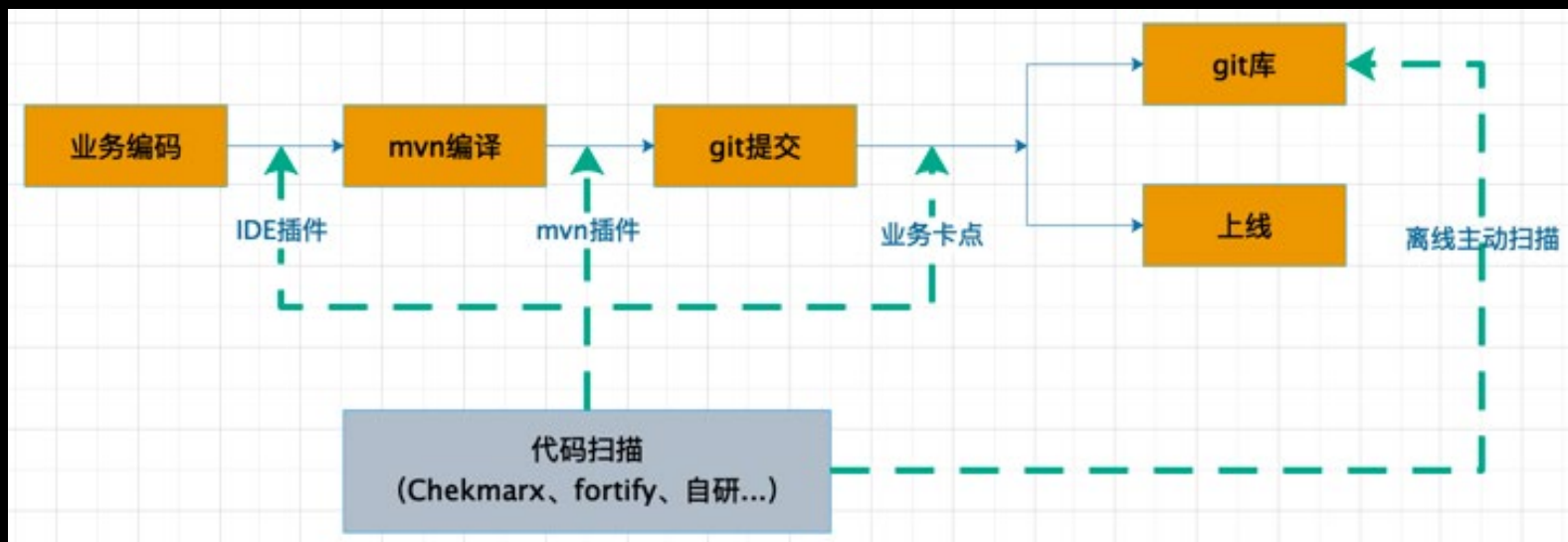
- MyBatis层order by、group by等地方不能使用#{param}方式进行预编译，只能使用\${param}直接拼接



- MyBatis给了安全提示，但是也给了RD犯错的机会

⚠ Tips from MyBatis

NOTE It's not safe to accept input from a user and supply it to a statement unmodified in this way. This leads to potential SQL Injection attacks and therefore you should either disallow user input in these fields, or always perform your own escapes and checks.



- 大多数漏洞，任何静态分析技术都不能(接近)零误报零漏报地自动化检测
 - 举个栗子：order by \${param}拼接了参数，但是在Controller层限制了仅接受["column1","column2"]的用户输入
- 将漏洞检测转化成“合理的”规范检测是大型甲方常用“套路”（褒）

