# 2024漏洞分类指北手册

## 注入漏洞

### SQL注入（SQLi）

**参数：**

- 用户输入在包含在 SQL 查询中之前未经过正确的清理。
- 在代码中动态生成 SQL 查询。
- 使用连接字符串构建 SQL 查询。
- 缺乏 SQL 查询的输入验证。
- 查找从数据库检索数据或通信的任何用户输入！

**应遵循的步骤：**

1. 识别 Web 表单或 URL 参数中的用户输入字段。
2. 在这些字段中输入恶意SQL命令。
3. 观察是否有任何 SQL 错误或意外行为。
4. 如果出现错误或意外行为，则表明存在 SQL 注入漏洞。
5. 尝试并测试不同的 SQL 注入技术（例如基于 UNION、基于布尔或基于时间）来利用该漏洞。

### 跨站脚本 (XSS)

**参数：**

- 缺乏输入验证和清理。
- 将用户输入反射回网页，无需编码。
- 内容安全策略 (CSP) 使用不当
- Web 应用程序中缺乏正确的输出编码
- 在没有适当过滤的情况下使用 JavaScript 等客户端脚本语言。

**应遵循的步骤：**

1. 识别用户输入字段或网页上显示用户数据的区域。
2. 在这些字段中输入脚本或 JavaScript 代码。
3. 提交输入并观察脚本是否执行。
4. 如果脚本执行，则表明存在 XSS 漏洞。
5. 测试不同的XSS负载，例如<script>alert('XSS')</script>或<img src="javascript:alert('XSS')">以利用该漏洞

人们可以使用 Burp Intruder 尝试多个有效负载并观察利用的响应。

### 跨站请求伪造 (CSRF)

**参数：**

- 敏感操作中缺少反 CSRF 令牌。
- 缺乏对请求来源的验证。

- 容易受到未经用户同意而执行未经授权的操作的请求的影响。
- 未能实现 cookie 的 SameSite 属性。



# 2024漏洞分类指北手册

**应遵循的步骤：**

1. 识别敏感操作，例如更改密码或转移资金。
2. 提交请求以从外部站点执行这些操作。
3. 观察是否在未经用户同意的情况下执行操作。
4. 如果未经同意执行操作，则表明存在 CSRF 漏洞。
5. 为cookie实现防CSRF令牌和SameSite属性，以防止CSRF攻击。

## 远程代码执行 (RCE)

**参数：**

- 易受攻击的代码执行环境。
- 数据的不安全反序列化。
- 缺乏输入验证和清理。
- 网络协议或服务的安全措施不足。
- 攻击者执行任意代码。

**遵循的步骤：**

1. 识别可以在服务器上执行代码的输入字段或参数，例如用户输入字段、HTTP 请求参数、表单数据、文件上传、API接口、数据库查询和服务器端脚本语言。
2. 在这些字段中输入执行代码的有效负载或命令。
3. 提交输入，观察代码是否执行。
4. 如果出现代码执行，则表明存在RCE漏洞。
5. 测试不同的 RCE 负载（例如命令注入或 shell 注入）以利用该漏洞。

## 命令注入

**参数：**

- 缺乏系统命令的输入验证和清理。
- 使用连接字符串来构造系统命令。
- 攻击者执行任意命令。
- 易受攻击的代码执行环境。

**遵循的步骤：**

1. 识别用于执行系统命令的输入字段或参数。
2. 在这些字段中输入执行任意命令的有效负载或命令。

3. 提交输入并观察命令是否执行。
4. 如果发生命令执行，则表明存在命令注入漏洞。
5. 测试不同的命令注入有效负载。

XML注入

**参数：**

- 缺乏 XML 数据的输入验证和清理。
- 在 XML 解析器中使用不受信任的 XML 数据。
- 攻击者执行恶意 XML 代码。
- 易受攻击的 XML 处理库或框架。



**应遵循的步骤：**

1. 识别接受 XML 数据的输入字段或参数。
2. 输入可以操作 XML 解析器的有效负载或 XML 代码。
3. 提交输入并观察代码是否被执行或解析。
4. 如果发生代码执行或解析，则表明存在 XML 注入漏洞。 5. 测试不同的 XML 注入负载，例如 <![CDATA[<payload>]]> 或 <!ENTITY % xx '<payload>'> 以利用该漏洞。

LDAP注入

**参数：**

- 缺乏 LDAP 查询的输入验证和清理。
- 在 LDAP 查询中使用不受信任的输入。
- 攻击者执行恶意 LDAP 语句。
- 易受攻击的 LDAP 库或框架。

**遵循的步骤：**

1. 识别接受 LDAP 查询的输入字段或参数。
2. 输入可以操作 LDAP 查询的有效负载或 LDAP 语句。
3. 提交输入并观察查询是否执行。
4. 如果发生查询执行，则表明存在 LDAP 注入漏洞。
5. 测试不同的 LDAP 注入有效负载，例如 *()

XPath 注入

**参数：**

- 缺乏 XPath 查询的输入验证和清理。
- 在 XPath 查询中使用不受信任的输入。
- 攻击者执行恶意 XPath 语句。
- 易受攻击的 XPath 处理库或框架。

**遵循的步骤：**

1. 识别接受 XPath 查询的输入字段或参数。

2. 输入可以操作 XPath 查询的有效负载或 XPath 语句。
3. 提交输入并观察查询是否执行。
4. 如果发生查询执行，则表明存在 XPath 注入漏洞。
5. 测试不同的 XPath 注入有效负载（例如 ’ or 1=1 或 ’1’=’1’）以利用该漏洞。

**HTML注入**

**参数：**

- 缺乏 HTML 数据的输入验证和清理。
- 将不受信任的 HTML 数据反射回网页，无需编码。 - 执行恶意 HTML 代码。
- 易受攻击的 HTML 渲染引擎或框架。





**遵循的步骤：**

1. 识别接受 HTML 数据的输入字段或参数。
2. 输入可以操纵 HTML 渲染的有效负载或 HTML 代码。
3. 提交输入并观察代码是否被执行或渲染。
4. 如果发生代码执行或渲染，则表明存在 HTML 注入漏洞。
5. 测试不同的HTML注入负载，例如“<h><a href=https://bing.com>恶意链接</a></h>”以利用该漏洞。

**服务器端包含 (SSI) 注入**

**参数：**

- 缺乏 SSI 命令的输入验证和清理。
- 在 SSI 命令中使用不受信任的输入。
- 执行恶意 SSI 代码。
- 易受攻击的 SSI 处理引擎或框架。

**遵循的步骤：**

1. 识别接受 SSI 命令的输入字段或参数。
2. 输入可以操纵 SSI 处理的有效负载或 SSI 代码。
3.提交输入，观察代码是否执行。
4. 如果发生代码执行，则表明存在 SSI 注入漏洞。
5. 测试不同的 SSI 注入负载以利用该漏洞。

实施输入验证并清理 SSI 命令以降低 SSI 注入的风险。

## 失效的身份验证和会话管理

**会话固定**

**参数：**

- 身份验证后缺乏会话重新生成。
- 通过未加密通道传输的会话 ID。
- 缺乏会话验证或过期控制。

**遵循的步骤：**

1. 观察认证后Session ID是否不变。
2. 检查会话ID是否通过不安全的通道传输。
3. 测试会话是否无限期保持活动状态或缺乏过期控制。
4. 如果满足这些条件中的任何一个，则表明存在会话固定漏洞。
5. 尝试通过将特定会话 ID 强加给用户来固定会话。

**暴力攻击**

**参数：**

- 缺乏帐户锁定机制。
- 弱密码或可预测密码。
- 缺乏对登录尝试的速率限制。
- 不监控异常登录模式。





**遵循的步骤：**

1. 尝试使用各种用户名和密码组合登录帐户。
2. 测试是否存在账户锁定机制或速率限制。
3. 监控登录尝试是否存在任何异常模式。
4. 如果通过过多的尝试而成功登录，则表明容易遭受暴力攻击。
5. 实施强密码策略和帐户锁定机制以降低风险。

