Web 应用程序报告

该报告包含有关 web 应用程序的重要安全信息。

安全报告

该报告由 HCL AppScan Standard 创建 10.0.0, 规则： 0
扫描开始时间： 2024/6/2 1:28:01

目录

- 将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头
- 为 Web 服务器或 Web 应用程序下载相关的安全补丁
- 验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

咨询

- “Content-Security-Policy”头缺失或不安全
- “X-Content-Type-Options”头缺失或不安全
- “X-XSS-Protection”报头缺失或不安全
- 临时文件下载
- 发现可能的服务器路径泄露模式
- 发现内部 IP 泄露模式
- 应用程序错误
- 整数溢出

介绍

该报告包含由 HCL AppScan Standard 执行的 Web 应用程序安全性扫描的结果。

低严重性问题：	16
参考严重性问题：	14
报告中包含的严重性问题总数：	30
扫描中发现的严重性问题总数：	30

常规信息

扫描文件名称： 7

扫描开始时间： 2024/6/2 1:28:01

测试策略： Web Services

主机 localhost

端口 8080

操作系统： 未知

Web 服务器： 未知

应用程序服务器： 任何

主机 localhost

端口 8081

操作系统： 未知

Web 服务器： 未知

应用程序服务器： 任何

登陆设置

登陆方法： 无

摘要

问题类型 8

TOC

	问题类型	问题的数量
低	“Content-Security-Policy”头缺失或不安全	5
低	“X-Content-Type-Options”头缺失或不安全	5
低	“X-XSS-Protection”头缺失或不安全	5
低	临时文件下载	1
参	发现可能的服务器路径泄露模式	1
参	发现内部 IP 泄露模式	2
参	应用程序错误	7
参	整数溢出	4

有漏洞的 URL 10

TOC

	URL	问题的数量
低	http://localhost:8081/mv/personalized	1
低	http://localhost:8081/song/allSong/1/5	3
低	http://localhost:8081/songList/detail-userId/1	3
低	http://localhost:8081/songList/getRandomSongList/4	3
低	http://localhost:8081/user/detail/1	4
低	http://localhost:8081/user/login	4
参	http://localhost:8080/js/app.js	2
参	http://localhost:8080/js/chunk-vendors.js	1
参	http://localhost:8081/listSong/add	4
参	http://localhost:8081/mv/all	5

修订建议 7

TOC

	修复任务	问题的数量
低	除去 Web 站点中的内部 IP 地址	2
低	除去虚拟目录中的旧版本文件	1
低	将服务器配置为使用安全策略的“Content-Security-Policy”头	5
低	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头	5
低	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头	5
低	为 Web 服务器或 Web 应用程序下载相关的安全补丁	1
低	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常	11

安全风险 5

TOC

	风险	问题的数量
低	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置	17
低	可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息	15
低	可能会下载临时脚本文件，这会泄露应用程序逻辑及其他诸如用户名和密码之类的敏感信息	1
参	可能会检索 Web 服务器安装的绝对路径，这可能会帮助攻击者开展进一步攻击和获取有关 Web 应用程序文件系统结构的信息	1
参	可能会收集敏感的调试信息	11

原因 5

TOC

	原因	问题的数量
低	Web 应用程序编程或配置不安全	17
低	在生产环境中留下临时文件	1
参	未安装第三方产品的最新补丁或最新修补程序	1
参	未对入局参数值执行适当的边界检查	11
参	未执行验证以确保用户输入与预期的数据类型匹配	11

WASC 威胁分类

TOC

威胁	问题的数量
可预测资源位置	1
信息泄露	25
整数溢出	4

按问题类型分类的问题

低 “Content-Security-Policy”头缺失或不安全 5 TOC

问题 1 / 5 TOC

“Content-Security-Policy”头缺失或不安全	
严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/user/detail/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用安全策略的“Content-Security-Policy”头

推理： AppScan 检测到 Content-Security-Policy 响应头缺失或具有不安全策略，这可能会更大程度地暴露于各种跨站点注入攻击之下

未经处理的测试响应：

```
HTTP/1.1 204
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8080
Keep-Alive: timeout=60
Access-Control-Allow-Headers: Content-Type,Authorization
Date: Sat, 01 Jun 2024 17:41:23 GMT
Access-Control-Allow-Methods: OPTIONS
Content-Type: application/json;charset=UTF-8...
```

问题 2 / 5 TOC

“Content-Security-Policy”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/song/allSong/1/5
实体：	5 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用安全策略的“Content-Security-Policy”头

推理： AppScan 检测到 Content-Security-Policy 响应头缺失或具有不安全策略，这可能会更大程度地暴露于各种跨站点注入攻击之下

未经处理的测试响应：

```
HTTP/1.1 204
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8080
Keep-Alive: timeout=60
Access-Control-Allow-Headers: Content-Type,Authorization
Date: Sat, 01 Jun 2024 17:41:23 GMT
Access-Control-Allow-Methods: OPTIONS
Content-Type: application/json;charset=UTF-8...
```

问题 3 / 5

TOC

“Content-Security-Policy”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/mv/personalized
实体：	personalized (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用安全策略的“Content-Security-Policy”头

推理： AppScan 检测到 Content-Security-Policy 响应头缺失或具有不安全策略，这可能会更大程度地暴露于各种跨站点注入攻击之下

未经处理的测试响应：

```
HTTP/1.1 204
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8080
```

```
Keep-Alive: timeout=60
Access-Control-Allow-Headers: Content-Type,Authorization
Date: Sat, 01 Jun 2024 17:41:23 GMT
Access-Control-Allow-Methods: OPTIONS
Content-Type: application/json;charset=UTF-8...
```

问题 4 / 5

TOC

“Content-Security-Policy”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/songList/detail-userId/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用安全策略的“Content-Security-Policy”头

推理： AppScan 检测到 Content-Security-Policy 响应头缺失或具有不安全策略，这可能会更大程度地暴露于各种跨站点注入攻击之下

未经处理的测试响应：

```
HTTP/1.1 204
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8080
Keep-Alive: timeout=60
Access-Control-Allow-Headers: Content-Type,Authorization
Date: Sat, 01 Jun 2024 17:41:23 GMT
Access-Control-Allow-Methods: OPTIONS
Content-Type: application/json;charset=UTF-8...
```

问题 5 / 5

TOC

“Content-Security-Policy”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/songList/getRandomSongList/4
实体：	4 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用安全策略的“Content-Security-Policy”头

推理： AppScan 检测到 Content-Security-Policy 响应头缺失或具有不安全策略，这可能会更大程度地暴露于各种跨站点注入攻击之下

未经处理的测试响应：

```
HTTP/1.1 204
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8080
Keep-Alive: timeout=60
Access-Control-Allow-Headers: Content-Type,Authorization
Date: Sat, 01 Jun 2024 17:41:23 GMT
Access-Control-Allow-Methods: OPTIONS
Content-Type: application/json;charset=UTF-8...
```

低 “X-Content-Type-Options”头缺失或不安全 5

TOC

问题 1 / 5

TOC

“X-Content-Type-Options”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/songList/detail-userId/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头

推理： AppScan 检测到“X-Content-Type-Options”响应头缺失或具有不安全值，这可能会更大程度地暴露

于偷渡式下载攻击之下

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 1,
   "userId": 1,
   "title": "abcd",
   "pic": "http://p2.music.126.net/e5cvcdgeosDKTDrkTfZXnQ==/109951166155165682.jpg",
   "introduction": "a song list",

...
```

问题 2 / 5

TOC

“X-Content-Type-Options”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/songList/getRandomSongList/4
实体：	4 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头

推理： AppScan 检测到“X-Content-Type-Options”响应头缺失或具有不安全值，这可能会更大程度地暴露于偷渡式下载攻击之下

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 2,
   "userId": 1,
   "title": "asd",
   "pic": "http://p2.music.126.net/diLAroTRclQuyJW8_WW6aQ==/109951167232406424.jpg",
   "introduction": "asd",

...
```

问题 3 / 5

TOC

“X-Content-Type-Options”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/user/detail/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头

推理： AppScan 检测到“X-Content-Type-Options”响应头缺失或具有不安全值，这可能会更大程度地暴露于偷渡式下载攻击之下

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": {
  "id": 1,
```

```
  "username": "root",
  "nickname": "1",
  "sex": 1,
  "phoneNum": "114514",
  "email": "string",

...
```

问题 4 / 5

TOC

“X-Content-Type-Options”头缺失或不安全	
严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/user/login
实体：	login (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头

推理： AppScan 检测到“X-Content-Type-Options”响应头缺失或具有不安全值，这可能会更大程度地暴露于偷渡式下载攻击之下

未经处理的测试响应：

```
...

Referer: http://localhost:8080/
Connection: keep-alive
Host: localhost:8081
Content-Length: 0
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
```

```
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": {
  "user": {
   "id": 1,
   "username": "root",
   "nickname": "1",
   "sex": 1,
   "phoneNum": "114514",

...
```

问题 5 / 5

TOC

“X-Content-Type-Options”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/song/allSong/1/5
实体：	5 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头

推理： AppScan 检测到“X-Content-Type-Options”响应头缺失或具有不安全值，这可能会更大程度地暴露于偷渡式下载攻击之下

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
```

```
...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 3594,
   "songId": "1934322814",
   "singerId": "189873",
   "pic": "http://p2.music.126.net/diLAroTRclQuyJW8_WW6aQ==/109951167232406424.jpg",
   "lyric": "[00:00.00] 作词 : 南征北战NZBZ\n[00:01.00] 作曲 : 南征北战NZBZ\n[00:02.00] 编曲 : 南征北战NZBZ\n[00:03.00] 吉他 : 冯冲\n[00:04.00] 伴唱 : 陈聆子\n[00:05.00] 贝斯 : 张博\n[00:06.00] 录音师 : 郑晓桐\n[00:07.00] 混音 : 陆怡帆 Kazze\
...
```

低 “X-XSS-Protection”头缺失或不安全 5 TOC

问题 1 / 5 TOC

“X-XSS-Protection”头缺失或不安全	
严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/songList/detail-userId/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头

推理： AppScan 检测到 X-XSS-Protection 响应头缺失或具有不安全值，这可能会造成跨站点脚本编制攻击

未经处理的测试响应：

```
...

Connection: keep-alive
```

```
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 1,
   "userId": 1,
   "title": "abcd",
   "pic": "http://p2.music.126.net/e5cvcdgeosDKTDrkTfZXnQ==/109951166155165682.jpg",
   "introduction": "a song list",

...
```

问题 2 / 5

TOC

“X-XSS-Protection”头缺失或不安全	
严重性:	低
CVSS 分数:	5.0
URL:	http://localhost:8081/songList/getRandomSongList/4
实体:	4 (Page)
风险:	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因:	Web 应用程序编程或配置不安全
固定值:	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头

推理： AppScan 检测到 X-XSS-Protection 响应头缺失或具有不安全值，这可能会造成跨站点脚本编制攻击

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 2,
   "userId": 1,
   "title": "asd",
   "pic": "http://p2.music.126.net/diLAroTRclQuyJW8_WW6aQ==/109951167232406424.jpg",
   "introduction": "asd",

...
```

问题 3 / 5

TOC

“X-XSS-Protection”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/user/detail/1
实体：	1 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头

推理： AppScan 检测到 X-XSS-Protection 响应头缺失或具有不安全值，这可能会造成跨站点脚本编制攻击

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": {
  "id": 1,
  "username": "root",
  "nickname": "1",
  "sex": 1,
  "phoneNum": "114514",
  "email": "string",

...
```

问题 4 / 5

TOC

“X-XSS-Protection”头缺失或不安全

严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/user/login
实体：	login (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头