- 背景与现状
- 初阶安全实践
- 最佳安全实践：Security By Default

解决可以用#{ }传参的SQL注入

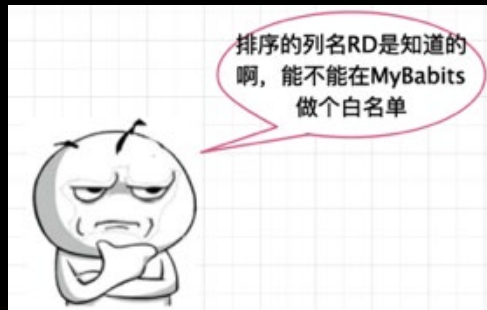
- “合理的”的安全编码规范？

- ✓ MyBatis下，能用#{ }方式的一律不许用\${ }方式拼接
- ？ 类似order by、group by这些不能用#{ }的，必须在业务层校验输入

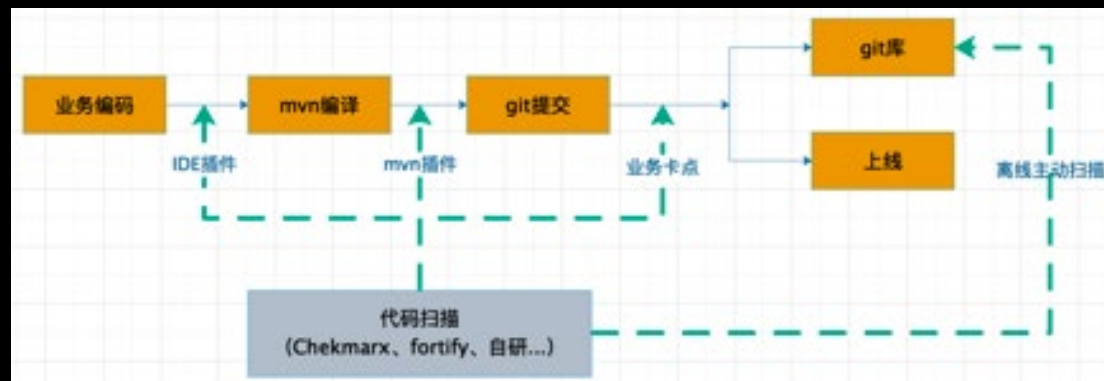
这种“教育为主”的规范，安全部门无法知道RD到底有没有正确对不能使用#{ }地方进行输入限制



限制 → 污点净化
“污点净化”是白盒检测的难点



```
<select id="sortSQL">
  SELECT field_name FROM table_name WHERE 1 = 1
  <if test="columnName != null and columnName=='name'.toString()">
    order by name
    <if test="orderName !=null and orderName=='desc'.toString()">
      desc
    </if>
    <if test="orderName !=null and orderName=='asc'.toString()">
      asc
    </if>
  </if>
</select>
```



业务自行限制
(污点净化差异性)



Mapper处限制
(污点净化标准化)



- MyBatis Generator(MBG)是用于自动生成MyBatis Mapper文件的工具，可以有效提高研发效率，使用范围很广
- MBG自动生成的Mapper存在使用\${}直接拼接参数的情况

- order by处存在\${}
- Example_Where_Clause和Update_By_Example_Where_Clause处存在\${}

```
<select id="selectByExample" parameterType="com.venscor.codesec.codescan.bean.CodeRuleExample" resultMap="BaseResultMap">
  select
  <if test="distinct">
    distinct
  </if>
  <include refid="Base_Column_List" />
  from codecheckrule
  <if test="_parameter != null">
    <include refid="Example_Where_Clause" />
  </if>
  <if test="orderByClause != null">
    order by ${orderByClause}
  </if>
</select>
```

MBG生成的 order by 存在\${} 拼接

- order by处存在\${}
- Example_Where_Clause和Update_By_Example_Where_Clause处存在\${}

```
<sql id="Example_Where_Clause">
  <!--
    WARNING - @mbg.generated
    This element is automatically generated by MyBatis Generator, do not modify.
    This element was generated on Fri Dec 27 14:20:43 CST 2019.
  -->
  <where>
    <foreach collection="oredCriteria" item="criteria" separator="or">
      <if test="criteria.valid">
        <trim prefix="( " prefixOverrides="and" suffix=")">
          <foreach collection="criteria.criteria" item="criterion">
            <choose>
              <when test="criterion.noValue">
                and ${criterion.condition}
              </when>
              <when test="criterion.singleValue">
                and ${criterion.condition} #{criterion.value}
              </when>
              <when test="criterion.betweenValue">
                and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
              </when>
              <when test="criterion.listValue">
                and ${criterion.condition}
                <foreach close=")" collection="criterion.value" item="listItem" open="( " separator=",">
                  #{listItem}
                </foreach>
              </when>
            </choose>
          </foreach>
        </trim>
      </if>
    </foreach>
  </where>
</sql>
```