**会话劫持**

**参数：**

- 会话管理实践薄弱。
- 使用可预测的会话 ID。
- 缺乏安全传输协议。
- 缺乏检测未经授权的访问的机制。

**遵循的步骤：**

1. 监视以纯文本形式传输的会话 ID 的网络流量。
2. 检查易于猜测或预测的会话 ID。
3. 测试容易被劫持而不被发现的会话。
4. 如果可能出现未经授权的会话访问，则表明存在会话劫持漏洞。

5. 实施安全会话管理实践并使用加密来传输会话数据。

**密码破解**

**参数：**

- 以纯文本或弱哈希格式存储密码。
- 使用弱哈希算法。
- 密码存储中缺乏加盐。
- 没有针对密码复杂性或过期的策略。

**遵循的步骤：**

1. 访问密码数据库并分析密码的存储方式。
2. 通过尝试使用常用技术破解密码来测试密码的强度。
3. 检查密码散列中是否缺少盐。
4. 如果密码很容易被破解或者以不安全的格式存储，则表明密码容易被破解。
5. 实施强大的密码散列算法、加盐并强制执行密码策略以提高安全性。

**弱密码存储**

**参数：**

- 以纯文本形式存储密码。
- 使用弱哈希算法而不加盐。
- 易受攻击的密码存储机制。





**遵循的步骤：**

1. 检查密码在系统中的存储方式。
2. 检查密码是否以纯文本形式存储或使用弱哈希算法。
3. 测试密码散列中是否不存在盐。
4. 如果密码存储不安全，则表明存在弱密码存储漏洞。
5. 实施安全密码存储机制，例如强哈希算法和加盐。

**不安全的身份验证**

**参数：**

- 缺乏多重身份验证。
- 使用不安全的身份验证协议，例如 HTTP 基本身份验证。
- 缺乏用于验证数据的安全传输通道。

**遵循的步骤：**

1. 检查所使用的身份验证过程和协议。
2. 测试是否存在多重身份验证。

3. 检查认证数据是否通过不安全的通道传输。
4. 如果身份验证缺乏安全措施，则表明存在不安全身份验证的漏洞。
5. 实施多重身份验证并使用 HTTPS 等安全协议进行身份验证。

### 饼干盗窃

**参数：**

- 通过未加密的通道传输 cookie。
- 缺乏安全 cookie 属性，例如 HttpOnly 和 Secure。
- 脆弱的会话管理实践。

**遵循的步骤：**

1. 监视网络流量以进行 cookie 传输。
2. 检查cookie是否缺少HttpOnly、Secure等安全属性。
3. 测试会话管理实践中的漏洞。
4. 如果cookie很容易被窃取或操纵，则表明cookie容易被窃取。
5. 实施安全的 cookie 处理实践并使用加密传输 cookie。

### 凭证重用

**参数：**

- 缺乏密码轮换策略。
- 在多个帐户中使用相同的凭据。
- 缺乏对凭证重用攻击的监控。

**遵循的步骤：**

1. 审查密码管理政策和实践。
2. 检查多个帐户是否使用相同的凭据。
3. 测试密码是否可以重复使用而无需轮换。





4. 监控跨账户的凭证重用模式。
5. 如果观察到凭证重用，则表明存在凭证重用攻击的漏洞。
6. 实施密码轮换策略并监控凭证重用，以降低风险。

## 敏感数据暴露

### 加密不足

**参数：**

- 敏感数据传输缺乏加密。
- 使用弱加密算法或过时的协议。
- 静态数据缺乏加密。
- 未能实施加密密钥管理实践。

**遵循的步骤：**

1. 检查数据传输协议并检查敏感数据是否以明文形式传输。
2. 评估所使用的加密算法和协议的强度和安全性。
3. 检查静态数据是否未加密存储。
4. 检查加密密钥管理实践，以确保正确处理和保护加密密钥。
5. 如果满足这些条件中的任何一个，则表明存在加密不足的漏洞。

**不安全的直接对象引用 (IDOR)**

**参数：**

- 无需适当授权即可直接访问对象或资源。
- 缺乏访问控制或授权检查。
- 通过可预测的对象引用暴露敏感数据。

**遵循的步骤：**

1. 识别可通过直接引用访问的资源或对象。
2. 测试是否强制执行授权检查来访问这些资源。
3. 尝试通过操纵对象引用或 URL 来访问敏感数据。
4. 如果可以通过直接对象引用对敏感数据进行未经授权的访问，则表明存在 IDOR 漏洞。
5. 实施适当的访问控制和授权检查以防止 IDOR 攻击。

**数据泄露**

**参数：**

- 敏感数据处理不当。
- 缺乏数据屏蔽或匿名化技术。
- 易受攻击的数据存储或传输实践。
- 对数据的访问控制不足。

**遵循的步骤：**

1. 检查数据处理流程并检查敏感数据是否存在任何不当处理。
2. 评估敏感数据是否得到充分屏蔽或匿名化以保护隐私。



3. 评估数据存储和传输实践中的漏洞。
4. 检查是否有效实施访问控制以限制未经授权的数据访问。
5. 如果敏感数据被泄露或暴露，则表明存在数据泄露的漏洞。
6、实施加密、数据脱敏、访问控制等数据保护措施，防止数据泄露。

**未加密的数据存储**

**参数：**

- 以纯文本或未加密格式存储敏感数据。
- 缺乏数据存储的加密机制。
- 无法保护包含敏感信息的数据库或文件系统。

**遵循的步骤：**

1. 检查数据存储实践并检查敏感数据是否以纯文本形式存储。
2. 评估是否采用加密机制来保护数据库或文件系统中存储的数据。
3. 评估为保护包含敏感信息的数据库或文件系统而实施的安全措施。
4. 如果数据未加密存储，则表明未加密数据存储存在漏洞。
5. 对数据存储实施加密，并加强安全措施，保护敏感数据免遭未经授权的访问。

**缺少安全标头**

**参数：**

- 缺少与安全相关的 HTTP 标头，例如内容安全策略 (CSP)、严格传输安全 (HSTS) 或 X-Content-Type-Options。
- 未能实施跨源资源共享 (CORS) 政策。
- 缺乏针对常见网络漏洞的保护。

**遵循的步骤：**

1. 分析 HTTP 响应并检查是否存在与安全相关的标头，例如 CSP、HSTS 和 X-Content-Type-Options。
2. 评估 CORS 策略以确保正确配置和实施。
3. 检查 Web 应用程序是否针对常见漏洞（例如 XSS、CSRF 和点击劫持）提供了充分的保护。
4. 如果安全标头丢失或配置错误，则表明存在缺少安全标头的漏洞。
5. 实施适当的安全标头和策略，以增强 Web 应用程序的安全状况。





**不安全的文件处理**

**参数：**

- 缺乏文件上传验证。
- 执行来自不受信任来源的文件。
- 易受攻击的文件访问控制。

- 无法清理文件名或路径。

**遵循的步骤：**

1. 检查文件上传功能并检查上传的文件的内容和文件类型是否经过验证。
2. 评估来自不受信任来源的文件是否在未经适当验证的情况下执行。
3. 评估文件访问控制以确保只有授权用户才能访问或操作文件。
4. 检查文件名或路径是否经过清理，以防止目录遍历攻击。
5. 如果文件处理实践不安全，则表明存在不安全文件处理的漏洞。
6. 实施安全文件上传流程、验证文件内容并实施适当的访问控制以降低被利用的风险。