推理： AppScan 检测到 X-XSS-Protection 响应头缺失或具有不安全值，这可能会造成跨站点脚本编制攻击

未经处理的测试响应：

```
...

Referer: http://localhost:8080/
Connection: keep-alive
Host: localhost:8081
Content-Length: 0
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": {
  "user": {
   "id": 1,
```

```
    "username": "root",
    "nickname": "1",
    "sex": 1,
    "phoneNum": "114514",

...
```

问题 5 / 5

TOC

“X-XSS-Protection”头缺失或不安全	
严重性：	低
CVSS 分数：	5.0
URL：	http://localhost:8081/song/allSong/1/5
实体：	5 (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置 可能会劝说初级用户提供诸如用户名、密码、信用卡号、社会保险号等敏感信息
原因：	Web 应用程序编程或配置不安全
固定值：	将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头

推理： AppScan 检测到 X-XSS-Protection 响应头缺失或具有不安全值，这可能会造成跨站点脚本编制攻击

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:24 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:23 GMT
```

```
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": [
  {
   "id": 3594,
   "songId": "1934322814",
   "singerId": "189873",
   "pic": "http://p2.music.126.net/diLAroTRclQuyJW8_WW6aQ==/109951167232406424.jpg",
   "lyric": "[00:00.00] 作词 : 南征北战NZBZ\n[00:01.00] 作曲 : 南征北战NZBZ\n[00:02.00] 编曲 : 南征北战
NZBZ\n[00:03.00] 吉他 : 冯冲\n[00:04.00] 伴唱 : 陈聆子\n[00:05.00] 贝斯 : 张博\n[00:06.00] 录音师 : 郑
晓桐\n[00:07.00] 混音 : 陆怡帆 Kazze\
...
```

低 临时文件下载 1 TOC

问题 1 / 1 TOC

临时文件下载

严重性:	低
CVSS 分数:	5.0
URL:	http://localhost:8081/user/detail/1
实体:	1 (Page)
风险:	可能会下载临时脚本文件，这会泄露应用程序逻辑及其他诸如用户名和密码之类的敏感信息
原因:	在生产环境中留下临时文件
固定值:	除去虚拟目录中的旧版本文件

推理: 测试尝试检索源代码文件。响应未产生错误且包含非 HTML 内容，表示源代码检索已成功。
未经处理的测试响应:

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY1ODQxLCJ1c2VySWQiOjF9.Cy2-0RZoNcvjoYu0kxuxkS5Sg3wETjTZwqMqXSl6hH0
Accept-Language: en-US


HTTP/1.1 200
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:47 GMT;
```

```
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: keep-alive
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60

...

...

Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Keep-Alive: timeout=60
Date: Sat, 01 Jun 2024 17:41:47 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Access-Control-Allow-Credentials: true

{
 "code": "200",
 "msg": "请求成功",
 "data": {
  "id": 11,
  "username": "user",
  "nickname": "string",
  "sex": 1,
  "phoneNum": "string",
  "email": "string",

...
```

参 发现可能的服务器路径泄露模式 1

TOC

问题 1 / 1

TOC

发现可能的服务器路径泄露模式

严重性:	参考
CVSS 分数:	0.0
URL:	http://localhost:8080/js/app.js
实体:	app.js (Page)
风险:	可能会检索 Web 服务器安装的绝对路径，这可能会帮助攻击者开展进一步攻击和获取有关 Web 应用程序文件系统结构的信息
原因:	未安装第三方产品的最新补丁或最新修补程序
固定值:	为 Web 服务器或 Web 应用程序下载相关的安全补丁

推理: 响应包含服务器上文件的绝对路径和/或文件名。

未经处理的测试响应:

```
...
...a522] {\\n  color: palevioletred;\\n}\\n.list li[data-v-7220a522]:hover::before {\\n  opacity:
1;\\n}\\nul img[data-v-7220a522] {\\n  width: 100%;\\n  border-radius: 5px;\\n  /* opacity: 1;
*/\\n--begin_hig...

...

...
...e {\\n  opacity: 1;\\n}\\n.new-songs .music-img-warp p[data-v-7220a522]::before {\\n  content:
\\\"\\\ea42\\\";\\n  position: absolute;\\n  top: 50%;\\n  left: 50%;\\n  transform: translate(-
50%, -50%);\\n ...

...

...
...22]::before {\\n  opacity: 1;\\n}\\n.mv-img-warp .play[data-v-7220a522]::before {\\n  content:
\\\"\\\ea42\\\";\\n  position: absolute;\\n  top: 50%;\\n  left: 50%;\\n  transform: translate(-
50%, -50%);\\n ...

...

...
..._js__WEBPACK_IMPORTED_MODULE_0___default()));\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-b49680e8] {\\n  list-style:
none;\\n}\\n.musicdetail[data-v-b49680e8] {\\n  display: flex;\\...

...

...
...nction: ease-out;\\n  animation-direction: alternate-reverse;\\n}\\n@keyframes playingIcon-
7e8fa5a6 {\\nfrom {\\n    transform: translate3d(0, 0, 0);\\n}\\nto {\\n ...

...
```

```
...
..._js__WEBPACK_IMPORTED_MODULE_0___default()));\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-550dc494] {\\n  list-style:
none;\\n}\\n.mvdetail[data-v-550dc494] {\\n  display: flex;\\n  ...

...

...
...  );\\n  border-radius: 5px;\\n}\\n.mv-list .icon-play[data-v-550dc494]::before {\\n  content:
\\\"\\\\ea42\\\";\\n  position: absolute;\\n  top: 50%;\\n  left: 45%;\\n  transform: translate(-
50%, -50%);\\n ...

...

...
..._js__WEBPACK_IMPORTED_MODULE_0___default()));\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-d2c6e038] {\\n  list-style:
none;\\n}\\n.new-mvs[data-v-d2c6e038] {\\n  z-index: 12;\\n  max...

...

...
...38]::before {\\n  opacity: 1;\\n}\\n.mv-img-wrap .play[data-v-d2c6e038]::before {\\n  content:
\\\"\\\\ea42\\\";\\n  position: absolute;\\n  top: 50%;\\n  left: 50%;\\n  transform: translate(-
50%, -50%);\\n ...

...

...
...s__WEBPACK_IMPORTED_MODULE_2__[\"default\"]);\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-5ca55b78] {\\n  list-style:
none;\\n}\\n.new-songs[data-v-5ca55b78] {\\n  max-width: 1300px;...

...

...
...0%;\\n  border-radius: 10px;\\n}\\n.img-wrap .iconfont[data-v-5ca55b78]::before {\\n  content:
\\\"\\\\ea42\\\";\\n  position: absolute;\\n  bottom: 12px;\\n  right: 12px;\\n  width: 35px;\\n
height: 35px;\...

...

...
...s__WEBPACK_IMPORTED_MODULE_2__[\"default\"]);\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-14dd7112] {\\n  list-style:
none;\\n}\\n.playlists[data-v-14dd7112] {\\n  z-index: 12;\\n  w...

...

...
...ranslateY(-100%);\\n  transition: 0.5s;\\n}\\n.list li[data-v-14dd7112]::before {\\n  content:
\\\"\\\\ea42\\\";\\n  position: absolute;\\n  bottom: 25px;\\n  right: 5px;\\n  width: 35px;\\n
height: 35px;\\...

...

...
..._js__WEBPACK_IMPORTED_MODULE_0___default()));\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-284217fd] {\\n  list-style:
none;\\n}\\n.playlist-detail[data-v-284217fd] {\\n  width: 1200p...

...

...
...ex;\\n  align-items: center;\\n}\\n.img-wrap .iconfont[data-v-284217fd]::before {\\n  content:
\\\"\\\\ea42\\\";\\n  position: absolute;\\n  bottom: 18px;\\n  right: 18px;\\n  width: 25px;\\n
height: 25px;\...

...

...
...s__WEBPACK_IMPORTED_MODULE_2__[\"default\"]);\n//
Module\n___CSS_LOADER_EXPORT___.push([module.id, \"\\nul[data-v-28a7b29f] {\\n  list-style:
none;\\n}\\n.searchresult[data-v-28a7b29f] {\\n  max-width: 1300...

...
```

```
...
...ranslateY(-100%);\\n  transition: 0.5s;\\n}\\n.list li[data-v-28a7b29f]::before {\\n  content:
\\\"\\\ea42\\\";\\n  position: absolute;\\n  bottom: 45px;\\n  right: 10px;\\n  width: 25px;\\n
height: 25px;\...

...

...
...9f]::before {\\n  opacity: 1;\\n}\\n.mv-img-wrap .play[data-v-28a7b29f]::before {\\n  content:
\\\"\\\ea42\\\";\\n  position: absolute;\\n  top: 50%;\\n  left: 50%;\\n  transform: translate(-
50%, -50%);\\n ...

...

...
..._js__WEBPACK_IMPORTED_MODULE_0___default()));\n// Module\n___CSS_LOADER_EXPORT___.pus...
```

参 发现内部 IP 泄露模式 2 TOC

问题 1 / 2 TOC

发现内部 IP 泄露模式	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8080/js/app.js
实体：	app.js (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置
原因：	Web 应用程序编程或配置不安全
固定值：	除去 Web 站点中的内部 IP 地址

推理： AppScan 在响应中发现了看似为内部 IP 地址的内容。
未经处理的测试响应：

```
...

/******/   chunkLoadingGlobal.push = webpackJsonpCallback.bind(null,
chunkLoadingGlobal.push.bind(chunkLoadingGlobal));
/******/  }();
/******/
/************************************************************************/
/******/
/******/  // module cache are used so entry inlining is disabled
/******/  // startup
/******/  // Load entry module and return exports
/******/  __webpack_require__.O(undefined, ["chunk-vendors"], function() { return
__webpack_require__("./node_modules/whatwg-fetch/fetch.js"); })
/******/  __webpack_require__.O(undefined, ["chunk-vendors"], function() { return
__webpack_require__("./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22%3Afalse%7D&reconnect=10&hot=true&live-reload=true"); })
/******/  __webpack_require__.O(undefined, ["chunk-vendors"], function() { return
```

```
__webpack_require__("./node_modules/webpack/hot/dev-server.js"); })
/******/  var __webpack_exports__ = __webpack_require__.O(undefined, ["chunk-vendors"],
function() { return __webpack_require__("./src/main.js"); })
/******/  __webpack_exports__ = __webpack_require__.O(__webpack_exports__);
/******/
/******/ })()
;
...
```

问题 2 / 2

TOC

发现内部 IP 泄露模式

严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8080/js/chunk-vendors.js
实体：	chunk-vendors.js (Page)
风险：	可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置
原因：	Web 应用程序编程或配置不安全
固定值：	除去 Web 站点中的内部 IP 地址

推理： AppScan 在响应中发现了看似为内部 IP 地址的内容。

未经处理的测试响应：

```
...

/***/ "./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22%3Afalse%7D&reconnect=10&hot=true&live-reload=true":
/*!**********************************************************************************************
**********************************************************************************************
************************************************!*\
  !*** ./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22%3Afalse%7D&reconnect=10&hot=true&live-reload=true ***!

\**********************************************************************************************
**********************************************************************************************
************************************************/
/***/ (function(__unused_webpack_module, __webpack_exports__, __webpack_require__) {
...

...

/***/ "./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22%3Afalse%7D&reconnect=10&hot=true&live-reload=true":
/*!**********************************************************************************************
**********************************************************************************************
************************************************!*\
  !*** ./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22%3Afalse%7D&reconnect=10&hot=true&live-reload=true ***!

\**********************************************************************************************
**********************************************************************************************
************************************************/
/***/ (function(__unused_webpack_module, __webpack_exports__, __webpack_require__) {
```

```
...

...

"use strict";
eval("var __resourceQuery = \"?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%2...

...

...
...ourceURL=webpack-internal:///./node_modules/webpack-dev-server/client/index.js?
protocol=ws&hostname=192.168.50.213&port=8080&pathname=%2Fws&logging=none&progress=true&overlay=%
7B%22errors%22%3Atrue%2C%22warnings%22...

...
```