存在\${}拼接方式

```
<sql id="Update_By_Example_Where_Clause">
  <where>
    <foreach collection="example.oredCriteria" item="criteria" separator="or">
      <if test="criteria.valid">
        <trim prefix="( " prefixOverrides="and" suffix=")">
          <foreach collection="criteria.criteria" item="criterion">
            <choose>
              <when test="criterion.noValue">
                and ${criterion.condition}
              </when>
              <when test="criterion.singleValue">
                and ${criterion.condition} #{criterion.value}
              </when>
              <when test="criterion.betweenValue">
                and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
              </when>
              <when test="criterion.listValue">
                and ${criterion.condition}
                <foreach close=")" collection="criterion.value" item="listItem" open="( " separator=",">
                  #{listItem}
                </foreach>
              </when>
            </choose>
          </foreach>
        </trim>
      </if>
    </foreach>
  </where>
</sql>
```

存在\${}拼接方式

- 背景与现状
- 初阶安全实践
- 最佳安全实践：Security By Default

- 在初阶解决方案中，如果业务手写Mapper，CI/CD已经完成卡点检测（量小反弹小）
- 问题聚焦在使用MBG生成Mapper的安全问题

```
<where>
<foreach collection="oredCriteria" item="criteria" separator="or">
  <if test="criteria.valid">
    <trim prefix="(" prefixOverrides="and" suffix=")">
      <foreach collection="criteria.criteria" item="criterion">
        <choose>
          <when test="criterion.noValue">
            and ${criterion.condition}
          </when>
          <when test="criterion.singleValue">
            and ${criterion.condition} #{criterion.value}
          </when>
          <when test="criterion.betweenValue">
            and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
          </when>
          <when test="criterion.listValue">
            and ${criterion.condition}
            <foreach close=")" collection="criterion.value" item="listItem" open="(" separator=",">
              #{listItem}
            </foreach>
          </when>
        </choose>
      </foreach>
    </trim>
  </if>
</foreach>
</where>
```

在 condition 可被控制时存在注入风险

```
<select id="selectByExample" parameterType="com.venscor.codesec.codescan.bean.CodeRuleExample" resultMap="BaseResultMap">
  select
    <if test="distinct">
      distinct
    </if>
    <include refid="Base_Column_List" />
  from codecheckrule
  <if test="_parameter != null">
    <include refid="Example_Where_Clause" />
  </if>
  <if test="orderByClause != null">
    order by ${orderByClause}
  </if>
</select>
```

MBG 生成的 order by 存在 \${} 拼接

- 操作condition的API全部是protected作用域
- ✓ RD不会使用此API, 因此不会有外部可控参数导致SQL注入

```
<where>
  <foreach collection="oredCriteria" item="criteria" separator="or">
    <if test="criteria.valid">
      <trim prefix="(" prefixOverrides="and" suffix=")">
        <foreach collection="criteria.criteria" item="criterion">
          <choose>
            <when test="criterion.noValue">
              and ${criterion.condition}
            </when>
            <when test="criterion.singleValue">
              and ${criterion.condition} #{criterion.value}
            </when>
            <when test="criterion.betweenValue">
              and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
            </when>
            <when test="criterion.listValue">
              and ${criterion.condition}
              <foreach close=")" collection="criterion.value" item="listItem" open="(" separator=",">
                #{listItem}
              </foreach>
            </when>
          </choose>
        </foreach>
      </trim>
    </if>
  </foreach>
</where>
```