## 安全配置错误

### 默认密码

**参数：**

- 对帐户或系统使用默认或容易猜测的密码。
- 安装或部署后无法更改默认密码。
- 缺乏密码复杂性要求。
- 缺乏密码过期策略。

**遵循的步骤：**

1. 查看系统或设备文档以了解默认密码。
2. 检查安装或部署后是否仍在使用默认密码。
3. 评估密码策略的复杂性要求和过期设置。
4. 跨帐户或系统测试弱密码或容易猜到的密码。
5. 如果发现默认密码或弱密码，则表明默认密码存在漏洞。
6. 实施强密码策略，强制更改密码，并避免使用默认密码以降低风险。

### 目录列表

**参数：**

- 列出目录内容的可公开访问的目录。
- Web 服务器配置中缺少索引文件或默认页面。
- 缺乏目录列表的访问控制。
- 通过目录列表披露敏感信息。





**遵循的步骤：**

1. 导航到 Web 服务器上的目录并检查是否显示目录列表。

2. 检查默认页面或索引文件的 Web 服务器配置。
3. 测试目录访问控制以查看列表是否受到限制。
4. 查找通过目录列表暴露的敏感信息。
5. 如果目录可公开访问并列出内容，则表明目录列表存在漏洞。
6. 实施访问控制并禁用目录列表，以防止泄露敏感信息。

### 未受保护的 API接口

**参数：**

- 在没有身份验证或授权机制的情况下暴露 API。
- 通过 API 传输的数据缺乏加密。
- API 请求没有速率限制或限制。
- 通过 API 披露敏感信息。

**遵循的步骤：**

1. 识别公开的或无需身份验证即可访问的 API。
2. 测试API是否需要身份验证或授权才能访问。
3. 检查API传输的数据是否加密。
4. 评估 API 请求是否存在速率限制或节流机制。
5. 查找通过 API 响应暴露的敏感信息。
6. 如果 API 未受保护并暴露敏感数据，则表明未受保护的 API接口存在漏洞。
7. 实施身份验证、加密、速率限制和数据保护措施以保护 API接口的安全。

### 开放端口和服务

**参数：**

- 识别网络设备上的开放端口和服务。
- 系统上运行不必要或未经授权的服务。
- 缺乏端口过滤或防火墙规则来限制访问。
- 易受攻击的服务面临潜在攻击。

**遵循的步骤：**

1. 对网络设备进行端口扫描，识别开放端口。
2. 检查系统配置以确定正在运行的服务。
3. 测试端口过滤或防火墙规则以限制访问。
4. 查找在开放端口上运行的易受攻击的服务。
5. 如果暴露了不必要的或易受攻击的服务，则表明开放的端口和服务存在漏洞。
6. 实施端口过滤、防火墙规则，禁用不必要的服务以减少攻击面。



### 不当的访问控制

**参数：**

- 认证机制不足。
- 缺乏对资源或功能的授权检查。
- 未能执行最小特权原则。
- 由于访问控制不足而泄露敏感信息。

**遵循的步骤：**

1. 检查身份验证机制，确保其稳健且安全。
2. 评估是否对访问资源实施了适当的授权检查。
3. 检查是否强制执行最小权限原则来限制访问。
4. 查找无需适当访问控制即可访问敏感信息的实例。
5. 如果访问控制不充分，则表明存在访问控制不当的漏洞。
6. 实施强大的身份验证、授权机制和最小权限原则，以增强访问控制并保护敏感信息。

**信息披露**

**参数：**

- 在错误消息或响应中暴露敏感信息。
- 泄露文件路径、堆栈跟踪或系统配置。
- 缺乏适当的错误处理和屏蔽技术。
- 敏感数据的不安全传输。

**遵循的步骤：**

1. 触发 Web 应用程序或系统中的错误条件以观察错误消息。
2. 查看错误消息是否泄露了任何敏感信息。
3. 检查响应中是否公开了文件路径、堆栈跟踪或系统配置。
4. 评估漏洞的错误处理实践。
5. 测试敏感数据跨网络通道的不安全传输。
6. 如果敏感信息被暴露，则表明存在信息泄露的漏洞。
7. 实施适当的错误处理、数据屏蔽和加密技术，以防止敏感信息泄露。

**未打补丁的软件**

**参数：**

- 存在具有已知漏洞的过时软件版本。
- 未能及时应用安全补丁或更新。
- 缺乏漏洞管理流程。
- 暴露于利用已知漏洞的情况。

**遵循的步骤：**

1. 识别系统上安装的软件版本并将其与最新的可用版本进行比较。
2. 检查软件是否已打安全补丁或更新。
3. 审查漏洞管理流程，确保及时应用补丁。



4. 评估利用未修补软件中的已知漏洞的风险。

5. 如果发现过时或未打补丁的软件，则表明未打补丁的软件存在漏洞。
6. 实施定期补丁管理流程并及时更新安全建议，以降低被利用的风险。

### CORS 配置错误

**参数：**

- 跨源资源共享 (CORS) 策略配置不当。 - 缺乏对跨域请求的限制。
- 未能执行安全的 CORS 策略。
- 遭受跨源攻击，例如 CSRF 或数据盗窃。

**遵循的步骤：**

1. 检查 Web 服务器或应用程序上配置的 CORS 策略。
2. 检查是否允许跨源请求且没有适当的限制。
3. 评估是否强制执行安全 CORS 策略以防止跨源攻击。
4. 测试由于 CORS 策略配置错误而导致的 CSRF 或数据盗窃等漏洞。
5. 如果 CORS 策略配置错误，则表明存在 CORS 配置错误的漏洞。
6. 实施安全的 CORS 策略并限制跨域请求，以降低跨域攻击的风险。

### HTTP 安全标头配置错误

**参数：**

- 与安全相关的 HTTP 标头缺失或配置错误，例如内容安全策略 (CSP)、严格传输安全 (HSTS) 或 X-Frame-Options。
- 未能实现正确的 HTTP 安全标头。
- 由于标头配置错误而暴露于常见的 Web 漏洞。

**遵循的步骤：**

1. 分析 HTTP 响应并检查安全相关标头（例如 CSP、HSTS 和 X-Frame-Options）是否存在及其配置。
2. 评估是否实施了适当的安全标头来缓解常见的 Web 漏洞。
3. 检查 HTTP 安全标头是否正确配置，以防止 XSS、点击劫持或协议降级攻击等攻击。
4. 如果安全标头丢失或配置错误，则表明存在 HTTP 安全标头配置错误的漏洞。
5. 实施适当的 HTTP 安全标头和配置，以增强 Web 应用程序的安全状况。

### XML 外部实体 (XXE) 注入

**参数：**

- 缺乏 XML 输入的输入验证和清理。
- 在 XML 文档中使用外部实体没有适当的限制。
- 攻击者执行恶意 XML 代码。
- 易受攻击的 XML 解析库或框架。

**遵循的步骤：**

1. 识别接受 XML 输入的输入字段或参数。
2. 将包含外部实体的有效负载输入到这些字段中。
3. 提交输入并观察 XML 解析器是否处理外部实体。
4. 如果外部实体被解析并执行，则表明存在XXE注入漏洞。
5. 测试不同的XXE负载以利用该漏洞。
6. 实施输入验证并禁用外部实体处理以降低 XXE 注入的风险。

### XML 实体扩展 (XEE)

**参数：**

- XML 文档中的实体扩展不受限制。
- 缺乏针对递归实体扩展的保护。
- 攻击者执行恶意 XML 代码。
- 易受攻击的 XML 解析库或框架。

**遵循的步骤：**

1. 识别允许实体扩展的 XML 文档或输入字段。
2. 将包含递归实体扩展的有效负载输入到这些字段中。
3. 提交输入并观察 XML 解析器是否递归扩展实体。
4. 如果出现递归实体扩展，则表明存在 XEE 漏洞。
5. 测试不同的 XEE 负载以利用该漏洞。
6. 对实体扩展实施限制并禁用递归实体扩展以降低 XEE 风险。