参 应用程序错误 7 TOC

问题 1 / 7 TOC

应用程序错误	
严重性:	参考
CVSS 分数:	0.0
URL:	http://localhost:8081/user/login
实体:	username (Parameter)
风险:	可能会收集敏感的调试信息
原因:	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值:	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理: 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。
未经处理的测试响应:

```
...

Referer: http://localhost:8080/
Connection: keep-alive
Host: localhost:8081
Content-Length: 0
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
```

```
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:32 GMT
Content-Type: application/json;charset=UTF-8

...
```

问题 2 / 7

TOC

应用程序错误	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/user/login
实体：	password (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Referer: http://localhost:8080/
Connection: keep-alive
Host: localhost:8081
Content-Length: 0
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:32 GMT
Content-Type: application/json;charset=UTF-8

...
```

问题 3 / 7

TOC

应用程序错误	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/mv/all
实体：	order (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:33 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

问题 4 / 7

TOC

应用程序错误

严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/mv/all
实体：	pageNo (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:33 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

应用程序错误

严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/listSong/add
实体：	->"songId" (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US
Content-Type: application/json;charset=UTF-8

{
 "songId": "'",
 "songListId": 2
}

HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:33 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

应用程序错误

严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/mv/all
实体：	pageSize (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

问题 7 / 7

TOC

应用程序错误

严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/listSong/add
实体：	->"songListId" (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US
Content-Type: application/json;charset=UTF-8

{
 "songId": 3594,
 "songListId": "'"
}

HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

参 整数溢出 4 TOC

问题 1 / 4 TOC

整数溢出	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/listSong/add
实体：	->"songId" (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US
Content-Type: application/json;charset=UTF-8

{
 "songId": 99999999999999999999,
 "songListId": 2
}

HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

问题 2 / 4

TOC

整数溢出	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/mv/all
实体：	pageNo (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

问题 3 / 4

TOC

整数溢出	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/mv/all
实体：	pageSize (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...

Connection: keep-alive
Host: localhost:8081
Accept: application/json, text/plain, */*
Origin: http://localhost:8080
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US


HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: GET
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

问题 4 / 4

TOC

整数溢出	
严重性：	参考
CVSS 分数：	0.0
URL：	http://localhost:8081/listSong/add
实体：	->"songListId" (Parameter)
风险：	可能会收集敏感的调试信息
原因：	未对入局参数值执行适当的边界检查 未执行验证以确保用户输入与预期的数据类型匹配
固定值：	验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常

推理： 应用程序以错误消息响应，表示可能会泄露敏感信息的未定义状态。

未经处理的测试响应：

```
...
Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNzaW9ucyI6WyJyb290IiwiYWRtaW4iLCJ1c2VyIl0sInJvbGVz
IjpbInJvb3QiXSwiZXhwIjoxNzE3ODY4MzUxLCJ1c2VySWQiOjF9.ykkkO5qHgqS6j_LRI9s9eEZaJUyaQFJ5dorUb1YlGoM
Accept-Language: en-US
Content-Type: application/json;charset=UTF-8

{
 "songId": 3594,
 "songListId": 9999999999999999999
}

HTTP/1.1 500
Access-Control-Allow-Headers: Content-Type,Authorization
Access-Control-Allow-Methods: POST
Set-Cookie: rememberMe=deleteMe; Path=/; Max-Age=0; Expires=Fri, 31-May-2024 17:41:34 GMT;
SameSite=lax
Access-Control-Allow-Origin: http://localhost:8080
Connection: close
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Sat, 01 Jun 2024 17:41:33 GMT

...
```

修订建议

低 除去 Web 站点中的内部 IP 地址 TOC

该任务修复的问题类型

- 发现内部 IP 泄露模式

常规

内部 IP 通常显现在 Web 应用程序/服务器所生成的错误消息中，或显现在 HTML/JavaScript 注释中。
[1] 关闭 Web 应用程序/服务器中有问题的详细错误消息。
[2] 确保已安装相关的补丁。
[3] 确保内部 IP 信息未留在 HTML/JavaScript 注释中。

低 除去虚拟目录中的旧版本文件 TOC

该任务修复的问题类型

- 临时文件下载

常规

请勿将文件的备份/暂存版本归档在虚拟 Web 服务器根目录之下。 这通常在编辑器“就地”编辑这些文件之时发生。 相反地，当升级站点时，请将文件移动或复制到虚拟根目录以外的目录、在这个目录中编辑文件，然后再将文件移动（或复制）回虚拟根目录。 请确保，在虚拟根目录下，只有实际在使用的文件。

低 将服务器配置为使用安全策略的“Content-Security-Policy”头 TOC

该任务修复的问题类型

- “Content-Security-Policy”头缺失或不安全

常规

将服务器配置为发送“Content-Security-Policy”头。
关于 Apache，请参阅：
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
关于 IIS，请参阅：
https://technet.microsoft.com/pl-pl/library/cc753133%28v=ws.10%29.aspx
关于 nginx，请参阅：
http://nginx.org/en/docs/http/ngx_http_headers_module.html

低 将服务器配置为使用值为“1”（已启用）的“X-XSS-Protection”头 TOC

该任务修复的问题类型

- “X-XSS-Protection”头缺失或不安全

常规

配置您的服务器，以确保在所有传出请求上发送值为“1”（即已启用）的“X-XSS-Protection”报头。

关于 Apache，请参阅：
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
关于 IIS，请参阅：
https://technet.microsoft.com/pl-pl/library/cc753133%28v=ws.10%29.aspx
关于 nginx，请参阅：
http://nginx.org/en/docs/http/ngx_http_headers_module.html

低 将服务器配置为使用值为“nosniff”的“X-Content-Type-Options”头 TOC

该任务修复的问题类型

- “X-Content-Type-Options”头缺失或不安全

常规

将服务器配置为针对所有外发请求发送具有值“nosniff”的“X-Content-Type-Options”头。

关于 Apache，请参阅：
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
关于 IIS，请参阅：
https://technet.microsoft.com/pl-pl/library/cc753133%28v=ws.10%29.aspx
关于 nginx，请参阅：
http://nginx.org/en/docs/http/ngx_http_headers_module.html

低 为 Web 服务器或 Web 应用程序下载相关的安全补丁 TOC

该任务修复的问题类型

- 发现可能的服务器路径泄露模式

常规

有几种缓解技术：
[1] 如果漏洞存在于应用程序内，请修复服务器代码，以使得任何输出中都不包含文件位置。
[2] 否则，如果应用程序位于第三方产品中，请根据 Web 服务器或 Web 应用程序上使用的第三方产品下载相关的安全补丁。

低 验证参数值是否在其预计范围和类型内。不要输出调试错误消息和异常 TOC

该任务修复的问题类型

- 应用程序错误
- 整数溢出

常规

应用程序错误

[1] 检查入局请求，以了解所有预期的参数和值是否存在。 当参数缺失时，发出适当的错误消息，或使用缺省值。
[2] 应用程序应验证其输入是否由有效字符组成（解码后）。 例如，应拒绝包含空字节（编码为 %00）、单引号、引号等的输入值。
[3] 确保值符合预期范围和类型。 如果应用程序预期特定参数具有特定集合中的值，那么该应用程序应确保其接收的值确实属于该集合。 例如，如果应用程序预期值在 10..99 范围内，那么就该确保该值确实是数字，且在 10..99 范围内。

[4] 验证数据是否属于提供给客户端的集合。
[5] 请勿在生产环境中输出调试错误消息和异常。

整数溢出

[1] 检查入局请求，以了解所有预期的参数和值是否存在。 当参数缺失时，发出适当的错误消息，或使用缺省值。
[2] 应用程序应验证其输入是否由有效字符组成（解码后）。 例如，应拒绝包含空字节（编码为 %00）、单引号、引号等的输入值。
[3] 确保值符合预期范围和类型。 如果应用程序预期特定参数具有特定集合中的值，那么该应用程序应确保其接收的值确实属于该集合。 例如，如果应用程序预期值在 10..99 范围内，那么就该确保该值确实是数字，且在 10..99 范围内。
[4] 验证数据是否属于提供给客户端的集合。
[5] 请勿在生产环境中输出调试错误消息和异常。

.Net

应用程序错误

要在 ASP.NET 中禁用调试，请编辑 web.config 文件，使其包含以下属性：

```
<compilation
    debug="false"
/>
```

要获取更多信息，请参阅“HOW TO: Disable Debugging for ASP.NET Applications”，位置如下：
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

您可以使用验证控件，将输入验证添加到“Web 表单”页面。验证控件提供适用于所有常见类型的标准验证的易用机制（例如，测试验证日期是否有效，或验证值是否在范围内），以及进行定制编写验证的方法。此外，验证控件还使您能够完整定制向用户显示错误信息的方式。验证控件可搭配“Web 表单”页面的类文件中处理的任何控件使用，其中包括 HTML 和 Web 服务器控件。
要确保所有的必需参数都存在于请求中，请使用 "RequiredFieldValidator" 验证控件。该控件确保用户不会跳过 web 表单中的任何条目。
要确保用户输入仅包含有效值，您可以使用以下验证控件中的一种：
[1] “RangeValidator”：检查用户条目（值）是否在指定的上下界限之间。您可以检查配对数字、字母字符和日期内的范围。
[2] “RegularExpressionValidator”：检查条目是否与正则表达式定义的模式相匹配。此类型的验证使您能够检查可预见的字符序列，如社会保险号码、电子邮件地址、电话号码、邮政编码等中的字符序列。
重要注意事项：验证控件不会阻止用户输入或更改页面处理流程；它们只会设置错误状态，并产生错误消息。程序员的职责是，在执行进一步的应用程序特定操作前，测试代码中控件的状态。
有两种方法可检查用户输入的有效性：
1. 测试常规错误状态： 在您的代码中，测试页面的 IsValid 属性。该属性会将页面上所有验证控件的 IsValid 属性值汇总（使用逻辑 AND）。如果将其中一个验证控件设置为无效，那么页面属性将会返回 false。
2. 测试个别控件的错误状态：
在页面的“验证器”集合中循环，该集合包含对所有验证控件的引用。然后，您就可以检查每个验证控件的 IsValid 属性。

整数溢出

要在 ASP.NET 中禁用调试，请编辑 web.config 文件，使其包含以下属性：

