Spring Security

xieqiaoyun

Spring Security 提供什么功能？

- ☐ 全面支持身份验证、授权、异常处理
- ☐ 防止常见攻击
- ☐ 可与Servlet API 集成 ← 官方说的
- ☐ 可与Spring MVC 集成
- ☐ 可与Spring data集成
- ☐ 支持websocket

动手练习

- [ ] 1、/manager/** 需要 manager的角色才能访问， /manager/createuser 路径则额外需要 op_createuser的授权才能访问

- [ ] 2、CSRF攻击、XSS攻击、HTTP Strict Transport Security （HSTS） 攻击、X-Frame-Options挟持攻击、Session固定攻击的例子与SPRING secuity的应对之策

- [ ] 3、X-Content-Type-Options这个Header的作用

- [ ] 4、分析Spring Security RememberMe的特性， 实现基于数据库的Rememberme方式

- [ ] 5、cas +spring security的DEMO

- [ ] 6、WebSocket的安全整合

- [ ] 7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

1、/manager/** 需要 manager的角色才能访问，/manager/createuser 路径则额外需要 op_createuser 的授权才能访问

1、/manager/** 需要 manager的角色才能访问，/manager/createuser 路径则额外需要 op_createuser的授权才能访问

没啥好说的。。。

```
@Configuration
@Order(2)
public static class ManageSecurityConfig extends WebSecurityConfigurerAdapter{
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.antMatcher("/manage/api/**")
            .authorizeRequests()
            .antMatchers("/manage/api/**").hasRole("manage")
            .antMatchers("/manage/api/createuser/**")
            .access("hasRole('manage') and hasRole('op_createuser')")
            .antMatchers("/manage/api/**").denyAll()
            .anyRequest().authenticated()
            .and()
            .httpBasic();
        }
}
```

2、CSRF攻击、XSS攻击、HTTP Strict Transport Security （HSTS） 攻击、X-Frame-Options挟持攻击、Session固定攻击的例子与SPRING secuity的应对之策

XSS(跨站脚步攻击)：可执行其不同源的脚本，攻击者可将客户端代码注入到其它用户查看的网页，代表用户执行操作。可通过如下方式注入：

- 通过不安全的html标签<script><a><img><iframe>等注入：…
- 提交不安全的代码到服务端，如提交下面文本到服务端数据库："eval('var a=100;alert(a);')"

常见解决办法：对可执行的标签进行转义

Spring Security解决方案：

```
@Override
protected void configure(HttpSecurity http) throws Exception {
	http
	// ...
	.headers()
		.xssProtection()
			.block(true);
}
```

CSRF(跨站请求伪造)：通过未授权命令，窃取token，伪造请求获取信任；
常用img标签、隐藏表单、XMLHttpRequest对象、iframe等手段进行窃取

常见解决办法：

- ☐ 对可执行标签进行转义
- ☐ 正确使用http请求的方法
- ☐ 设置HttpOnly告知浏览器不在http/https之外暴露，避免通过document.cookie窃取cookie
- ☐ 设置过期时间
- ☐ 提供退出销毁cookie方法

HTTP Strict Transport Security （HSTS）攻击

- http到https重定向时，存在中间人攻击风险，此属性声明网站只能通过安全连接访问。安全连接方式如CA认证、公私钥校验、传输层加密、http公钥密码等

Spring Security配置

```
protected void configure(HttpSecurity http) throws Exception {
    http
    // ...
    .headers()
        .httpStrictTransportSecurity()
            .includeSubdomains(true)
            .maxAgeSeconds(31536000);
}
```

X-Frame-Options挟持攻击

网页被其它站点的iframe嵌入，存在安全隐患。通过设置header的X-Frame-Options属性达到防护效果。

IE8及以上支持

```
X-Frame-Options: DENY  ---------- 拒绝嵌入
X-Frame-Options: SAMEORIGIN ------ 允许被同源的站点嵌入
X-Frame-Options: ALLOW-FROM https://example.com/ -- 允许被指定站点嵌入
```

chrome不支持该选项

Spring Security配置

```
protected void configure(HttpSecurity http) throws Exception {
    http
    // ...
    .headers()
        .frameOptions()
            .sameOrigin();
}
```

