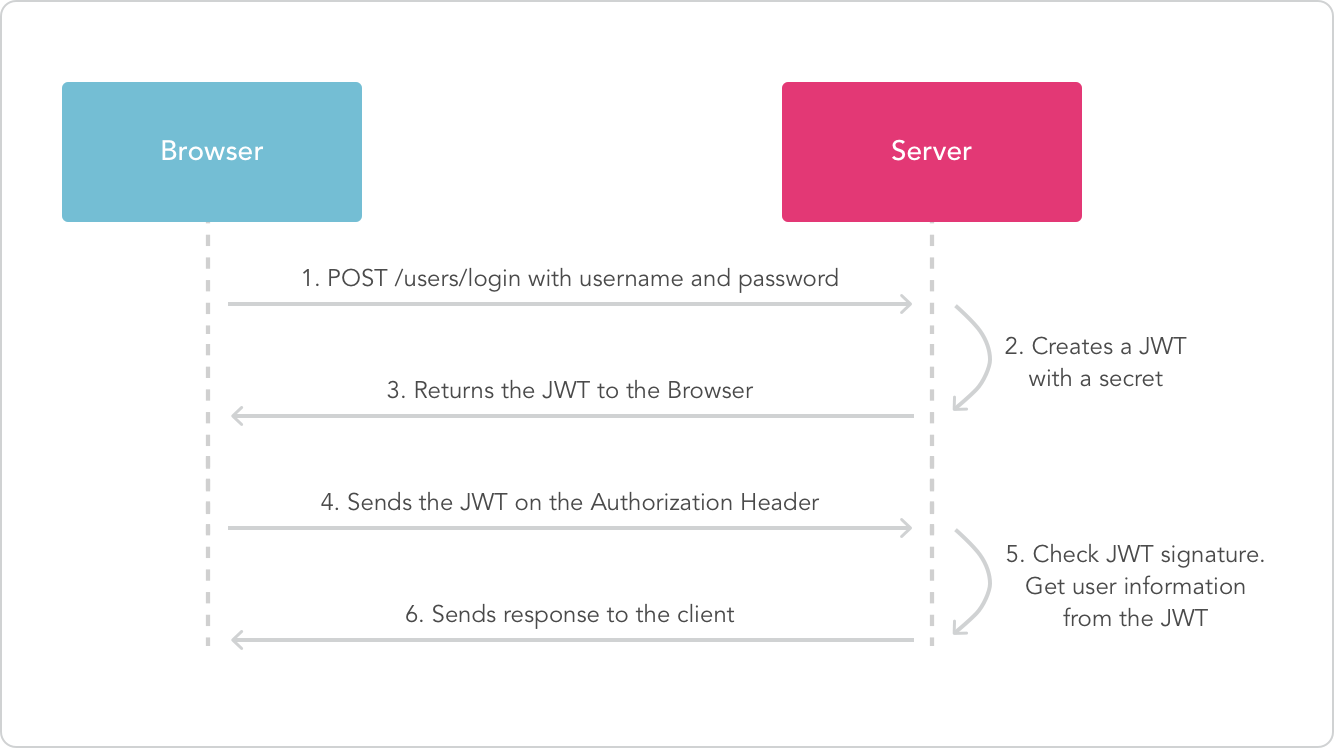
Web开发-JavaEE应用&SpringBoot栈&身份验证&JWT令牌&Security鉴权&安全绕过