在 condition 可被控制时存在注入风险

```
protected abstract static class GeneratedCriteria {
    ...

    protected void addCriterion(String condition) {
        if (condition == null) {
            throw new RuntimeException("Value for condition cannot be null");
        }
        criteria.add(new Criterion(condition));
    }

    protected void addCriterion(String condition, Object value, String property) {
        if (value == null) {
            throw new RuntimeException("Value for " + property + " cannot be null");
        }
        criteria.add(new Criterion(condition, value));
    }

    protected void addCriterion(String condition, Object value1, Object value2, String property) {
        if (value1 == null || value2 == null) {
            throw new RuntimeException("Between values for " + property + " cannot be null");
        }
        criteria.add(new Criterion(condition, value1, value2));
    }
}
```

普通 RD 不会使用此 API, 也就是说不会写出漏洞

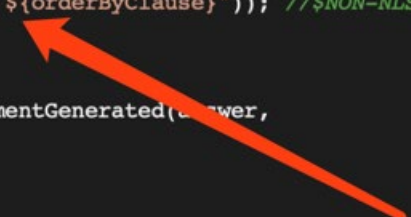
- order by处参数，RD可以拼接用户参数传入

✗ MBG提供了不安全的实现

- org.mybatis.generator/codegen/mybatis3/xmlmapper/elements/SelectByExampleWithBLOBsElementGenerator.java
- org.mybatis.generator/codegen/mybatis3/xmlmapper/elements/SelectByExampleWithoutBLOBsElementGenerator.java
- org.mybatis.generator/codegen/mybatis3/javamapper/elements/sqlprovider/ProviderSelectByExampleWithoutBLOBsMethodGenerator.java

```
//SelectByExampleWithoutBLOBsElementGenerator
public void addElements(XmlElement parentElement) {

    ...
    ifElement = new XmlElement("if"); //$NON-NLS-1$
    ifElement.addAttribute(new Attribute("test", "orderByClause != null")); //$NON-NLS-1$ //$NON-NLS-2$
    ifElement.addElement(new TextElement("order by ${orderByClause}")); //$NON-NLS-1$
    answer.addElement(ifElement);
    if (context.getPlugins()
        .sqlMapSelectByExampleWithBLOBsElementGenerated(answer,
            introspectedTable)) {
        parentElement.addElement(answer);
    }
}
```



- MBG生成的Mapper代码，对于Order By的处理是不安全的，存在SQL注入的风险



```
<select id="sortSQL">
    SELECT field_name FROM table_name WHERE 1 = 1
    <if test="columnName != null and columnName=='name'.toString()">
        order by name
        <if test="orderName !=null and orderName=='desc'.toString()">
            desc
        </if>
        <if test="orderName !=null and orderName=='asc'.toString()">
            asc
        </if>
    </if>
</select>
```

- MBG扩展机制是MBG提供的一种修改默认行为的方式

🔗 参考: <https://mybatis.org/generator/reference/extending.html>

- (1) 修改XMLMapper的order by生成
- (2) 修改JavaMapper的order by生成 (生成安全的mybatis注解)
- (3) 修改查询POJO
- (4) 修改扩展点 (让MBG加载我们的修改行为)
- (5) 修改其他有耦合关系的类

- 项目源码地址:

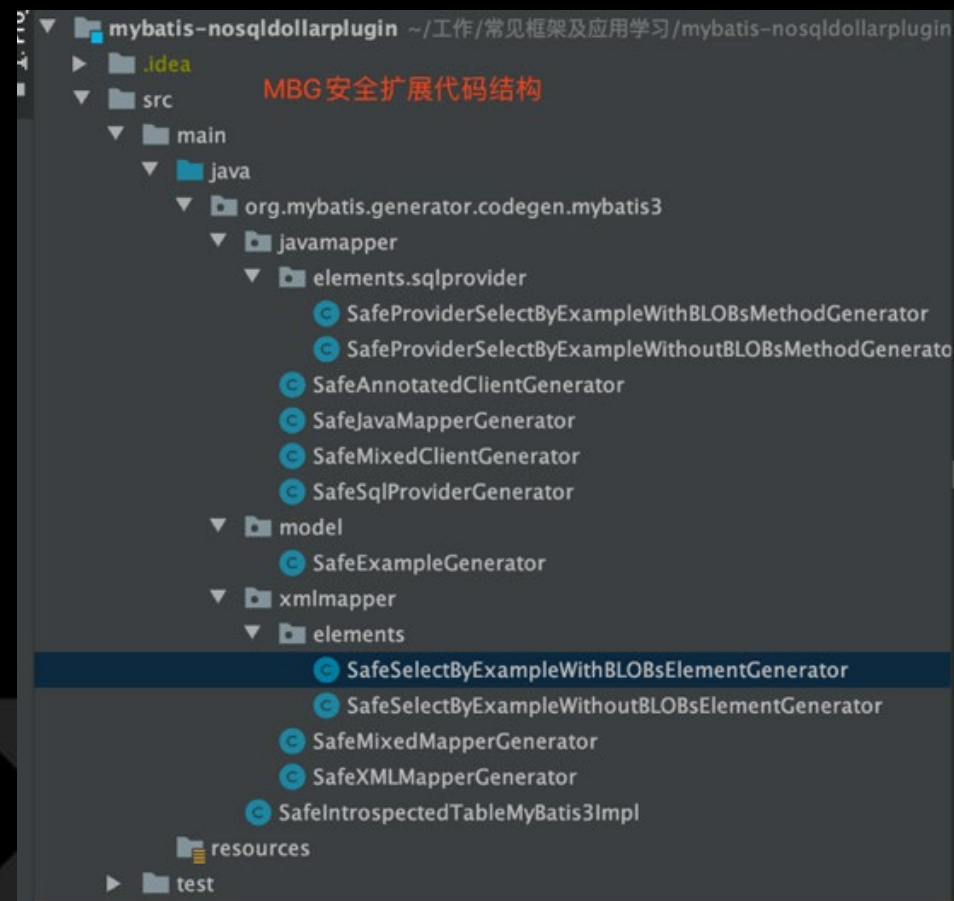
<https://github.com/Venscor/mybatis-generator-nodollar>

- 兼容性:

对1.3.x和1.4.x均兼容

- 落地情况:

在我司已成熟落地



- 安全部门编译插件加入公司私服
- 研发做以下配置

(1) 引入依赖

```
<dependency>
  <groupId>com.venscor.codesec</groupId>
  <artifactId>mybatis-generator-nosqldollar-plugin</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

(2) 研发修改一行配置, 引入MBG扩展 (就是这么简单)

```
<generatorConfiguration>
  <!-- 数据库驱动包位置 -->
  <classPathEntry
    location="/export/mvn_repository/mysql/mysql-connector-java/8.0.16/mysql-connector-java-8.0.16.jar"/>
  <!--
  <context id="MysqlTables" targetRuntime="MyBatis3">-->
  <context id="MysqlTables" targetRuntime="org.mybatis.generator.codegen.mybatis3.SafeIntrosppectedTableMyBatis3Impl">
  <!--
    <plugin type="com.venscor.mybatisplugin.NoSqlDollarPlugin"></plugin>-->
  <commentGenerator>
    <property name="suppressAllComments" value="false"/>
    <property name="suppressDate" value="false"/>
  </commentGenerator>
```

- MyBatis安全模型配合SAST，可有效解决SQL注入问题：
 - ✓ where等可预编译处以#{param} 实现Security By Default;
 - ✓ order by等拼接处，通过MBG安全扩展实现Security By Default。
- 甲方DevSecOps背景下，SAST配合安全加固可高效收敛“技术类”安全问题：
 - ✓ 用SDK突破业务代码的“污点净化” 差异性难点;
 - ✓ 用中间件加固突破“供应链安全”痛点问题;
 - ✓ ...

THANKS



招人，欢迎各位安全/研发专家加入快手安全！