```
<compilation
    debug="false"
/>
```

要获取更多信息，请参阅“HOW TO: Disable Debugging for ASP.NET Applications”，位置如下：
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

您可以使用验证控件，将输入验证添加到“Web 表单”页面。验证控件提供适用于所有常见类型的标准验证的易用机制（例如，测试验证日期是否有效，或验证值是否在范围内），以及进行定制编写验证的方法。此外，验证控件还使您能够完整定制向用户显示错误信息的方式。验证控件可搭配“Web 表单”页面的类文件中处理的任何控件使用，其中包括

HTML 和 Web 服务器控件。
要确保所有的必需参数都存在于请求中，请使用 "RequiredFieldValidator" 验证控件。该控件确保用户不会跳过 web 表单中的任何条目。
要确保用户输入仅包含有效值，您可以使用以下验证控件中的一种：
[1] “RangeValidator”：检查用户条目（值）是否在指定的上下界限之间。您可以检查配对数字、字母字符和日期内的范围。
[2] “RegularExpressionValidator”：检查条目是否与正则表达式定义的模式相匹配。此类型的验证使您能够检查可预见的字符序列，如社会保险号码、电子邮件地址、电话号码、邮政编码等中的字符序列。
重要注意事项：验证控件不会阻止用户输入或更改页面处理流程；它们只会设置错误状态，并产生错误消息。程序员的职责是，在执行进一步的应用程序特定操作前，测试代码中控件的状态。
有两种方法可检查用户输入的有效性：
1. 测试常规错误状态： 在您的代码中，测试页面的 IsValid 属性。该属性会将页面上所有验证控件的 IsValid 属性值汇总（使用逻辑 AND）。如果将其中一个验证控件设置为无效，那么页面属性将会返回 false。
2. 测试个别控件的错误状态：
在页面的“验证器”集合中循环，该集合包含对所有验证控件的引用。然后，您就可以检查每个验证控件的 IsValid 属性。

J2EE

应用程序错误

** 输入数据验证：** 输入数据验证：虽然为方便用户而在客户端层上提供数据验证，但仍必须使用 Servlet 在服务器层上执行数据验证。客户端验证本身就不安全，因为这些验证可轻易绕过，例如，通过禁用 Javascript。
一份好的设计通常需要 Web 应用程序框架，以提供服务器端实用程序例程，从而验证以下内容：
[1] 必需字段
[2] 字段数据类型（缺省情况下，所有 HTTP 请求参数都是“字符串”）
[3] 字段长度
[4] 字段范围
[5] 字段选项
[6] 字段模式
[7] cookie 值
[8] HTTP 响应好的做法是将以上例程作为“验证器”实用程序类中的静态方法实现。以下部分描述验证器类的一个示例。
[1] 必需字段“始终”检查字段不为空，并且其长度要大于零，不包括行距和后面的空格。如何验证必需字段的示例：

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
        isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] 输入的 Web 应用程序中的字段数据类型和输入参数欠佳。例如，所有 HTTP 请求参数或 cookie 值的类型都是“字符串”。开发者负责验证输入的数据类型是否正确。使用 Java 基本包装程序类，来检查是否可将字段值安全地转换为所需的基本数据类型。
验证数字字段（int 类型）的方式的示例：

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
        Integer.parseInt(value);
        isFieldValid = true;
        } catch (Exception e) {
        isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

好的做法是将所有 HTTP 请求参数转换为其各自的数据类型。例如，将请求参数的“integerValue”存储在请求属性中，并按以下示例所示来使用：

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

应用程序应处理的主要 Java 数据类型：
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] 字段长度“始终”确保输入参数（HTTP 请求参数或 cookie 值）有最小长度和/或最大长度的限制。以下示例验证 userName 字段的长度是否在 8 至 20 个字符之间：

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
        validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
        validatedValue.length() <= maxLength);
    }
```

```
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] 字段范围
始终确保输入参数是在由功能需求定义的范围内。
以下示例验证输入 numberOfChoices 是否在 10 至 20 之间：

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
        // numberOfChoices is valid, continue processing request
        ...
        }
    }
}
```

[5] 字段选项 Web 应用程序通常会为用户显示一组可供选择的选项（例如，使用 SELECT HTML 标记），但不能执行服务器端验证以确保选定的值是其中一个允许的选项。请记住，恶意用户能够轻易修改任何选项值。始终针对由功能需求定义的受允许的选项来验证选定的用户值。以下示例验证用户针对允许的选项列表进行的选择：

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
        List list = Arrays.asList(options);
        if (list != null) {
        isValidValue = list.contains(value);
        }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] 字段模式
始终检查用户输入与由功能需求定义的模式是否匹配。例如，如果 userName 字段应仅允许字母数字字符，且不区分大小写，那么请使用以下正则表达式：^[a-zA-Z0-9]*$

Java 1.3 或更早的版本不包含任何正则表达式包。建议将“Apache 正则表达式包”（请参阅以下“资源”）与 Java 1.3 一起使用，以解决该缺乏支持的问题。执行正则表达式验证的示例：

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
        RE r = new RE(expression);
        match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 引进了一种新的正则表达式包 (java.util.regex)。以下是使用新的 Java 1.4 正则表达式包的 Validator.matchPattern 修订版：

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
        match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] cookie 值使用 javax.servlet.http.Cookie 对象来验证 cookie 值。适用于 cookie 值的相同的验证规则（如上所述）取决于应用程序需求（如验证必需值、验证长度等）。验证必需 cookie 值的示例：

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
```

```
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
        // validate the cookie value
        if (Validator.validateRequired(cookies[i].getValue()) {
        // valid cookie value, continue processing request
        ...
        }
        }
    }
}
```

[8] HTTP 响应
[8-1] 过滤用户输入要保护应用程序免遭跨站点脚本编制的攻击，请通过将敏感字符转换为其对应的字符实体来清理 HTML。这些是 HTML 敏感字符：< > " ' % ;) (& +

以下示例通过将敏感字符转换为其对应的字符实体来过滤指定字符串：

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
        return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
        switch (value.charAt(i)) {
        case '<':
        result.append("&lt;");
        break;
        case '>':
        result.append("&gt;");
        break;
        case '"':
        result.append("&quot;");
        break;
        case '\'':
        result.append("&#39;");
        break;
        case '%':
        result.append("&#37;");
        break;
        case ';':
        result.append("&#59;");
        break;
        case '(':
        result.append("&#40;");
        break;
        case ')':
        result.append("&#41;");
        break;
        case '&':
        result.append("&amp;");
        break;
        case '+':
        result.append("&#43;");
        break;
        default:
        result.append(value.charAt(i));
        break;
        }
        return result;
    }
    ...
}
...
// Filter the HTTP response using Validator.filter
PrintWriter out = response.getWriter();
// set output response
out.write(Validator.filter(response));
```

```
    out.close();
```

Java Servlet API 2.3 引进了“过滤器”，它支持拦截和转换 HTTP 请求或响应。
以下示例使用 Validator.filter 来用“Servlet 过滤器”清理响应：

```
  // Example to filter all sensitive characters in the HTTP response using a Java Filter.
  // This example is for illustration purposes since it will filter all content in the response, including
HTML tags!
  public class SensitiveCharsFilter implements Filter {
      ...
      public void doFilter(ServletRequest request,
          ServletResponse response,
          FilterChain chain)
          throws IOException, ServletException {

          PrintWriter out = response.getWriter();
          ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
          chain.doFilter(request, wrapper);

          CharArrayWriter caw = new CharArrayWriter();
          caw.write(Validator.filter(wrapper.toString()));

          response.setContentType("text/html");
          response.setContentLength(caw.toString().length());
          out.write(caw.toString());
          out.close();
      }
      ...
      public class CharResponseWrapper extends HttpServletResponseWrapper {
          private CharArrayWriter output;

          public String toString() {
          return output.toString();
          }

          public CharResponseWrapper(HttpServletResponse response){
          super(response);
          output = new CharArrayWriter();
          }

          public PrintWriter getWriter(){
          return new PrintWriter(output);
          }
      }
  }

  }
```

[8-2] 保护 cookie
在 cookie 中存储敏感数据时，确保使用 Cookie.setSecure（布尔标志）在 HTTP 响应中设置 cookie 的安全标志，以指导浏览器使用安全协议（如 HTTPS 或 SSL）发送 cookie。
保护“用户”cookie 的示例：

```
  // Example to secure a cookie, i.e. instruct the browser to
  // send the cookie using a secure protocol
  Cookie cookie = new Cookie("user", "sensitive");
  cookie.setSecure(true);
  response.addCookie(cookie);
```

推荐使用的 JAVA 工具用于服务器端验证的两个主要 Java 框架是：
[1] Jakarta Commons Validator（与 Struts 1.1 集成）Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。Jakarta Commons Validator 是一种强大的框架，用来实现所有以上数据验证需求。这些规则配置在定义表单字段的输入验证规则的 XML 文件中。在缺省情况下，Struts 支持在使用 Struts“bean:write”标记撰写的所有数据上，过滤 [8] HTTP 响应中输出的危险字符。可通过设置“filter=false”标志来禁用该过滤。
Struts 定义以下基本输入验证器，但也可定义定制的验证器：
required：如果字段包含空格以外的任何字符，便告成功。
mask：如果值与掩码属性给定的正则表达式相匹配，便告成功。
range：如果值在 min 和 max 属性给定的值的范围内（(value >= min) & (value <= max)），便告成功。
maxLength：如果字段长度小于或等于 max 属性，便告成功。
minLength：如果字段长度大于或等于 min 属性，便告成功。
byte、short、integer、long、float、double：如果可将值转换为对应的基本类型，便告成功。
date：如果值代表有效日期，便告成功。可能会提供日期模式。
creditCard：如果值可以是有效的信用卡号码，便告成功。
e-mail：如果值可以是有效的电子邮件地址，便告成功。
使用“Struts 验证器”来验证 loginForm 的 userName 字段的示例：

```
<form-validation>
    <global>
        ...
        <validator name="required"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateRequired"
        msg="errors.required">
        </validator>
        <validator name="mask"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateMask"
        msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
        <!-- userName is required and is alpha-numeric case insensitive -->
        <field property="userName" depends="required,mask">
        <!-- message resource key to display if validation fails -->
        <msg name="mask" key="login.userName.maskmsg"/>
        <arg0 key="login.userName.displayname"/>
        <var>
        <var-name>mask</var-name>
        <var-value>^[a-zA-Z0-9]*$</var-value>
        </var>
        </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

[2] JavaServer Faces 技术
“JavaServer Faces 技术”是一组代表 UI 组件、管理组件状态、处理事件和输入验证的 Java API (JSR 127)。
JavaServer Faces API 实现以下基本验证器，但可定义定制的验证器： validate_doublerange：在组件上注册 DoubleRangeValidator
validate_length：在组件上注册 LengthValidator
validate_longrange：在组件上注册 LongRangeValidator
validate_required：在组件上注册 RequiredValidator
validate_stringrange：在组件上注册 StringRangeValidator
validator：在组件上注册定制的 Validator

JavaServer Faces API 定义以下 UIInput 和 UIOutput 处理器（标记）：
input_date：接受以 java.text.Date 实例格式化的 java.util.Date
output_date：显示以 java.text.Date 实例格式化的 java.util.Date
input_datetime：接受以 java.text.DateTime 实例格式化的 java.util.Date

output_datetime：显示以 java.text.DateTime 实例格式化的 java.util.Date
input_number：显示以 java.text.NumberFormat 格式化的数字数据类型（java.lang.Number 或基本类型）
output_number：显示以 java.text.NumberFormat 格式化的数字数据类型（java.lang.Number 或基本类型）
input_text：接受单行文本字符串。
output_text：显示单行文本字符串。
input_time：接受以 java.text.DateFormat 时间实例格式化的 java.util.Date
output_time：显示以 java.text.DateFormat 时间实例格式化的 java.util.Date
input_hidden：允许页面作者在页面中包括隐藏变量
input_secret：接受不含空格的单行文本，并在输入时，将其显示为一组星号
input_textarea：接受多行文本
output_errors：显示整个页面的错误消息，或与指定的客户端标识相关联的错误消息
output_label：将嵌套的组件显示为指定输入字段的标签
output_message：显示本地化消息

使用 JavaServer Faces 来验证 loginForm 的 userName 字段的示例：

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use_faces>
```

引用
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java 正则表达式包 —
http://jakarta.apache.org/regexp/
Jakarta 验证器 —
http://jakarta.apache.org/commons/validator/
JavaServer Faces 技术 —
http://java.sun.com/j2ee/javaserverfaces/
** 错误处理：
许多 J2EE Web 应用程序体系结构都遵循“模型视图控制器（MVC）”模式。在该模式中，Servlet 扮演“控制器”的角色。Servlet 将应用程序处理委派给 EJB 会话 Bean（模型）之类的 JavaBean。然后，Servlet 再将请求转发给 JSP（视图），以呈现处理结果。Servlet 应检查所有的输入、输出、返回码、错误代码和已知的异常，以确保实际处理按预期进行。
数据验证可保护应用程序免遭恶意数据篡改，而有效的错误处理策略则是防止应用程序意外泄露内部错误消息（如异常堆栈跟踪）所不可或缺的。好的错误处理策略会处理以下项：
[1] 定义错误
[2] 报告错误
[3] 呈现错误
[4] 错误映射
[1] 定义错误
应避免在应用程序层（如 Servlet）中硬编码错误消息。 相反地，应用程序应该使用映射到已知应用程序故障的错误密