#知识点：  
1、安全开发-原生PHP-超级全局变量  
2、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
0、安全开发-原生PHP-数据库通讯  
1、安全开发-原生PHP-身份验证技术  
2、安全开发-Cookie&Session&Token  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-弱类型脆弱  
2、安全开发-原生PHP-函数&数据类型  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-开发组件集合  
2、安全开发-原生PHP-模版引擎渲染  
3、安全开发-原生PHP-第三方编辑器  
---------------------------------------------  
1、安全开发-框架技术-ThinkPHP  
2、安全开发-框架安全-版本&写法  
3、安全开发-ThinkPHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-文件安全操作  
2、安全开发-原生PHP-上传读取删除包含等  
3、安全开发-原生PHP-代码审计文件安全---------------------------------------------  
1、安全开发-原生PHP-RCE安全  
2、安全开发-原生PHP-代码执行&命令注入  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-PHP.INI安全  
2、安全开发-原生PHP-全局文件&单函数  
3、安全开发-原生PHP-流量检测&AI算法  
---------------------------------------------  
1、安全开发-原生JS-语法模型概念  
2、安全开发-Ajax技术-JQuery&Axios  
3、安全开发-Ajax应用-文件上传&用户登录---------------------------------------------  
1、安全开发-原生JS-DOM树&BOM对象  
2、安全开发-原生JS-DOM安全&安全案例  
---------------------------------------------  
1、安全开发-原生JS-数据加密&代码混淆  
2、安全开发-原生JS-数据解密安全案例  
---------------------------------------------  
1、安全开发-NodeJS-开发环境&功能实现  
2、安全开发-NodeJS-安全漏洞&案例分析  
3、安全开发-NodeJS-特有漏洞&代码审计  
---------------------------------------------  
1、安全开发-WebPack-构建打包器  
2、安全开发-WebPack-源码泄漏还原  
---------------------------------------------  
1、安全开发-微信小程序-搭建&开发&架构&安全  
2、安全开发-微信小程序-编译调试&反编译&泄露  
---------------------------------------------  
1、安全开发-JavaEE-Servlet技术  
2、安全开发-JavaEE-监听器&过滤器  
---------------------------------------------  
0、安全开发-JavaEE-构建工具-Maven  
1、安全开发-JavaEE-ORM框架-JDBC  
2、安全开发-JavaEE-ORM框架-Mybatis  
3、安全开发-JavaEE-ORM框架-Hibernate  
4、安全开发-JavaEE-ORM框架-SQL注入&预编译  
---------------------------------------------  
1、安全开发-JavaEE-类加载器&反射机制&链安全  
2、安全开发-JavaEE-成员变量&成员方法&构造方法  
---------------------------------------------  
1、安全开发-JavaEE-动态代理&序列化&反序列化  
2、安全开发-JavaEE-readObject&toString方法---------------------------------------------  
1、安全开发-JavaEE-原生序列化-URLDNS链分析  
2、安全开发-JavaEE-FastJson-JdbcRowSetImpl链分析  
---------------------------------------------  
1、安全开发-JavaEE-JNDI注入-LADP&RMI&DNS等  
2、安全开发-JavaEE-JNDI注入-项目工具&手工原理等  
---------------------------------------------  
1、安全开发-JavaEE-第三方依赖开发安全  
2、安全开发-JavaEE-数据转换&FastJson&XStream  
3、安全开发-JavaEE-Shiro身份验证&Log4j日志处理  
---------------------------------------------  
1、安全开发-JavaEE-开发框架-SpringBoot&路由&传参  
2、安全开发-JavaEE-模版引擎-Thymeleaf&Freemarker&Velocity  
---------------------------------------------  
1、安全开发-JavaEE-常见依赖-Actuator&Swagger  
2、安全开发-JavaEE-安全问题-配置安全&接口测试  
---------------------------------------------  
1、安全开发-JavaEE-身份验证-JWT&Security  
2、安全开发-JavaEE-安全问题-不安全写法&版本漏洞  
   
#章节点：(待补充)  
#具体点：(待补充)



演示案例：

* WEB开发-JavaEE-Servlet&路由&生命周期
* WEB开发-JavaEE-过滤器Filter&监听器Listen
* WEB开发-JavaEE-Maven&JDBC&Mybatis&Hibernate
* WEB开发-JavaEE-ORM框架-SQL注入&预编译
* WEB开发-JavaEE-类加载器&反射机制&安全问题
* WEB开发-JavaEE-成员变量&构造方法&成员方法
* WEB开发-JavaEE-动态代理&序列化&反序列化
* WEB开发-JavaEE-利用链&危险方法&重写方法
* WEB开发-JavaEE-原生反序列化&URLDNS链
* WEB开发-JavaEE-FastJson反序列化&JDBC链
* WEB开发-JavaEE-JNDI注入&LDAP&RMI服务
* WEB开发-JavaEE-第三方依赖-FastJson&XStream
* WEB开发-JavaEE-第三方依赖-Shrio验证&Log4j日志
* WEB开发-JavaEE-开发框架-SpringBoot&路由&传参
* WEB开发-JavaEE-模版引擎-Thymeleaf&Freemarker&Velocity
* WEB开发-JavaEE-监控依赖-SpringBoot&Actuator&配置安全
* WEB开发-JavaEE-API依赖-SpringBoot&Swagger&接口测试
* WEB开发-JavaEE-身份验证-JWT令牌&Security机制&安全问题