### XML炸弹

**参数：**

- 使用包含恶意制作的元素的 XML 文档。
- 创建旨在消耗资源的大型嵌套 XML 结构。 - 通过 XML 解析执行拒绝服务攻击。

**遵循的步骤：**

1. 分析 XML 文档中的大型嵌套结构或元素。
2. 检查XML 文档是否包含旨在消耗过多资源的实体。
3. 将 XML 文档提交给 XML 解析器并观察资源使用情况。

4. 如果XML解析器消耗过多资源或崩溃，则表明容易遭受XML炸弹攻击。
5. 测试不同的 XML 炸弹有效负载以利用该漏洞。
6. 实施实体扩展限制或 XML 有效负载大小限制等措施，以降低 XML 炸弹攻击的风险。

## 访问控制损坏

### 授权不足

**参数：**

- 缺乏对资源或功能的适当访问控制。
- 缺乏基于角色的访问控制机制。
- 用户权限验证不足。
- 将敏感功能暴露给未经授权的用户。

**遵循的步骤：**

1. 审查访问控制机制以确保其得到正确实施。
2. 评估是否根据用户角色分配角色和权限。
3. 测试用户权限验证中是否存在任何差距或漏洞。
4. 识别未经授权的用户可以访问的敏感功能。
5. 如果发现授权不足，则表明存在未经授权访问的漏洞。
6. 实施适当的访问控制和基于角色的授权以降低风险。

### 权限提升

**参数：**

- 用户有机会将其权限提升到超出其授权级别。 - 缺乏对权限升级尝试的适当验证。
- 脆弱的身份验证机制。
- 向未经授权的用户公开特权功能。

**遵循的步骤：**

1. 确定用户可能提升其权限的功能或流程。
2. 测试是否存在允许未经授权的权限升级的漏洞。
3. 检查身份验证机制是否存在任何可能促进权限升级的弱点。
4. 识别未经授权的用户可以访问的任何特权功能。
5. 如果可能进行权限提升，则表明存在未经授权的权限提升漏洞。
6. 实施严格的验证检查和访问控制，以防止未经授权的权限升级。

# 2024漏洞分类指北手册

## 不安全的直接对象引用

**参数：**

- 无需适当授权即可直接访问对象或资源。
- 缺乏访问控制或授权检查。
- 通过可预测的对象引用暴露敏感数据。

**遵循的步骤：**

1. 识别可通过直接引用访问的资源或对象。
2. 测试是否强制执行授权检查来访问这些资源。
3. 检查敏感数据是否通过可预测的对象引用暴露。
4. 如果可能对敏感数据进行未经授权的访问，则表明存在不安全的直接对象引用的漏洞。
5. 实施适当的访问控制和授权检查，以防止未经授权访问敏感资源。

## 强力浏览

**参数：**

- 在没有适当访问控制的情况下敏感页面或功能的可用性。 - 将内部或隐藏功能暴露给未经授权的用户。
- 缺乏针对暴力尝试访问受限资源的保护。

**遵循的步骤：**

1. 尝试在未经适当身份验证或授权的情况下直接访问页面或功能。
2. 识别可通过直接浏览访问的任何敏感页面或功能。
3. 测试是否存在允许暴力尝试访问受限资源的漏洞。
4. 如果可能发生未经授权的访问，则表明存在强制浏览的漏洞。
5. 实施适当的访问控制和保护机制，限制对敏感资源的未经授权的访问。

## 缺少功能级访问控制

**参数：**

- 访问控制缺乏粒度，导致访问权限过高。
- 功能或操作级别缺乏访问控制。
- 将敏感操作暴露给未经授权的用户。

**遵循的步骤：**

1. 检查访问控制机制以评估访问权限的粒度。
2. 确定是否在功能或操作级别应用访问控制。
3. 测试功能级访问控制中是否存在任何差距或漏洞。
4. 识别未经授权的用户可以访问的敏感操作。
5. 如果访问控制缺乏粒度，则表明存在功能级访问控制缺失的漏洞。
6. 实施细粒度的访问控制，根据用户权限限制对敏感操作的访问。

# 2024漏洞分类指北手册

## 不安全的反序列化

### 通过反序列化远程执行代码

**参数：**

- 使用不安全的反序列化方法或库。
- 接受来自不受信任来源的序列化数据。
- 缺乏反序列化数据的输入验证和清理。
- 执行嵌入序列化对象中的恶意代码。

**遵循的步骤：**

1. 确定应用程序中接受序列化数据的反序列化点。
2. 测试来自不受信任来源的序列化数据是否可以反序列化。
3. 评估是否对反序列化数据执行输入验证和清理。
4. 向序列化对象中注入恶意代码，观察反序列化时是否执行。
5. 如果发生代码执行，则表明存在通过反序列化远程执行代码的漏洞。
6. 实施安全反序列化实践和输入验证以降低风险。

### 数据篡改

**参数：**

- 缺乏对传入数据的完整性检查。
- 传输数据缺乏加密。
- 脆弱的数据验证机制。
- 攻击者未经授权修改数据。

**遵循的步骤：**

1. 监控传入数据是否存在完整性检查或加密缺失的情况。
2. 测试数据验证机制是否可以被绕过或操纵。
3. 未经适当授权，试图修改传输中的数据或存储的数据。
4. 评估是否可能对数据进行未经授权的修改。
5. 如果数据可以被篡改，则表明数据存在被篡改的漏洞。
6. 实施完整性检查、加密和可靠验证以防止数据篡改。

### 对象注入

**参数：**

- 接受来自不受信任来源的序列化对象。
- 缺乏反序列化对象的输入验证和清理。
- 执行嵌入序列化对象中的恶意代码。

**遵循的步骤：**

1. 识别应用程序中从不受信任的来源接受序列化对象的点。

2. 测试是否对反序列化对象执行输入验证和清理。
3. 向序列化对象中注入恶意代码，观察反序列化时是否执行。
4. 检查是否存在允许未经授权的对象注入的漏洞。



5. 如果发生代码执行，则表明存在对象注入漏洞。
6. 实施安全反序列化实践和输入验证以降低风险。

## API安全问题

### 不安全的 API 接口

**参数：**

- API 接口缺乏身份验证或授权机制。
- 通过 API 传输的数据缺乏加密。
- 通过不受保护的 API 接口暴露敏感功能。

**遵循的步骤：**

1. 识别缺乏身份验证或授权机制的 API 接口。
2. 检查API传输的数据是否加密。
3. 寻找可通过未受保护的 API 接口访问的敏感功能。
4. 测试 API 接口是否存在未经授权的访问或数据泄露。
5. 如果接口不安全，则表明不安全的 API 接口存在漏洞。
6. 实施适当的身份验证、授权和加密机制以保护 API 接口的安全。

### API密钥暴露

**参数：**

- 在客户端代码或配置文件中存储 API 密钥。
- 通过网络流量或错误消息泄露 API 密钥。
- 缺乏针对 API 密钥枚举攻击的保护。

**遵循的步骤：**

1. 检查硬编码 API 密钥的客户端代码和配置文件。
2. 监控 API 密钥传输的网络流量。
3. 检查错误消息或响应中是否包含敏感的API密钥信息。
4. 测试允许枚举 API 密钥的漏洞。
5. 如果 API 密钥被暴露，则表明存在 API 密钥暴露漏洞。
6. 实施安全存储实践，防止密钥泄漏，并防止密钥枚举以降低风险。