Session固定攻击

常见手段：

- ☐ 发送会话标识的连接给好友
- ☐ 利用服务端生成的固定id
- ☐ 利用子站cookie

防护方法：

- ☐ 不接受GET或POST上送会话标识
- ☐ 进行鉴权、授权身份确认
- ☐ 通过HTTP cookie上送会话标识

Spring security通过配置session-fixation-protection属性选择防护措施：

none ——————没有防护

newSession ————重新产生新会话

migrateSession ———创建新会话，复制现有属性

changeSessionId ——使用Servlet容器固定会话保护

3、X-Content-Type-Options这个Heacler的作用

防止MIME类型嗅探或修改，只作用于script和style，貌似某些浏览器对于img标签也起作用

如下设置，当style的MIME type不为"text/css"时，script的MIME type不为将会阻塞：

X-Content-Type-Options: nosniff

- application/ecmascript
- application/javascript
- application/x-ecmascript
- application/x-javascript
- text/ecmascript
- text/javascript
- text/javascript1.0
- text/javascript1.1
- text/javascript1.2
- text/javascript1.3
- text/javascript1.4
- text/javascript1.5
- text/jscript
- text/livescript
- text/x-ecmascript
- text/x-javascript

4、分析Spring Security RememberMe的特性，实现基于数据库的Rememberme方式

没啥好说的，以后再做。。。

5、cas +spring security的DEMO

没啥好说的，以后再做。。。

6、WebSocket的安全整合

没啥好说的，以后再做。。。

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

被template套路了之后，全部理清了，又被MyBatis坑了两天。。。

大概思路如下，请看下页

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

一定要约定好，为了前后端分离，html页面不要拦截，各模块需要登录的接口/api/**进行登录验证

```
@Override
protected void configure(HttpSecurity http) throws Exception{
    http.csrf().disable()
        .antMatcher("/api/**")
        .authorizeRequests()
        .antMatchers("/api/**").denyAll()
        .anyRequest().authenticated()
        .and()
        .httpBasic();

    http.antMatcher("/kingcenter/api/**")
        .authorizeRequests()
        .antMatchers("/kingcenter/api/**").denyAll()
        .antMatchers("/kingcenter/api/{username}").access("@authz.check(#username,pricipal)")
        .anyRequest().authenticated()
        .and().httpBasic();

}
```

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

跳转登录的动作由前端触发，封装请求模块方法

```
function ajax(params){
    var multipart = params && params.multipart?true:false;
    var contentType = 'application/json; charset=utf-8';
    var dataType = 'json';
    var jwt = global.sessionStorage.getItem('imk');
    var options = $.extend({dataType:dataType,contentType:contentType},params);
    if(multipart){
        options.cache = false;
        options.contentType = false;
        options.processData = false;
    }
    options.beforeSend = function(jqXhr,settings){
        var headers = jqXhr.getAllResponseHeaders();
        if(jwt){
            jqXhr.setRequestHeader('Authorization','Imking '+jwt)
        }
        // show loading
        global._loading.show();
    };
    options.complete = function(jqXhr,textStatus){
        // textStatus: "success", "notmodified", "nocontent", "error", "timeout", "abort", or "parsererror"
        // close loading
        console.log('complete',jqXhr,textStatus)
        global._loading.hide();
        if(401 == jqXhr.status){
            document.location.href = '/auth/login.html?redirectUrl='+window.encodeURIComponent(window.location.href)
        }
    };
    options.error = function(jqXhr,status){ // "timeout" "error" "abort" "parsererror"
```

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

点 admin manage kingcenter 调用了**/api/下面的接口，触发跳转到登录页

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

自定义UserDetailsService

```
public class MyUserDetailsService implements UserDetailsService {
    @Autowired
    UserService userService;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userService.getUserByName(username);
        if (user == null) {
            throw new UsernameNotFoundException("Could not find user " + username);
        }
        List<GrantedAuthority> authorities =  userService.getAuthorities(username);
        user.setAuthorities(authorities);
        return new CustomUserDetails(user);
    }

    private final static class CustomUserDetails extends User implements UserDetails{
        private CustomUserDetails(User user) {
            //super(user);
        }

        @Override
        public List<GrantedAuthority> getAuthorities(
//          List<GrantedAuthority> auths = getAuthori
//          return auths;
            return AuthorityUtils.createAuthorityList("ROLE_USER");
        }

        @Override
        public boolean isAccountNonExpired() {
```

被orm弄糊涂了请忽略是否正确，不是宝宝想要的效果

```
@Autowired
MyUserDetailsService userService;

@Override
protected void configure(AuthenticationManagerBuilder auth)
    auth.authenticationProvider(myAuthenticationProvider);
    auth.userDetailsService(new MyUserDetailsService());
}
```

7、用bootstrap +jquery 实现ajax方式的SPRING登录表单

没了，后面想改成不要Mybatis试试。。。

Thank You

— xieqiaoyun