钥。好的做法是定义错误密钥，且该错误密钥映射到 HTML 表单字段或其他 Bean 属性的验证规则。例如，如果需要"user_name"字段，其内容为字母数字，并且必须在数据库中是唯一的，那么就应定义以下错误密钥：

(a) ERROR_USERNAME_REQUIRED：该错误密钥用于显示消息，以通知用户需要 "user_name" 字段；
(b) ERROR_USERNAME_ALPHANUMERIC：该错误密钥用于显示消息，以通知用户 "user_name" 字段应该是字母数字；
(c) ERROR_USERNAME_DUPLICATE：该错误密钥用于显示消息，以通知用户 "user_name" 值在数据库中重复；
(d) ERROR_USERNAME_INVALID：该错误密钥用于显示一般消息，以通知用户 "user_name" 值无效；

好的做法是定义用于存储和报告应用程序错误的以下框架 Java 类：
- ErrorKeys：定义所有错误密钥

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error：封装个别错误

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
    this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
    this.key = key;
    this.values = values;
    }

    // Returns the error key
    public String getKey() {
    return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
    return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors：封装错误的集合

```
// Example: Errors encapsulates the Error objects being reported to the presentation layer.
```

```
    // Errors are stored in a HashMap where the key is the bean property name and value is an
    // ArrayList of Error objects.
    public Class Errors implements Serializable {

        // Adds an Error object to the Collection of errors for the specified bean property.
        public void addError(String property, Error error) {
        ArrayList propertyErrors = (ArrayList)errors.get(property);
        if (propertyErrors == null) {
        propertyErrors = new ArrayList();
        errors.put(property, propertyErrors);
        }
        propertyErrors.put(error);
        }

        // Returns true if there are any errors
        public boolean hasErrors() {
        return (errors.size > 0);
        }

        // Returns the Errors for the specified property
        public ArrayList getErrors(String property) {
        return (ArrayList)errors.get(property);
        }

        private HashMap errors = new HashMap();
    }
```

以下是使用上述框架类来处理“user_name”字段验证错误的示例：

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] 报告错误
有两种方法可报告 web 层应用程序错误：
(a) Servlet 错误机制
(b) JSP 错误机制

[2-a] Servlet 错误机制

Servlet 可通过以下方式报告错误：
- 转发给输入 JSP（已将错误存储在请求属性中），或
- 使用 HTTP 错误代码参数来调用 response.sendError，或
- 抛出异常

好的做法是处理所有已知应用程序错误（如 [1] 部分所述），将这些错误存储在请求属性中，然后转发给输入 JSP。输入 JSP 应显示错误消息，并提示用户重新输入数据。以下示例阐明转发给输入 JSP（userInput.jsp）的方式：

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

如果 Servlet 无法转发给已知的 JSP 页面，那么第二个选项是使用 response.sendError 方法，将 HttpServletResponse.SC_INTERNAL_SERVER_ERROR（状态码 500）作为参数，来报告错误。请参阅 javax.servlet.http.HttpServletResponse 的 Javadoc，以获取有关各种 HTTP 状态码的更多详细信息。返回 HTTP 错误的示例：

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

作为最后的手段，Servlet 可以抛出异常，且该异常必须是以下其中一类的子类：
- RuntimeException
- ServletException
- IOException

[2-b] JSP 错误机制
JSP 页面通过定义 errorPage 伪指令来提供机制，以处理运行时异常，如以下示例所示：

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

未捕获的 JSP 异常被转发给指定的 errorPage，并且原始异常设置在名称为 javax.servlet.jsp.jspException 的请求参数中。错误页面必须包括 isErrorPage 伪指令，如下所示：

```
<%@ page isErrorPage="true" %>
```

isErrorPage 伪指令导致“exception”变量初始化为所抛出的异常对象。
[3] 呈现错误
J2SE Internationalization API 提供使应用程序资源外部化以及将消息格式化的实用程序类，其中包括：

(a) 资源束

(b) 消息格式化

[3-a] 资源束

资源束通过将本地化数据从使用该数据的源代码中分离来支持国际化。资源束通过将本地化数据从使用该数据的源代码中分离来支持国际化。每一资源束都会为特定的语言环境存储键/值对的映射。

java.util.PropertyResourceBundle 将内容存储在外部属性文件中，对其进行使用或扩展都很常见，如以下示例所示：

```
##############################################
# ErrorMessages.properties
##############################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

可定义多种资源，以支持不同的语言环境（因此名为资源束）。例如，可定义 ErrorMessages_fr.properties 以支持该束系列的法语成员。如果请求的语言环境的资源成员不存在，那么会使用缺省成员。在以上示例中，缺省资源是 ErrorMessages.properties。应用程序（JSP 或 Servlet）会根据用户的语言环境从适当的资源检索内容。

[3-b] 消息格式化

J2SE 标准类 java.util.MessageFormat 提供使用替换占位符来创建消息的常规方法。MessageFormat 对象包含嵌入了格式说明符的模式字符串，如下所示：

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

以下是使用 ResourceBundle 和 MessageFormat 来呈现错误消息的更加全面的示例：

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }

    // Returns the error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Locale locale) {
        return getErrorMessage(errorKey, null, locale);
    }

    // Returns a formatted error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
        // Get localized ErrorMessageResource
        ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
        // Get localized error message
        String errorMessage = errorMessageResource.getString(errorKey);
        if (args != null) {
        // Format the message using the specified placeholders args
        return MessageFormat.format(errorMessage, args);
        } else {
```

```
            return errorMessage;
            }
        }

        // default environment locale
        private Locale defaultLocale = Locale.getDefaultLocale();
    }
    ...
    // Get the user's locale
    Locale userLocale = request.getLocale();
    // Check if there were any validation errors
    Errors errors = (Errors)request.getAttribute("errors");
    if (errors != null && errors.hasErrors()) {
        // iterate through errors and output error messages corresponding to the "user_name" property
        ArrayList userNameErrors = errors.getErrors("user_name");
        ListIterator iterator = userNameErrors.iterator();
        while (iterator.hasNext()) {
            // Get the next error object
            Error error = (Error)iterator.next();
            String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
            output.write(errorMessage + "\r\n");
        }
    }
```

建议定义定制 JSP 标记（如 displayErrors），以迭代处理并呈现错误消息，如以上示例所示。
[4] 错误映射
通常情况下，“Servlet 容器”会返回与响应状态码或异常相对应的缺省错误页面。可以使用定制错误页面来指定状态码或异常与 Web 资源之间的映射。好的做法是开发不会泄露内部错误状态的静态错误页面（缺省情况下，大部分 Servlet 容器都会报告内部错误消息）。该映射配置在“Web 部署描述符（web.xml）”中，如以下示例所指定：

```
    <!-- Mapping of HTTP error codes and application exceptions to error pages -->
    <error-page>
      <exception-type>UserValidationException</exception-type>
      <location>/errors/validationError.html</error-page>
    </error-page>
    <error-page>
      <error-code>500</exception-type>
      <location>/errors/internalError.html</error-page>
    </error-page>
    <error-page>
    ...
    </error-page>
    ...
```

推荐使用的 JAVA 工具用于服务器端验证的两个主要 Java 框架是：
[1] Jakarta Commons Validator（与 Struts 1.1 集成）Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。验证规则配置在 XML 文件中，该文件定义了表单字段的输入验证规则以及对应的验证错误密钥。Struts 提供国际化支持以使用资源束和消息格式化来构建本地化应用程序。
使用“Struts 验证器”来验证 loginForm 的 userName 字段的示例：

```
    <form-validation>
        <global>
            ...
            <validator name="required"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateRequired"
            msg="errors.required">
            </validator>
            <validator name="mask"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateMask"
```

```
            msg="errors.invalid">
            </validator>
            ...
        </global>
        <formset>
            <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
            <!-- message resource key to display if validation fails -->
            <msg name="mask" key="login.userName.maskmsg"/>
            <arg0 key="login.userName.displayname"/>
            <var>
            <var-name>mask</var-name>
            <var-value>^[a-zA-Z0-9]*$</var-value>
            </var>
            </field>
            ...
            </form>
            ...
        </formset>
    </form-validation>
```

Struts JSP 标记库定义了有条件地显示一组累计错误消息的“errors”标记，如以下示例所示：

```
    <%@ page language="java" %>
    <%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
    <%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
    <html:html>
    <head>
    <body>
        <html:form action="/logon.do">
        <table border="0" width="100%">
        <tr>
            <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
            </th>
            <td align="left">
            <html:text property="username" size="16"/>
            </td>
        </tr>
        <tr>
        <td align="right">
            <html:submit><bean:message key="button.submit"/></html:submit>
        </td>
        <td align="right">
            <html:reset><bean:message key="button.reset"/></html:reset>
        </td>
        </tr>
        </table>
        </html:form>
    </body>
    </html:html>
```

[2] JavaServer Faces 技术
“JavaServer Faces 技术”是一组代表 UI 组件、管理组件状态、处理事件、验证输入和支持国际化的 Java API（JSR 127）。

JavaServer Faces API 定义“output_errors”UIOutput 处理器，该处理器显示整个页面的错误消息，或与指定的客户端标识相关联的错误消息。
使用 JavaServer Faces 来验证 loginForm 的 userName 字段的示例：

```
    <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
    <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

```
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use_faces>
```

引用
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java 正则表达式包 —
http://jakarta.apache.org/regexp/
Jakarta 验证器 —
http://jakarta.apache.org/commons/validator/
JavaServer Faces 技术 —
http://java.sun.com/j2ee/javaserverfaces/

整数溢出

** 输入数据验证：** 输入数据验证：虽然为方便用户而在客户端层上提供数据验证，但仍必须使用 Servlet 在服务器层上执行数据验证。客户端验证本身就不安全，因为这些验证可轻易绕过，例如，通过禁用 Javascript。
一份好的设计通常需要 Web 应用程序框架，以提供服务器端实用程序例程，从而验证以下内容：
[1] 必需字段
[2] 字段数据类型（缺省情况下，所有 HTTP 请求参数都是“字符串”）
[3] 字段长度
[4] 字段范围
[5] 字段选项
[6] 字段模式
[7] cookie 值
[8] HTTP 响应好的做法是将以上例程作为“验证器”实用程序类中的静态方法实现。以下部分描述验证器类的一个示例。
[1] 必需字段“始终”检查字段不为空，并且其长度要大于零，不包括行距和后面的空格。如何验证必需字段的示例：

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
        isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] 输入的 Web 应用程序中的字段数据类型和输入参数欠佳。例如，所有 HTTP 请求参数或 cookie 值的类型都是“字符串”。开发者负责验证输入的数据类型是否正确。使用 Java 基本包装程序类，来检查是否可将字段值安全地转换为所需的基本数据类型。
验证数字字段（int 类型）的方式的示例：

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
        Integer.parseInt(value);
        isFieldValid = true;
        } catch (Exception e) {
        isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

好的做法是将所有 HTTP 请求参数转换为其各自的数据类型。例如，将请求参数的“integerValue”存储在请求属性中，并按以下示例所示来使用：

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

应用程序应处理的主要 Java 数据类型：
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] 字段长度“始终”确保输入参数（HTTP 请求参数或 cookie 值）有最小长度和/或最大长度的限制。以下示例验证 userName 字段的长度是否在 8 至 20 个字符之间：

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
        validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
        validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] 字段范围
始终确保输入参数是在由功能需求定义的范围内。
以下示例验证输入 numberOfChoices 是否在 10 至 20 之间：

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
        // numberOfChoices is valid, continue processing request
        ...
        }
    }
}
```