#开发框架-SpringBoot  
参考：<https://springdoc.cn/spring-boot/>  
   
#身份验证的常见技术：  
1、JWT  
2、Shiro  
3、Spring Security  
4、OAuth 2.0  
5、SSO  
6、JAAS等  
   
#身份验证-JWT技术  
JWT(JSON Web Token)是由服务端用加密算法对信息签名来保证其完整性和不可伪造；Token里可以包含所有必要信息，这样服务端就无需保存任何关于用户或会话的信息；  
JWT用于身份认证、会话维持等。由三部分组成，header、payload与signature。  
1、引入依赖  
<dependency>  
 <groupId>com.auth0</groupId>  
 <artifactId>java-jwt</artifactId>  
 <version>3.4.0</version>  
</dependency>  
2、创建JWT  
JWT.create()  
3、配置JWT  
JWT.create()  
//header  
.withHeader(map)  
//payload  
.withClaim("userid",id)  
.withClaim("username",user)  
.withClaim("password",pass)  
//signature  
.sign(Algorithm.HMAC256("xiaodisec"));  
4、解析JWT  
//构建解密注册  
JWTVerifier jwt = JWT.require(Algorithm.HMAC256("xiaodisec")).build();  
//解密注册数据  
DecodedJWT verify = jwt.verify(jwtdata);  
//提取解密数据  
Integer userid = verify.getClaim("userid").asInt();  
5、登录校验  
总结：在未知的算法密钥下，即使修改JWT值里的内容去伪造用户，也无法达到认证成功  
6、安全问题  
参考：<https://mp.weixin.qq.com/s/xH_v825bNqDszwmMOe8CBw>  
   
   
#身份验证-Spring Security  
Spring Security安全框架，是Spring Boot底层安全模块默认的技术选型，可以实现强大的Web安全控制。  
WebSecurityConfigurerAdapter：自定义Security策略  
AuthenticationManagerBuilder：自定义认证策略  
@EnableWebSecurity：开启WebSecurity模式  
"认证"和"授权"(访问控制)   
"认证"(Authentication)  
"授权"(Authorization)  
这个概念是通用的，而不是只在 Spring Security 中存在。  
参考官网：https://spring.io/projects/spring-security  
   
1、新建Spring Security+web+thymeleaf项目  
2、配置application.properties模版解析  
3、添加前端页面文件到templates目录  
4、创建路由控制器并指向前端页面文件  
@Controller  
public class RouterController {  
 @RequestMapping("/index")  
 public String index() {  
 return "index";  
 }  
   
 @RequestMapping("/toLogin")  
 public String toLogin() {  
 return "views/login";  
 }  
   
 @RequestMapping("/level1/{id}")  
 public String level1(@PathVariable("id") int id) {  
 return "views/level1/"+id;  
 }  
   
 @RequestMapping("/level2/{id}")  
 public String level2(@PathVariable("id") int id) {  
 return "views/level2/"+id;  
 }  
   
 @RequestMapping("/level3/{id}")  
 public String level3(@PathVariable("id") int id) {  
 return "views/level3/"+id;  
 }  
   
}  
   
5、创建Security授权文件并开启访问策略  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
   
 http.authorizeHttpRequests()  
 .antMatchers("/").permitAll()  
 .antMatchers("/level1/\*\*").hasRole("vip1")  
 .antMatchers("/level2/\*\*").hasRole("vip2")  
 .antMatchers("/level3/\*\*").hasRole("vip3");  
 http.formLogin();  
 }  
   
6、添加认证用户密码并进行密码加密操作  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.inMemoryAuthentication().passwordEncoder(new BCryptPasswordEncoder())  
 .withUser("admin").password(new BCryptPasswordEncoder().encode("123456")).roles("vip1", "vip2", "vip3")  
 .and()  
 .withUser("xiaodi").password(new BCryptPasswordEncoder().encode("123456")).roles("vip1<https://mp.weixin.qq.com/s/5tj6O4TA04QWyWnsd-EmEA>  
er("xiaodisec").password(new BCryptPasswordEncoder().encode("123456")).roles("vip2")  
 .and()  
 .withUser("gay").password(new BCryptPasswordEncoder().encode("123456")).roles("vip3");  
   
 }  
   
7、安全问题  
参考：  
https://mp.weixin.qq.com/s/5tj6O4TA04QWyWnsd-EmEA  
<https://mp.weixin.qq.com/s/M1FiPKJRAWgwaKCtyNW8eQ>  
除去本身的代码不安全写法外，还有版本漏洞导致的安全问题  
演示：antMatchers 配置认证绕过

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)