### 缺乏速率限制

**参数：**

- API接口上缺乏速率限制或节流机制。
- 通过过多的 API 请求而遭受滥用或拒绝服务攻击。

**遵循的步骤：**

1. 识别缺乏速率限制或节流机制的 API接口。
2. 测试 API接口是否存在过多请求或潜在滥用。
3. 检查是否存在允许绕过速率限制控制的漏洞。
4. 监视系统性能和网络流量是否存在拒绝服务攻击的迹象。
5. 如果未实施速率限制，则表明存在缺乏速率限制的漏洞。



6. 实施速率限制控制，以限制 API 请求数量并防止滥用。

**输入验证不足**

**参数：**

- 缺乏对传递到 API接口的输入参数的验证。
- 接受格式错误或意外的输入数据。
- 由于验证不充分，容易受到 SQL 注入或 XSS 等注入攻击。

**遵循的步骤：**

1. 检查 API接口接受的输入参数。
2. 测试输入验证机制是否存在。
3. 向 API接口提供格式错误或意外的输入数据并观察系统行为。
4. 检查是否存在由于输入验证不充分而导致 SQL 注入或 XSS 等漏洞被利用的情况。
5. 如果输入未经过正确验证，则表明存在输入验证不充分的漏洞。
6. 实施强大的输入验证例程，以有效地清理和验证输入数据。

## 不安全的通讯

**中间人 (MITM) 攻击：**

**参数：**

- 异常网络流量模式
- 不寻常的 SSL/TLS 证书更改
- HTTP 请求/响应中的可疑重定向或修改

**遵循的步骤：**

1. 监视网络流量是否有异常模式或未经授权的设备拦截客户端和服务器之间的通信。
2. 查找 SSL/TLS 证书中的意外更改，例如自签名证书或未知颁发机构颁发的证

书。

3. 分析 HTTP 请求和响应是否存在任何重定向或修改迹象，例如意外的 HTTP 状态代码或内容更改。

**传输层安全性不足：**

**参数：**

- 弱加密算法或密码套件
- 缺乏完美的前向保密（PFS）
- 证书验证缺失或薄弱

**遵循的步骤：**

1. 扫描 SSL/TLS 配置中使用的弱加密算法或已弃用的密码套件。
2. 检查 SSL/TLS 配置是否支持完美前向保密 (PFS)，这可确保即使长期私钥受到损害，会话密钥也不会受到损害。



3. 验证 SSL/TLS 客户端是否正确验证服务器证书，并拒绝与具有无效、过期或自签名证书的服务器的连接。

**不安全的 SSL/TLS 配置：**

**参数：**

- SSL/TLS 协议版本较弱
- 缺乏 HTTP 严格传输安全 (HSTS)
- 缺乏安全密码套件

**遵循的步骤：**

1. 确定是否启用了 SSLv2 或 SSLv3 协议，因为已知它们不安全且容易受到攻击。
2. 检查 Web 服务器是否发送 HTTP 严格传输安全 (HSTS) 标头以强制通过 HTTPS 建立安全连接并防止降级攻击。
3. 确保在 SSL/TLS 配置中仅启用强大且安全的密码套件，例如 AES-GCM、AES-CBC 或 ChaCha20-Poly1305。

**不安全的通信协议：**

**参数：**

- 使用不安全的通信协议，例如 HTTP 而不是 HTTPS - 传输中的数据缺乏加密
- 缺乏通信验证机制

**遵循的步骤：**

1. 识别通过不安全协议（例如 HTTP、FTP 或 Telnet）传输敏感数据的实例。

2. 监控网络流量，以检测未加密传输数据的情况，从而导致数据容易被拦截和窃听。
3. 确保通信协议实施强大的身份验证机制，以验证通信方的身份并防止未经授权的访问。

## 客户端漏洞

### 基于 DOM 的 XSS

**参数：**

- 客户端脚本中缺乏适当的输入验证和清理。
- 在文档对象模型 (DOM) 中执行不受信任的用户输入。 - 由于客户端脚本执行而导致 XSS 攻击的漏洞。

**遵循的步骤：**

1. 检查用于输入处理和操作的客户端脚本。
2. 确定将用户输入合并到 DOM 中的区域。
3. 测试允许在 DOM 内执行不受信任输入的漏洞。
4. 将XSS有效负载注入输入字段或参数并观察它们是否被执行。
5. 如果不受信任的输入可以在 DOM 内执行脚本，则表明存在基于 DOM 的 XSS 漏洞。



6. 实施严格的输入验证和输出编码以降低 XSS 风险。 **不安全的跨源通信**

**参数：**

- 缺乏适当的跨源资源共享（CORS）政策。
- 缺乏跨源通信安全机制。
- 由于不安全的通信通道而容易受到跨源攻击。 **遵循的步骤：**

1. 检查在网页或 API 上实施的跨源资源共享 (CORS) 策略。
2. 测试允许未经授权的跨源通信的漏洞。
3. 检查跨源通信安全机制（例如内容安全策略（CSP））是否配置正确。
4. 尝试跨不同来源访问资源，观察是否允许访问。
5. 如果可能发生未经授权的跨域通信，则表明存在不安全跨域通信的漏洞。
6. 实施严格的CORS策略，利用CSP等安全机制限制跨源通信，防止攻击。

### 浏览器缓存中毒

**参数：**

- 缺乏适当的缓存控制标头或指令。
- 由于缓存恶意内容而导致缓存中毒攻击的漏洞。
- 缺乏针对缓存中毒漏洞的保护。

**遵循的步骤：**

1. 分析 HTTP 响应中 Web 服务器发送的缓存控制标头或指令。
2. 测试缓存机制是否容易受到缓存中毒攻击。
3. 检查缓存的内容是否包含潜在的恶意或未经授权的数据。
4. 尝试通过向缓存资源注入恶意内容来毒害浏览器缓存。
5. 如果浏览器缓存可能被未经授权的数据投毒，则表明存在浏览器缓存投毒漏洞。
6. 实施适当的缓存控制标头，验证缓存内容，并应用针对缓存中毒攻击的保护措施来降低风险。

**点击劫持**

**参数：**

- 缺乏针对点击劫持攻击（例如帧破坏）的保护。
- 缺少 X-Frame-Options 标头或frame-ancestors 指令。
- 存在通过点击劫持进行 UI 修复攻击的漏洞。

**遵循的步骤：**

1. 分析网页以确定它们是否容易受到点击劫持攻击。
2. 测试是否存在帧破坏技术或保护机制。
3. 检查X-Frame-Options header 或frame-ancestors 指令是否正确实现。



4. 尝试在合法网页之上叠加欺骗性内容并诱骗用户点击隐藏元素。
5. 如果网页可以被操纵以执行意外操作，则表明存在点击劫持漏洞。
6. 实施框架破坏技术，设置 X-Frame-Options 标头，或使用框架祖先指令来防止点击劫持攻击并保护用户交互。

**HTML5 安全问题**

**参数：**

- 使用已弃用或不安全的 HTML5 功能。
- 容易遭受各种攻击，例如 XSS、CSRF 或 DOM 操作。
- 缺乏对 HTML5 元素和 API 的适当安全控制。

**遵循的步骤：**

1. 回顾 Web 应用程序中使用的 HTML5 特性和功能。
2. 测试与 HTML5 功能相关的漏洞，例如 XSS 或 CSRF。
3. 检查是否对 HTML5 元素和 API 实施了适当的安全控制。
4. 评估是否使用了已弃用或不安全的 HTML5 功能。
5. 如果Web应用程序容易受到攻击或缺乏适当的安全控制，则表明HTML5存在安全问题。
6. 更新以保护 HTML5 功能、实施安全控制并执行彻底的安全测试以减轻 HTML5 相关风险。