[5] 字段选项 Web 应用程序通常会为用户显示一组可供选择的选项（例如，使用 SELECT HTML 标记），但不能执行服务器端验证以确保选定的值是其中一个允许的选项。请记住，恶意用户能够轻易修改任何选项值。始终针对由功能需求定义的受允许的选项来验证选定的用户值。以下示例验证用户针对允许的选项列表进行的选择：

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
        List list = Arrays.asList(options);
        if (list != null) {
        isValidValue = list.contains(value);
        }
        } catch (Exception e) {
        }
        return isValidValue;
    }
```

```
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] 字段模式
始终检查用户输入与由功能需求定义的模式是否匹配。例如，如果 userName 字段应仅允许字母数字字符，且不区分大小写，那么请使用以下正则表达式：^[a-zA-Z0-9]*$

Java 1.3 或更早的版本不包含任何正则表达式包。建议将“Apache 正则表达式包”（请参阅以下“资源”）与 Java 1.3 一起使用，以解决该缺乏支持的问题。执行正则表达式验证的示例：

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
        RE r = new RE(expression);
        match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 引进了一种新的正则表达式包 (java.util.regex)。以下是使用新的 Java 1.4 正则表达式包的 Validator.matchPattern 修订版：

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
        match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] cookie 值使用 javax.servlet.http.Cookie 对象来验证 cookie 值。适用于 cookie 值的相同的验证规则（如上所述）取决于应用程序需求（如验证必需值、验证长度等）。验证必需 cookie 值的示例：

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
        // validate the cookie value
        if (Validator.validateRequired(cookies[i].getValue()) {
        // valid cookie value, continue processing request
        ...
        }
        }
    }
}
```

[8] HTTP 响应
[8-1] 过滤用户输入要保护应用程序免遭跨站点脚本编制的攻击，请通过将敏感字符转换为其对应的字符实体来清理 HTML。这些是 HTML 敏感字符：< > " ' % ;) (& +

以下示例通过将敏感字符转换为其对应的字符实体来过滤指定字符串：

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
        return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
        switch (value.charAt(i)) {
        case '<':
        result.append("&lt;");
        break;
        case '>':
        result.append("&gt;");
        break;
        case '"':
        result.append("&quot;");
        break;
        case '\'':
        result.append("&#39;");
        break;
        case '%':
        result.append("&#37;");
        break;
        case ';':
        result.append("&#59;");
        break;
        case '(':
        result.append("&#40;");
        break;
        case ')':
        result.append("&#41;");
        break;
        case '&':
        result.append("&amp;");
        break;
        case '+':
        result.append("&#43;");
        break;
        default:
        result.append(value.charAt(i));
```

```
            break;
            }
            return result;
        }
        ...
    }
    ...
    // Filter the HTTP response using Validator.filter
    PrintWriter out = response.getWriter();
    // set output response
    out.write(Validator.filter(response));
    out.close();
```

Java Servlet API 2.3 引进了“过滤器”，它支持拦截和转换 HTTP 请求或响应。
以下示例使用 Validator.filter 来用“Servlet 过滤器”清理响应：

```
    // Example to filter all sensitive characters in the HTTP response using a Java Filter.
    // This example is for illustration purposes since it will filter all content in the response, including
HTML tags!
    public class SensitiveCharsFilter implements Filter {
        ...
        public void doFilter(ServletRequest request,
            ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {

            PrintWriter out = response.getWriter();
            ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
            chain.doFilter(request, wrapper);

            CharArrayWriter caw = new CharArrayWriter();
            caw.write(Validator.filter(wrapper.toString()));

            response.setContentType("text/html");
            response.setContentLength(caw.toString().length());
            out.write(caw.toString());
            out.close();
        }
        ...
        public class CharResponseWrapper extends HttpServletResponseWrapper {
            private CharArrayWriter output;

            public String toString() {
            return output.toString();
            }

            public CharResponseWrapper(HttpServletResponse response){
            super(response);
            output = new CharArrayWriter();
            }

            public PrintWriter getWriter(){
            return new PrintWriter(output);
            }
        }
    }

    }
```

[8-2] 保护 cookie
在 cookie 中存储敏感数据时，确保使用 Cookie.setSecure（布尔标志）在 HTTP 响应中设置 cookie 的安全标志，以指导浏览器使用安全协议（如 HTTPS 或 SSL）发送 cookie。
保护“用户”cookie 的示例：

```
// Example to secure a cookie, i.e. instruct the browser to
// send the cookie using a secure protocol
Cookie cookie = new Cookie("user", "sensitive");
cookie.setSecure(true);
response.addCookie(cookie);
```

推荐使用的 JAVA 工具用于服务器端验证的两个主要 Java 框架是：
[1] Jakarta Commons Validator（与 Struts 1.1 集成）Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。Jakarta Commons Validator 是一种强大的框架，用来实现所有以上数据验证需求。这些规则配置在定义表单字段的输入验证规则的 XML 文件中。在缺省情况下，Struts 支持在使用 Struts“bean:write”标记撰写的所有数据上，过滤 [8] HTTP 响应中输出的危险字符。可通过设置“filter=false”标志来禁用该过滤。
Struts 定义以下基本输入验证器，但也可定义定制的验证器：
required：如果字段包含空格以外的任何字符，便告成功。
mask：如果值与掩码属性给定的正则表达式相匹配，便告成功。
range：如果值在 min 和 max 属性给定的值的范围内（(value >= min) & (value <= max)），便告成功。
maxLength：如果字段长度小于或等于 max 属性，便告成功。
minLength：如果字段长度大于或等于 min 属性，便告成功。
byte、short、integer、long、float、double：如果可将值转换为对应的基本类型，便告成功。
date：如果值代表有效日期，便告成功。可能会提供日期模式。
creditCard：如果值可以是有效的信用卡号码，便告成功。
e-mail：如果值可以是有效的电子邮件地址，便告成功。
使用“Struts 验证器”来验证 loginForm 的 userName 字段的示例：

```
<form-validation>
    <global>
        ...
        <validator name="required"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateRequired"
        msg="errors.required">
        </validator>
        <validator name="mask"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateMask"
        msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
        <!-- userName is required and is alpha-numeric case insensitive -->
        <field property="userName" depends="required,mask">
        <!-- message resource key to display if validation fails -->
        <msg name="mask" key="login.userName.maskmsg"/>
        <arg0 key="login.userName.displayname"/>
        <var>
        <var-name>mask</var-name>
        <var-value>^[a-zA-Z0-9]*$</var-value>
        </var>
        </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

[2] JavaServer Faces 技术
“JavaServer Faces 技术”是一组代表 UI 组件、管理组件状态、处理事件和输入验证的 Java API (JSR 127)。
JavaServer Faces API 实现以下基本验证器，但可定义定制的验证器： validate_doublerange：在组件上注册 DoubleRangeValidator
validate_length：在组件上注册 LengthValidator

validate_longrange：在组件上注册 LongRangeValidator
validate_required：在组件上注册 RequiredValidator
validate_stringrange：在组件上注册 StringRangeValidator
validator：在组件上注册定制的 Validator

JavaServer Faces API 定义以下 UIInput 和 UIOutput 处理器（标记）：
input_date：接受以 java.text.Date 实例格式化的 java.util.Date
output_date：显示以 java.text.Date 实例格式化的 java.util.Date
input_datetime：接受以 java.text.DateTime 实例格式化的 java.util.Date
output_datetime：显示以 java.text.DateTime 实例格式化的 java.util.Date
input_number：显示以 java.text.NumberFormat 格式化的数字数据类型（java.lang.Number 或基本类型）
output_number：显示以 java.text.NumberFormat 格式化的数字数据类型（java.lang.Number 或基本类型）
input_text：接受单行文本字符串。
output_text：显示单行文本字符串。
input_time：接受以 java.text.DateFormat 时间实例格式化的 java.util.Date
output_time：显示以 java.text.DateFormat 时间实例格式化的 java.util.Date
input_hidden：允许页面作者在页面中包括隐藏变量
input_secret：接受不含空格的单行文本，并在输入时，将其显示为一组星号
input_textarea：接受多行文本
output_errors：显示整个页面的错误消息，或与指定的客户端标识相关联的错误消息
output_label：将嵌套的组件显示为指定输入字段的标签
output_message：显示本地化消息

使用 JavaServer Faces 来验证 loginForm 的 userName 字段的示例：

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use_faces>
```

引用
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java 正则表达式包 —
http://jakarta.apache.org/regexp/
Jakarta 验证器 —
http://jakarta.apache.org/commons/validator/
JavaServer Faces 技术 —
http://java.sun.com/j2ee/javaserverfaces/
** 错误处理：
许多 J2EE Web 应用程序体系结构都遵循“模型视图控制器（MVC）”模式。在该模式中，Servlet 扮演“控制器”的角色。Servlet 将应用程序处理委派给 EJB 会话 Bean（模型）之类的 JavaBean。然后，Servlet 再将请求转发给 JSP（视图），以呈现处理结果。Servlet 应检查所有的输入、输出、返回码、错误代码和已知的异常，以确保实际处理按

预期进行。
数据验证可保护应用程序免遭恶意数据篡改，而有效的错误处理策略则是防止应用程序意外泄露内部错误消息（如异常堆栈跟踪）所不可或缺的。好的错误处理策略会处理以下项：
[1] 定义错误
[2] 报告错误
[3] 呈现错误
[4] 错误映射
[1] 定义错误
应避免在应用程序层（如 Servlet）中硬编码错误消息。 相反地，应用程序应该使用映射到已知应用程序故障的错误密钥。好的做法是定义错误密钥，且该错误密钥映射到 HTML 表单字段或其他 Bean 属性的验证规则。例如，如果需要"user_name"字段，其内容为字母数字，并且必须在数据库中是唯一的，那么就应定义以下错误密钥：

(a) ERROR_USERNAME_REQUIRED：该错误密钥用于显示消息，以通知用户需要 "user_name" 字段；
(b) ERROR_USERNAME_ALPHANUMERIC：该错误密钥用于显示消息，以通知用户 "user_name" 字段应该是字母数字；
(c) ERROR_USERNAME_DUPLICATE：该错误密钥用于显示消息，以通知用户 "user_name" 值在数据库中重复；
(d) ERROR_USERNAME_INVALID：该错误密钥用于显示一般消息，以通知用户 "user_name" 值无效；

好的做法是定义用于存储和报告应用程序错误的以下框架 Java 类：
- ErrorKeys：定义所有错误密钥

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error：封装个别错误

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
    this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
    this.key = key;
    this.values = values;
    }

    // Returns the error key
    public String getKey() {
    return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
    return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors：封装错误的集合

```
// Example: Errors encapsulates the Error objects being reported to the presentation layer.
// Errors are stored in a HashMap where the key is the bean property name and value is an
// ArrayList of Error objects.
public Class Errors implements Serializable {

    // Adds an Error object to the Collection of errors for the specified bean property.
    public void addError(String property, Error error) {
    ArrayList propertyErrors = (ArrayList)errors.get(property);
    if (propertyErrors == null) {
    propertyErrors = new ArrayList();
    errors.put(property, propertyErrors);
    }
    propertyErrors.put(error);
    }

    // Returns true if there are any errors
    public boolean hasErrors() {
    return (errors.size > 0);
    }

    // Returns the Errors for the specified property
    public ArrayList getErrors(String property) {
    return (ArrayList)errors.get(property);
    }

    private HashMap errors = new HashMap();
}
```

以下是使用上述框架类来处理“user_name”字段验证错误的示例：

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] 报告错误
有两种方法可报告 web 层应用程序错误：
(a) Servlet 错误机制
(b) JSP 错误机制

[2-a] Servlet 错误机制
Servlet 可通过以下方式报告错误：
- 转发给输入 JSP（已将错误存储在请求属性中），或
- 使用 HTTP 错误代码参数来调用 response.sendError，或
- 抛出异常

好的做法是处理所有已知应用程序错误（如 [1] 部分所述），将这些错误存储在请求属性中，然后转发给输入 JSP。输入 JSP 应显示错误消息，并提示用户重新输入数据。以下示例阐明转发给输入 JSP（userInput.jsp）的方式：

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

如果 Servlet 无法转发给已知的 JSP 页面，那么第二个选项是使用 response.sendError 方法，将 HttpServletResponse.SC_INTERNAL_SERVER_ERROR（状态码 500）作为参数，来报告错误。请参阅 javax.servlet.http.HttpServletResponse 的 Javadoc，以获取有关各种 HTTP 状态码的更多详细信息。返回 HTTP 错误的示例：

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
        messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

作为最后的手段，Servlet 可以抛出异常，且该异常必须是以下其中一类的子类：
- RuntimeException
- ServletException
- IOException

[2-b] JSP 错误机制
JSP 页面通过定义 errorPage 伪指令来提供机制，以处理运行时异常，如以下示例所示：

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

未捕获的 JSP 异常被转发给指定的 errorPage，并且原始异常设置在名称为 javax.servlet.jsp.jspException 的请求参数中。错误页面必须包括 isErrorPage 伪指令，如下所示：

```
<%@ page isErrorPage="true" %>
```

isErrorPage 伪指令导致“exception”变量初始化为所抛出的异常对象。
[3] 呈现错误
J2SE Internationalization API 提供使应用程序资源外部化以及将消息格式化的实用程序类，其中包括：

(a) 资源束
(b) 消息格式化

[3-a] 资源束
资源束通过将本地化数据从使用该数据的源代码中分离来支持国际化。资源束通过将本地化数据从使用该数据的源代码中分离来支持国际化。每一资源束都会为特定的语言环境存储键/值对的映射。
java.util.PropertyResourceBundle 将内容存储在外部属性文件中，对其进行使用或扩展都很常见，如以下示例所示：

```
###############################################
# ErrorMessages.properties
###############################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

可定义多种资源，以支持不同的语言环境（因此名为资源束）。例如，可定义 ErrorMessages_fr.properties 以支持该束系列的法语成员。如果请求的语言环境的资源成员不存在，那么会使用缺省成员。在以上示例中，缺省资源是 ErrorMessages.properties。应用程序（JSP 或 Servlet）会根据用户的语言环境从适当的资源检索内容。
[3-b] 消息格式化
J2SE 标准类 java.util.MessageFormat 提供使用替换占位符来创建消息的常规方法。MessageFormat 对象包含嵌入了格式说明符的模式字符串，如下所示：

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

以下是使用 ResourceBundle 和 MessageFormat 来呈现错误消息的更加全面的示例：

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }

    // Returns the error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Locale locale) {
        return getErrorMessage(errorKey, null, locale);
    }

    // Returns a formatted error message for the specified error key in the specified locale
```

```
        public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
            // Get localized ErrorMessageResource
            ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
            // Get localized error message
            String errorMessage = errorMessageResource.getString(errorKey);
            if (args != null) {
            // Format the message using the specified placeholders args
            return MessageFormat.format(errorMessage, args);
            } else {
            return errorMessage;
            }
        }

        // default environment locale
        private Locale defaultLocale = Locale.getDefaultLocale();
    }
    ...
    // Get the user's locale
    Locale userLocale = request.getLocale();
    // Check if there were any validation errors
    Errors errors = (Errors)request.getAttribute("errors");
    if (errors != null && errors.hasErrors()) {
        // iterate through errors and output error messages corresponding to the "user_name" property
        ArrayList userNameErrors = errors.getErrors("user_name");
        ListIterator iterator = userNameErrors.iterator();
        while (iterator.hasNext()) {
            // Get the next error object
            Error error = (Error)iterator.next();
            String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
            output.write(errorMessage + "\r\n");
        }
    }
```

建议定义定制 JSP 标记（如 displayErrors），以迭代处理并呈现错误消息，如以上示例所示。
[4] 错误映射
通常情况下，“Servlet 容器”会返回与响应状态码或异常相对应的缺省错误页面。可以使用定制错误页面来指定状态码或异常与 Web 资源之间的映射。好的做法是开发不会泄露内部错误状态的静态错误页面（缺省情况下，大部分 Servlet 容器都会报告内部错误消息）。该映射配置在“Web 部署描述符（web.xml）”中，如以下示例所指定：

```
    <!-- Mapping of HTTP error codes and application exceptions to error pages -->
    <error-page>
      <exception-type>UserValidationException</exception-type>
      <location>/errors/validationError.html</error-page>
    </error-page>
    <error-page>
      <error-code>500</exception-type>
      <location>/errors/internalError.html</error-page>
    </error-page>
    <error-page>
    ...
    </error-page>
    ...
```

推荐使用的 JAVA 工具用于服务器端验证的两个主要 Java 框架是：
[1] Jakarta Commons Validator（与 Struts 1.1 集成）Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。Jakarta Commons Validator 是 Java 框架，定义如上所述的错误处理机制。验证规则配置在 XML 文件中，该文件定义了表单字段的输入验证规则以及对应的验证错误密钥。Struts 提供国际化支持以使用资源束和消息格式化来构建本地化应用程序。
使用“Struts 验证器”来验证 loginForm 的 userName 字段的示例：

```
    <form-validation>
        <global>
```

```
        ...
        <validator name="required"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateRequired"
        msg="errors.required">
        </validator>
        <validator name="mask"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateMask"
        msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
        <!-- userName is required and is alpha-numeric case insensitive -->
        <field property="userName" depends="required,mask">
        <!-- message resource key to display if validation fails -->
        <msg name="mask" key="login.userName.maskmsg"/>
        <arg0 key="login.userName.displayname"/>
        <var>
        <var-name>mask</var-name>
        <var-value>^[a-zA-Z0-9]*$</var-value>
        </var>
        </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

Struts JSP 标记库定义了有条件地显示一组累计错误消息的“errors”标记，如以下示例所示：

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
        <html:errors property="username"/>
        <bean:message key="prompt.username"/>
        </th>
        <td align="left">
        <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces 技术
“JavaServer Faces 技术”是一组代表 UI 组件、管理组件状态、处理事件、验证输入和支持国际化的 Java API（JSR 127）。

JavaServer Faces API 定义“output_errors”UIOutput 处理器，该处理器显示整个页面的错误消息，或与指定的客户端标识相关联的错误消息。
使用 JavaServer Faces 来验证 loginForm 的 userName 字段的示例：

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use_faces>
```

引用
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java 正则表达式包 —
http://jakarta.apache.org/regexp/
Jakarta 验证器 —
http://jakarta.apache.org/commons/validator/
JavaServer Faces 技术 —
http://java.sun.com/j2ee/javaserverfaces/

PHP

应用程序错误

** 输入数据验证：** 输入数据验证：虽然为方便用户而在客户端层上提供数据验证，但仍必须始终在服务器层上执行数据验证。客户端验证本身就不安全，因为这些验证可轻易绕过，例如，通过禁用 Javascript。
一份好的设计通常需要 Web 应用程序框架，以提供服务器端实用程序例程，从而验证以下内容：
[1] 必需字段
[2] 字段数据类型（缺省情况下，所有 HTTP 请求参数都是“字符串”）
[3] 字段长度
[4] 字段范围
[5] 字段选项
[6] 字段模式
[7] cookie 值
[8] HTTP 响应好的做法是实现一个或多个验证每个应用程序参数的函数。以下部分描述一些检查的示例。
[1] 必需字段“始终”检查字段不为空，并且其长度要大于零，不包括行距和后面的空格。如何验证必需字段的示例：

```
// PHP example to validate required fields
function validateRequired($input) {
    ...
    $pass = false;
    if (strlen(trim($input))>0){
        $pass = true;
    }
```

```
    return $pass;
    ...
}
...
if (validateRequired($fieldName)) {
    // fieldName is valid, continue processing request
    ...
}
```

[2] 输入的 Web 应用程序中的字段数据类型和输入参数欠佳。例如，所有 HTTP 请求参数或 cookie 值的类型都是“字符串”。开发者负责验证输入的数据类型是否正确。[3] 字段长度“始终”确保输入参数（HTTP 请求参数或 cookie 值）有最小长度和/或最大长度的限制。[4] 字段范围
始终确保输入参数是在由功能需求定义的范围内。
[5] 字段选项 Web 应用程序通常会为用户显示一组可供选择的选项（例如，使用 SELECT HTML 标记），但不能执行服务器端验证以确保选定的值是其中一个允许的选项。请记住，恶意用户能够轻易修改任何选项值。始终针对由功能需求定义的受允许的选项来验证选定的用户值。[6] 字段模式
始终检查用户输入与由功能需求定义的模式是否匹配。例如，如果 userName 字段应仅允许字母数字字符，且不区分大小写，那么请使用以下正则表达式：^[a-zA-Z0-9]+$

[7] cookie 值
适用于 cookie 值的相同的验证规则（如上所述）取决于应用程序需求（如验证必需值、验证长度等）。
[8] HTTP 响应[8-1] 过滤用户输入要保护应用程序免遭跨站点脚本编制的攻击，开发者应通过将敏感字符转换为其对应的字符实体来清理 HTML。这些是 HTML 敏感字符：< > " ' % ;) (& +

PHP 包含一些自动化清理实用程序函数，如 htmlentities()：

```
$input = htmlentities($input, ENT_QUOTES, 'UTF-8');
```

此外，为了避免“跨站点脚本编制”的 UTF-7 变体，您应该显式定义响应的 Content-Type 头，例如：

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] 保护 cookie
在 cookie 中存储敏感数据且通过 SSL 来传输时，请确保先在 HTTP 响应中设置 cookie 的安全标志。这将会指示浏览器仅通过 SSL 连接来使用该 cookie。
为了保护 cookie，您可以使用以下代码示例：

```
<$php

    $value = "some_value";
    $time = time()+3600;
    $path = "/application/";
    $domain = ".example.com";
    $secure = 1;

    setcookie("CookieName", $value, $time, $path, $domain, $secure, TRUE);
?>
```

此外，我们建议您使用 HttpOnly 标志。当 HttpOnly 标志设置为 TRUE 时，将只能通过 HTTP 协议来访问 cookie。这意味着无法用脚本语言（如 JavaScript）来访问 cookie。该设置可有效地帮助减少通过 XSS 攻击盗用身份的情况（虽然并非所有浏览器都支持该设置）。
在 PHP 5.2.0 中添加了 HttpOnly 标志。
引用[1] 使用 HTTP 专用 cookie 来减轻“跨站点脚本编制”的影响：
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP 安全协会：
http://phpsec.org/
[3] PHP 和 Web 应用程序安全博客 (Chris Shiflett)：
http://shiflett.org/