### 分布式拒绝服务（DDoS）

**参数：**

- 网络流量或网络服务器请求出现异常峰值。
- 可疑的僵尸网络活动或针对单个服务器的多个流量源。
- 服务器资源不堪重负，导致服务中断或停机。

**遵循的步骤：**

1. 监控网络流量模式并查找流量突然增加的情况。
2. 分析服务器日志以查找来自不同 IP 地址的多个请求的指示。
3. 评估高峰流量期间的服务器性能，以确定资源耗尽情况。
4. 调查任何有关服务中断或停机的报告。
5. 如果存在协同攻击或流量过大的迹象，则表明容易遭受 DDoS 攻击。
6. 实施 DDoS 缓解策略，例如流量过滤、速率限制或部署 DDoS 防护服务来降低风险。

### 应用层拒绝服务

**参数：**

- 针对特定应用程序功能或端点的有针对性的攻击。
- 故意利用漏洞来耗尽应用程序资源。
- 由于过载而导致关键应用程序功能不可用。





**遵循的步骤：**

1. 识别容易受到 DoS 攻击的关键应用程序功能或端点。
2. 测试是否存在允许利用耗尽应用程序资源的漏洞。
3. 在高峰使用期间监控应用程序性能是否存在过载迹象。
4. 调查有关用户因应用程序过载而无法访问关键功能的所有报告。
5. 如果有迹象表明有针对性的攻击或资源耗尽，则表明存在应用层 DoS 攻击的漏洞。
6、实施限速、请求验证、负载均衡等措施，防御应用层DoS攻击。

### 资源枯竭

**参数：**

- 持续消耗系统资源，例如CPU、内存或磁盘空间。
- 可疑的恶意活动或滥用导致资源耗尽。
- 由于资源耗尽导致系统不稳定或性能下降。

**遵循的步骤：**

1. 监控系统资源使用指标，例如 CPU 利用率、内存消耗和磁盘空间可用性。
2. 分析系统日志是否存在任何异常或持续的资源消耗模式。
3. 评估系统性能和稳定性是否有退化或不稳定的迹象。
4. 调查因资源耗尽而导致系统速度减慢或故障的所有报告。
5. 如果资源持续耗尽或系统性能受到影响，则表明容易遭受资源耗尽攻击。
6. 实施资源管理策略，例如设置资源限制、优化代码效率以及实施监控和警报系统以降低资源耗尽的风险。

**斯洛洛里斯攻击 （Slowloris）是一种低带宽拒绝服务攻击（DoS/DDoS）的变种**

**参数：**

- 通过发送部分 HTTP 请求故意减慢服务器响应时间。
- 尝试利用服务器对并发连接的限制。
- 由于请求处理时间过长而导致服务器运行中断。

**遵循的步骤：**

1. 监视服务器日志中来自客户端 IP 地址的不完整或部分 HTTP 请求。
2. 分析服务器性能指标以了解响应时间增加或服务器超时的迹象。
3. 测试不同负载下的服务器响应时间，以确定性能瓶颈。
4. 调查用户遇到服务器行为缓慢或无响应的任何报告。
5. 如果有迹象表明请求处理时间延长或服务器速度变慢，则表明存在 Slowloris 攻击的漏洞。
6. 实施缓解技术，例如连接速率限制、服务器端超时或部署入侵检测系统来检测和防止 Slowloris 攻击。





**XML 拒绝服务**

**参数：**

- 利用 XML 解析器漏洞导致资源耗尽。
- 发送精心设计的 XML 有效负载，旨在压倒解析器功能。
- 由于处理过载而导致基于 XML 的服务或应用程序中断。

**遵循的步骤：**

1. 分析 XML 解析器实现中的已知漏洞或弱点。
2. 使用旨在触发资源耗尽的精心设计的有效负载来测试 XML 解析功能。
3. 在 XML 数据处理期间监视系统性能是否存在过载或速度减慢的迹象。
4. 调查有关基于 XML 的服务或应用程序在负载下变得无响应或失败的任何报告。

5. 如果XML解析操作导致资源耗尽或服务中断，则表明存在XML DoS攻击的漏洞。
6. 实施安全的 XML 解析实践、输入验证和 XML 有效负载大小限制，以防御 XML DoS 攻击。

## 其他网络漏洞

### 服务器端请求伪造 (SSRF)

**参数：**

- 接受用户为服务器端请求提供的 URL。
- 缺乏 URL 参数的输入验证。
- 能够通过 SSRF 攻击访问内部资源。
- 由于未经授权的服务器端请求而暴露敏感数据或服务。

**遵循的步骤：**

1. 识别接受服务器端请求 URL 的端点或功能。
2. 测试输入验证是否应用于 URL 参数。
3. 尝试通过受操纵的 URL 访问内部资源或敏感服务。
4. 检查对内部系统或敏感数据提出的未经授权的请求。
5. 如果可能出现未经授权的服务器端请求，则表明存在 SSRF 攻击漏洞。
6. 实施严格的输入验证并强制将允许的 URL 列入白名单，以防止 SSRF 漏洞。

### HTTP 参数污染 (HPP)

**参数：**

- HTTP 请求中多次出现相同的参数。
- 缺乏对重复或冲突参数的正确处理。
- 由于参数污染而可能对应用程序逻辑或数据进行操纵。

**遵循的步骤：**

1. 分析HTTP请求中是否多次出现相同的参数。



2. 测试应用程序是否正确处理重复或冲突的参数。

3. 尝试通过使用冲突值污染参数来操纵应用程序行为或数据。
4. 检查是否存在参数污染导致的意外结果或错误。
5. 如果应用程序行为或数据可以通过参数污染来操纵，则表明存在 HPP 漏洞。
6. 实施严格的参数处理和验证，防止参数污染攻击。

### 不安全的重定向和转发

**参数：**

- 使用未经验证或用户控制的重定向或转发 URL。
- 缺乏对重定向和转发参数的正确验证。
- 容易受到网络钓鱼攻击或通过操纵重定向进行未经授权的访问。

**遵循的步骤：**

1. 确定接受用户控制的 URL 的重定向或转发功能。
2. 测试输入验证是否应用于重定向和转发参数。
3. 尝试操纵重定向或将 URL 转发到未经授权的目的地。
4. 检查是否存在潜在的网络钓鱼尝试或通过操纵重定向而导致的未经授权的访问。
5. 如果用户可以被重定向到未经授权的位置，则表明存在不安全重定向和转发的漏洞。
6. 对重定向和转发 URL 实施严格验证，以降低未经授权的访问或网络钓鱼攻击的风险。

**文件包含漏洞**

**参数：**

- 包含来自用户提供的或不受信任来源的文件。
- 缺乏文件包含参数的输入验证和清理。
- 可能执行任意代码或泄露敏感信息。

**遵循的步骤：**

1. 识别包含来自用户提供或不受信任来源的文件的功能。
2. 测试输入验证和清理是否应用于文件包含参数。
3. 尝试包含任意文件或操纵文件路径来访问敏感资源。
4. 检查是否存在意外的文件泄露或因文件包含而执行的任意代码。
5. 如果可以包含任意文件或暴露敏感信息，则表明存在文件包含漏洞。
6. 实施严格的输入验证并对文件包含实施限制以防止攻击。

**安全标头绕过**

**参数：**

- 安全标头的缺失或实施较弱，例如内容安全策略 (CSP) 或 X-Content-Type-Options。



- 由于缺乏适当的标头实施而容易受到各种攻击。

- 遭受 XSS、点击劫持或 MIME 嗅探攻击。

**遵循的步骤：**

1. 检查已实施的安全标头，例如 CSP 或 X-Content-Type-Options。

2. 测试安全标头是否正确实施和配置。

3. 尝试绕过安全控制或利用由于弱标头实现而导致的漏洞。
4. 检查是否存在允许 XSS、点击劫持或 MIME 嗅探攻击的漏洞。
5. 如果安全标头可以被绕过或漏洞被利用，则表明存在安全标头绕过漏洞。
6. 实施安全标头的安全配置并执行严格的标头策略，以防止绕过攻击并降低相关风险。