整数溢出

** 输入数据验证：** 输入数据验证：虽然为方便用户而在客户端层上提供数据验证，但仍必须始终在服务器层上执行数据验证。客户端验证本身就不安全，因为这些验证可轻易绕过，例如，通过禁用 Javascript。
一份好的设计通常需要 Web 应用程序框架，以提供服务器端实用程序例程，从而验证以下内容：
[1] 必需字段
[2] 字段数据类型（缺省情况下，所有 HTTP 请求参数都是“字符串”）
[3] 字段长度
[4] 字段范围
[5] 字段选项
[6] 字段模式
[7] cookie 值
[8] HTTP 响应好的做法是实现一个或多个验证每个应用程序参数的函数。以下部分描述一些检查的示例。
[1] 必需字段“始终”检查字段不为空，并且其长度要大于零，不包括行距和后面的空格。如何验证必需字段的示例：

```
// PHP example to validate required fields
function validateRequired($input) {
    ...
    $pass = false;
    if (strlen(trim($input))>0){
        $pass = true;
    }
    return $pass;
    ...
}
...
if (validateRequired($fieldName)) {
    // fieldName is valid, continue processing request
    ...
}
```

[2] 输入的 Web 应用程序中的字段数据类型和输入参数欠佳。例如，所有 HTTP 请求参数或 cookie 值的类型都是“字符串”。开发者负责验证输入的数据类型是否正确。[3] 字段长度“始终”确保输入参数（HTTP 请求参数或 cookie 值）有最小长度和/或最大长度的限制。[4] 字段范围
始终确保输入参数是在由功能需求定义的范围内。
[5] 字段选项 Web 应用程序通常会为用户显示一组可供选择的选项（例如，使用 SELECT HTML 标记），但不能执行服务器端验证以确保选定的值是其中一个允许的选项。请记住，恶意用户能够轻易修改任何选项值。始终针对由功能需求定义的受允许的选项来验证选定的用户值。[6] 字段模式
始终检查用户输入与由功能需求定义的模式是否匹配。例如，如果 userName 字段应仅允许字母数字字符，且不区分大小写，那么请使用以下正则表达式：^[a-zA-Z0-9]+$

[7] cookie 值
适用于 cookie 值的相同的验证规则（如上所述）取决于应用程序需求（如验证必需值、验证长度等）。

[8] HTTP 响应[8-1] 过滤用户输入要保护应用程序免遭跨站点脚本编制的攻击，开发者应通过将敏感字符转换为其对应的字符实体来清理 HTML。这些是 HTML 敏感字符：< > " ' % ;) (& +

PHP 包含一些自动化清理实用程序函数，如 htmlentities()：

```
$input = htmlentities($input, ENT_QUOTES, 'UTF-8');
```

此外，为了避免“跨站点脚本编制”的 UTF-7 变体，您应该显式定义响应的 Content-Type 头，例如：

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] 保护 cookie
在 cookie 中存储敏感数据且通过 SSL 来传输时，请确保先在 HTTP 响应中设置 cookie 的安全标志。这将会指示浏览器仅通过 SSL 连接来使用该 cookie。
为了保护 cookie，您可以使用以下代码示例：

```
<$php

    $value = "some_value";
    $time = time()+3600;
    $path = "/application/";
    $domain = ".example.com";
    $secure = 1;

    setcookie("CookieName", $value, $time, $path, $domain, $secure, TRUE);
?>
```

此外，我们建议您使用 HttpOnly 标志。当 HttpOnly 标志设置为 TRUE 时，将只能通过 HTTP 协议来访问 cookie。这意味着无法用脚本语言（如 JavaScript）来访问 cookie。该设置可有效地帮助减少通过 XSS 攻击盗用身份的情况（虽然并非所有浏览器都支持该设置）。
在 PHP 5.2.0 中添加了 HttpOnly 标志。
引用[1] 使用 HTTP 专用 cookie 来减轻“跨站点脚本编制”的影响：
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP 安全协会：
http://phpsec.org/
[3] PHP 和 Web 应用程序安全博客 (Chris Shiflett)：
http://shiflett.org/

咨询

“Content-Security-Policy”头缺失或不安全

TOC

测试类型：
应用程序级别测试

威胁分类：
信息泄露

原因：
Web 应用程序编程或配置不安全

安全性风险：
- 可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置
-

受影响产品：

CWE:
200

引用：
有用 HTTP 头列表
内容安全策略简介
MDN Web 文档 - 内容安全策略

技术描述：
“内容安全策略”标头旨在修改浏览器呈现页面的方式，从而防止各种跨站点注入，包括跨站点脚本。以不妨碍网站正常运行的方式正确设置标头值非常重要。例如，如果将标头设置为阻止执行内联 JavaScript，则网站不得在其页面中使用内联 JavaScript。
为了防止跨站点脚本、跨框架脚本和点击劫持，使用适当的值设置以下策略非常重要：
'default-src' 和 'frame-ancestors'策略、*或*所有 'script-src'、'object-src' 和 'frame-ancestors' 策略。
对于 'default-src'、'script-src' 和 'object-src'，应避免使用不安全的值，例如 '*'、'data:'、'unsafe-inline' 或 'unsafe-eval'。
对于 'frame-ancestors'，应避免使用不安全的值，例如 '*' 或 'data:'。
有关更多信息，请参阅以下链接。

“X-Content-Type-Options”头缺失或不安全

TOC

测试类型：

应用程序级别测试

威胁分类：

信息泄露

原因：

Web 应用程序编程或配置不安全

安全性风险：

- 可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置
-

受影响产品：

CWE:

200

引用：

有用 HTTP 头列表
减少 MIME 类型安全风险

技术描述：

“X-Content-Type-Options”头（具有“nosniff”值）防止 IE 和 Chrome 忽略响应的内容类型。
此操作可能防止不可信内容（例如，用户上载内容）在用户浏览器上执行（例如，在恶意命名之后）。

“X-XSS-Protection”报头缺失或不安全

TOC

测试类型：

应用程序级别测试

威胁分类：

信息泄露

原因：

Web 应用程序编程或配置不安全

安全性风险：

- 可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置
-

受影响产品：

CWE:

200

引用：

有用 HTTP 头列表
IE XSS 过滤器

技术描述：

值为‘1’的“X-XSS-Protection”报头强制跨站点脚本编制过滤器进入启用模式，即使用户已禁用。
此过滤器内置于最新版本的 Web 浏览器（IE 8 +、Chrome 4+）中，在缺省情况下通常为已启用状态。虽然此过滤器不是第一个也不是唯一一个针对跨站点脚本编制的防御程序，但它可以作为额外保护层。

临时文件下载

TOC

测试类型：

应用程序级别测试

威胁分类：

可预测资源位置

原因：

在生产环境中留下临时文件

安全性风险：

受影响产品：

CWE:

531

X-Force：

52887

引用：
WASC 威胁分类：可预期的资源位置

技术描述：
Web 服务器通常会使“公共网关接口（CGI）”文件扩展名（如 .pl）与 Perl 之类的某个处理程序相关联。 当 URL 路径结尾是 .pl 时，路径所指定的文件名会发送给 Perl 执行；文件内容不会返回给浏览器。 然而，当在适当的位置编辑脚本文件时，编辑器可以用新的文件扩展名来保存所编辑的脚本的备份副本，例如：.bak、.sav、.old、~ 等等。Web 服务器通常没有这些文件扩展名的特定处理程序。 如果攻击者请求这类文件，文件内容会直接发送到浏览器。
从虚拟目录下除去这些临时文件很重要，因为它们可能含有调试目的所用的敏感信息，也可能显露有并非当前逻辑，但仍可能受到利用的应用程序逻辑攻击。

发现可能的服务器路径泄露模式

TOC

测试类型：
应用程序级别测试

威胁分类：
信息泄露

原因：
未安装第三方产品的最新补丁或最新修补程序

安全性风险：
可能会检索 Web 服务器安装的绝对路径，这可能会帮助攻击者开展进一步攻击和获取有关 Web 应用程序文件系统结构的信息

受影响产品：

CWE:
200

X-Force：
52839

技术描述：
AppScan 检测到包含文件绝对路径的响应（例如，Windows 中的 c:\dir\file，或 Unix 中的 /dir/file）。

攻击者可能能够利用这一信息访问服务器机器目录结构上的敏感信息，进而对站点发起进一步攻击。

发现内部 IP 泄露模式

TOC

测试类型：
应用程序级别测试

威胁分类：
信息泄露

原因：
Web 应用程序编程或配置不安全

安全性风险：
可能会收集有关 Web 应用程序的敏感信息，如用户名、密码、机器名和/或敏感文件位置

受影响产品：

CWE:
200

X-Force：
52657

技术描述：
AppScan 检测到包含内部 IP 地址的响应。
内部 IP 定义为以下 IP 范围内的 IP：
10.0.0.0 - 10.255.255.255
172.16.0.0 - 172.31.255.255
192.168.0.0 - 192.168.255.255

内部 IP 公开对于攻击者非常有价值，因为它揭示了内部网络的 IP 联网模式。获知内部网络的 IP 联网模式可能会帮助攻击者计划针对内部网络的进一步攻击。

应用程序错误

TOC

测试类型：
应用程序级别测试

威胁分类：
信息泄露

原因：
- 未对入局参数值执行适当的边界检查
- 未执行验证以确保用户输入与预期的数据类型匹配

安全性风险：
可能会收集敏感的调试信息

受影响产品：

CWE:
550

X-Force：
52502

引用：
使用单引号入侵站点的示例，可参阅“How I hacked PacketStorm (by Rain Forest Puppy), RFP's site”
“Web Application Disassembly with ODBC Error Messages”（作者：David Litchfield）
CERT 咨询（CA-1997-25）：清理 CGI 脚本中用户提供的数据

技术描述：
如果攻击者通过伪造包含非应用程序预期的参数或参数值的请求，来探测应用程序（如以下示例所示），那么应用程序可能会进入易受攻击的未定义状态。 攻击者可以从应用程序对该请求的响应中获取有用的信息，且可利用该信息，以找出应用程序的弱点。
例如，如果参数字段是单引号括起来的字符串（如在 ASP 脚本或 SQL 查询中），那么注入的单引号将会提前终止字符串流，从而更改脚本的正常流程/语法。
错误消息中泄露重要信息的另一个原因，是脚本编制引擎、Web 服务器或数据库配置错误。
以下是一些不同的变体：
[1] 除去参数
[2] 除去参数值
[3] 将参数值设置为空值
[4] 将参数值设置为数字溢出（+/- 99999999）
[5] 将参数值设置为危险字符，如 ' " \' \") ;
[6] 将某字符串附加到数字参数值
[7] 在参数名称后追加“.”（点）或“[]”（尖括号）

整数溢出

TOC

测试类型：
应用程序级别测试

威胁分类：
整数溢出

原因：
- 未对入局参数值执行适当的边界检查
- 未执行验证以确保用户输入与预期的数据类型匹配

安全性风险：

可能会收集敏感的调试信息

受影响产品：

CWE:

550

引用：

使用单引号入侵站点的示例，可参阅“How I hacked PacketStorm (by Rain Forest Puppy), RFP's site”
“Web Application Disassembly with ODBC Error Messages”（作者：David Litchfield）
CERT 咨询（CA-1997-25）：清理 CGI 脚本中用户提供的数据

技术描述：

如果攻击者通过伪造包含非应用程序预期的参数或参数值的请求，来探测应用程序（如以下示例所示），那么应用程序可能会进入易受攻击的未定义状态。 攻击者可以从应用程序对该请求的响应中获取有用的信息，且可利用该信息，以找出应用程序的弱点。
例如，如果参数字段是单引号括起来的字符串（如在 ASP 脚本或 SQL 查询中），那么注入的单引号将会提前终止字符串流，从而更改脚本的正常流程/语法。
错误消息中泄露重要信息的另一个原因，是脚本编制引擎、Web 服务器或数据库配置错误。
以下是一些不同的变体：
[1] 除去参数
[2] 除去参数值
[3] 将参数值设置为空值
[4] 将参数值设置为数字溢出（+/- 99999999）
[5] 将参数值设置为危险字符，如 ' " \' \") ;
[6] 将某字符串附加到数字参数值
[7] 在参数名称后追加“.”（点）或“[]”（尖括号）