**点击劫持**

**参数：**

- 缺乏针对点击劫持攻击（例如帧破坏）的保护。
- 缺少 X-Frame-Options 标头或frame-ancestors 指令。
- 存在通过点击劫持进行 UI 修复攻击的漏洞。

**遵循的步骤：**

1. 分析网页以确定它们是否容易受到点击劫持攻击。
2. 测试是否存在帧破坏技术或保护机制。
3. 检查X-Frame-Options header 或frame-ancestors 指令是否正确实现。
4. 尝试在合法网页之上叠加欺骗性内容并诱骗用户点击隐藏元素。
5. 如果网页可以被操纵以执行意外操作，则表明存在点击劫持漏洞。
6. 实施框架破坏技术，设置 X-Frame-Options 标头，或使用框架祖先指令来防止点击劫持攻击并保护用户交互。

**会话超时不足**

**参数：**

- 会话超时设置不足，导致会话持续时间延长。
- 缺乏在不活动一段时间后自动终止会话的功能。
- 由于会话生命周期延长，容易受到会话劫持或未经授权的访问。

**遵循的步骤：**

1. 检查会话管理设置和会话超时配置。
2. 测试会话是否长时间保持活动状态而不超时。
3. 检查会话是否在一段时间不活动后自动终止。
4. 尝试劫持活动会话或由于延长会话生命周期而访问受限资源。
5. 如果会话持续超过定义的超时期限，则表明存在会话超时不足的漏洞。



6. 实施适当的会话超时设置和自动会话终止机制，以降低会话劫持和未经授权访问的风险。

**日志记录和监控不足**

**参数：**

- 缺乏应用程序活动和安全事件的全面日志记录。

- 对可疑或异常行为的监控不足。
- 由于可见性有限，难以检测和响应安全事件。

**遵循的步骤：**

1. 检查日志记录机制和配置，确保它们捕获相关的应用程序活动和安全事件。
2. 评估监控系统检测可疑或异常行为的能力。
3. 检查日志记录覆盖范围是否存在差距或日志数据的粒度是否不足。
4. 监视由于日志记录和监视不足而未被注意到的系统行为异常或安全事件。
5. 如果对系统活动或安全事件的可见性不足，则表明存在日志记录和监控不足的漏洞。
6. 实施全面的日志记录实践，增强监控能力，并建立强大的事件响应程序，以提高安全事件的可见性和响应能力。

**业务逻辑漏洞：**

**参数：**

- 与预期业务流程的偏差
- 数据验证和处理不一致
- 未经授权访问敏感功能

**遵循的步骤：**

1. 分析应用程序工作流程并识别与预期业务逻辑的任何偏差。

2. 测试输入验证机制以发现数据处理中的不一致或绕过情况。

3. 执行安全测试，确保敏感功能得到适当保护，防止未经授权的访问。

**API 滥用：**

**参数：**

- API请求过多或异常
- 未经授权访问受限 API接口
- API 使用模式发生意外变化

**遵循的步骤：**

1. 监控 API 流量是否存在异常情况，例如来自单一来源的大量请求或异常请求模式。
2. 审核 API接口，确保采用适当的身份验证和授权机制来限制对敏感数据和功能的访问。





3. 实施速率限制和访问控制，以降低 API 滥用的风险并执行使用政策。

## 移动设备上的不安全数据存储

**参数：**

- 将敏感数据存储在移动设备上不安全的位置，例如不受保护的本地存储或不安全的数据库。
- 移动设备上存储的数据缺乏加密或混淆。
- 由于不安全的存储实践，数据容易被盗或未经授权的访问。

**遵循的步骤：**

1. 分析移动应用程序在设备上存储的敏感数据。
2. 评估数据存储所实施的安全措施，例如加密和访问控制。
3. 测试是否存在允许未经授权访问存储数据或敏感数据存在于不安全位置的漏洞。
4. 尝试通过文件系统访问或其他方式提取或操作存储的数据。
5. 如果敏感数据存储不安全或容易受到未经授权的访问，则表明移动设备上的数据存储实践存在漏洞。
6. 实施安全存储机制，包括加密和适当的访问控制，以保护敏感数据免遭未经授权的访问或盗窃。

## 移动设备上的不安全数据传输

**参数：**

- 通过不安全的通道传输敏感数据，例如未加密的 HTTP 连接或不安全的网络协议。
- 移动设备和服务器之间传输的数据缺乏 TLS/SSL 加密。
- 传输数据容易被拦截和窃听。

**遵循的步骤：**

1. 监控移动设备和服务器之间的网络流量以传输敏感数据。
2. 评估数据传输所实施的安全措施，例如 TLS/SSL 加密。
3. 测试允许拦截或操纵传输数据的漏洞。
4. 试图拦截或窃听通过不安全通道传输的数据。
5. 如果敏感数据的传输没有适当的加密或容易被拦截，则表明移动设备上的数据传输实践存在漏洞。
6. 实施安全通信协议并强制执行 TLS/SSL 加密，以保护传输过程中的敏感数据并防止拦截攻击。

## 不安全的移动 API接口

**参数：**

- 移动 API接口缺乏身份验证或授权机制。

- 通过不受保护的移动 API接口暴露敏感功能。

- 未经授权的访问或滥用移动 API 的漏洞。

**遵循的步骤：**

1. 识别缺乏身份验证或授权机制的移动 API接口。
2. 测试是否可以通过不受保护的移动 API接口访问敏感功能。
3. 评估移动 API 上的访问控制和数据保护所实施的安全措施。
4. 尝试通过不安全的移动 API接口访问受限功能或操纵数据。
5. 如果可能出现未经授权的访问或滥用移动 API，则表明移动 API接口安全存在漏洞。
6. 实施适当的身份验证、授权和访问控制机制，以保护移动 API接口并防止未经授权的访问或滥用。

**移动应用逆向工程**

**参数：**

- 移动应用程序中源代码或敏感信息的可用性。
- 移动应用程序缺乏混淆或防篡改措施。
- 逆向工程攻击的漏洞，包括代码提取和分析。

**遵循的步骤：**

1. 分析移动应用程序是否存在可逆向工程的敏感信息或代码。
2. 评估移动应用程序中实施的混淆和防篡改措施的水平。
3. 测试是否存在允许从移动应用程序提取或操纵源代码和敏感数据的漏洞。
4. 尝试对移动应用程序进行逆向工程以发现专有算法或安全漏洞。
5. 如果源代码或敏感信息可以轻松提取或逆向工程，则表明移动应用程序存在安全漏洞。
6. 实施强大的混淆技术、防篡改措施和代码保护机制，以防止逆向工程攻击并保护移动应用程序中的知识产权和敏感数据。

## 物联网网络漏洞

**不安全的物联网设备管理**

**参数：**

- 物联网设备缺乏安全的管理界面，例如不安全的门户网站或 API。

- 缺乏适当的设备管理访问控制或身份验证机制。

- 未经授权访问或操纵物联网设备的漏洞。

**遵循的步骤：**

1. 确定用于物联网设备管理的管理接口或协议。

2. 测试是否有用于访问设备管理界面的安全身份验证机制。
3. 评估访问控制的实施情况，以限制对设备管理功能的未经授权的访问。
4. 尝试通过不安全的管理界面未经授权访问物联网设备或操纵设备设置。
5. 如果可能发生未经授权的访问或操纵，则表明物联网设备管理存在漏洞。
6. 实施安全管理界面，实施强身份验证，并应用访问控制来保护物联网设备免遭未经授权的访问和操纵。

**物联网设备的弱身份验证**

**参数：**

- 使用默认或弱凭据访问物联网设备。
- 缺乏密码策略或强制执行强身份验证措施。
- 物联网设备上存在凭证猜测或暴力攻击的漏洞。

**遵循的步骤：**

1. 识别 IoT 设备使用的身份验证机制，例如用户名/密码或基于令牌的身份验证。
2. 测试是否使用默认凭据或强制执行弱密码策略。
3. 评估身份验证机制的强度及其对凭证猜测或暴力攻击的敏感性。
4. 尝试利用弱身份验证措施来获得对物联网设备的未经授权的访问。
5. 如果由于认证较弱而导致未经授权的访问，则表明物联网设备认证存在漏洞。
6. 实施强密码策略，强制执行密码复杂性要求，并部署多重身份验证，以增强物联网设备身份验证的安全性。

**物联网设备漏洞**

**参数：**

- 物联网设备固件或软件中存在已知漏洞。
- 缺乏用于解决漏洞的安全补丁或更新。
- 容易被恶意行为者利用进行未经授权的访问或控制。

**遵循的步骤：**

1. 分析 IoT 设备固件或软件中的已知漏洞，例如 CVE 条目或安全建议。
2. 评估用于解决已识别漏洞的安全补丁或更新的可用性。
3. 通过尝试已知的攻击技术或漏洞来测试物联网设备对漏洞的敏感性。
4. 监控网络流量和设备行为，以发现未经授权的访问或控制迹象。
5. 如果存在漏洞并且可被利用，则表明物联网设备安全存在漏洞。

6. 应用安全补丁和更新、实施网络分段并采用入侵检测系统来降低物联网设备漏洞和未经授权访问的风险。

## 物联网（WoT）漏洞

### 未经授权访问智能家居

**参数：**

- 访问智能家居设备的身份验证机制较弱或默认。
- 智能家居通信缺乏适当的访问控制或加密。
- 攻击者或恶意行为者容易遭受未经授权的访问。

**遵循的步骤：**

1. 识别智能家居设备使用的身份验证机制，例如密码或生物识别身份验证。
2. 测试是否使用默认或弱凭据，或者是否完全绕过身份验证。
3. 评估用于保护智能家居通信的访问控制和加密的实施情况。
4. 尝试通过暴力攻击、绕过身份验证或利用通信漏洞来获取对智能家居设备的未经授权的访问。
5. 如果存在未经授权的访问，则表明智能家居安全存在漏洞。 6. 实施强大的身份验证机制，实施访问控制，并使用加密来保护智能家居设备免受未经授权的访问。

### 物联网数据隐私问题

**参数：**

- 物联网数据传输和存储的数据加密或匿名化不足。
- 缺乏用户同意或对数据收集和共享的控制。
- 容易遭受数据泄露或未经授权的数据访问。

**遵循的步骤：**

1. 分析物联网数据的传输和存储方式，评估数据加密和匿名化的实施情况。
2. 审查数据收集和共享的用户同意机制和控制措施。
3. 测试允许未经授权访问物联网数据或数据泄露的漏洞。
4. 监控网络流量和系统日志是否存在数据泄露或未经授权访问敏感数据的迹象。
5. 如果数据隐私措施不完善，则表明物联网数据隐私存在漏洞。
6. 实施强大的数据加密、匿名技术和用户同意控制，以保护物联网数据隐私并防止未经授权的访问或数据泄露。

# 2024漏洞分类指北手册

## 身份验证绕过

### 不安全的“记住我”功能

**参数：**

- 身份验证机制中“记住我”功能的实施薄弱或不充分。
- 缺乏适当的会话管理或“记住我”令牌过期。
- 容易遭受未经授权的访问或会话劫持。

**遵循的步骤：**

1. 确定为用户提供“记住我”选项的身份验证机制。
2. 测试是否实施了适当的安全措施来管理“记住我”令牌，例如令牌过期或安全存储。
3. 评估“记住我”功能对会话劫持或未经授权访问的敏感性。
4. 尝试利用“记住我”功能的弱点来获得未经授权的访问或劫持用户会话。
5. 如果可能发生未经授权的访问或会话劫持，则表明“记住我”功能存在漏洞。
6. 实施强大的会话管理实践，强制令牌过期，并使用安全存储机制来降低通过“记住我”功能进行未经授权访问的风险。

### 验证码绕过

**参数：**

- CAPTCHA 实施中的漏洞，例如可预测或较弱的质询响应机制。
- 验证码响应缺乏适当的验证或执行。
- 自动或手动绕过技术的漏洞。

**遵循的步骤：**

1. 分析 CAPTCHA 实施中挑战-响应机制中的弱点或漏洞。
2. 测试验证码响应是否经过正确验证和强制执行，以防止自动或手动绕过。
3. 评估验证码系统对绕过技术（例如基于 OCR 的攻击或手动利用）的敏感性。
4. 尝试使用自动化工具或手动方法绕过验证码保护。
5. 如果验证码保护可以被绕过，则表明验证码安全存在漏洞。
6. 实施更强有力的挑战-响应机制，改进响应验证并使用
   额外的安全措施可防止验证码绕过攻击并增强整体安全性。

再说安全.公众号

2024漏洞分类指北手册

## 服务器端请求伪造 (SSRF)

### 盲SSRF

**参数：**

- 利用 SSRF 漏洞时，服务器缺乏即时反馈或响应。
- 由于盲目利用技术，难以检测 SSRF 攻击。

**遵循的步骤：**

1. 识别应用程序或网络中潜在的 SSRF 漏洞。
2. 测试 SSRF 端点或功能是否缺乏即时响应或反馈。
3. 尝试通过SSRF攻击访问内部资源或服务。
4. 监视网络流量和服务器日志以查找 SSRF 利用的迹象。
5. 如果 SSRF 攻击成功且没有立即反馈，则表明存在盲目 SSRF 漏洞。
6. 实施安全控制和监控机制，以检测和防止盲目 SSRF 攻击，例如网络防火墙、输入验证和综合日志记录。

### 基于时间的盲SSRF

**参数：**

- 利用 SSRF 漏洞时，服务器缺乏即时反馈或响应。
- 由于基于时间的利用技术，难以检测 SSRF 攻击。

**遵循的步骤：**

1. 识别应用程序或网络中潜在的 SSRF 漏洞。
2. 测试 SSRF 端点或功能是否缺乏即时响应或反馈。
3. 尝试基于时间的利用技术来检测 SSRF 漏洞。
4. 监视 SSRF 攻击尝试期间的服务器响应时间或延迟。
5. 如果 SSRF 攻击导致响应延迟或超时，则表明存在基于时间的盲 SSRF 漏洞。
6. 实施安全控制和监控机制，以检测和防止基于时间的盲目 SSRF 攻击，例如速率限制、超时阈值和异常检测。

## 内容欺骗

### MIME 嗅探

**参数：**

- 对用户提供的文件缺乏正确的 MIME 类型验证。
- 缺少 Content-Disposition 标头或严格的文件类型强制执行。
- 内容类型误解或恶意文件执行的漏洞。

**遵循的步骤：**

1. 分析文件上传功能或内容服务端点以进行 MIME 类型验证。

2. 测试是否在没有正确的 Content-Disposition 标头或 MIME 类型验证的情况下提供文件。



3. 尝试上传或提供 MIME 类型不正确或意外的文件。

4. 监视服务器响应或文件处理是否有 MIME 嗅探行为的迹象。

5. 如果根据推断的 MIME 类型处理或执行文件，则表明存在 MIME 嗅探漏洞。
6. 实施严格的 MIME 类型验证、强制执行 Content-Disposition 标头并限制文件执行，以防止 MIME 嗅探攻击并防止恶意文件执行。

**X-内容类型-选项绕过**

**参数：**

- 缺乏 X-Content-Type-Options 标头的正确实现或强制执行。

- 缺少内容类型的服务器端验证。
- MIME 嗅探或内容类型误解攻击的漏洞。

**遵循的步骤：**

1. 检查服务器配置和 HTTP 响应是否存在 X-Content-Type Options 标头。
2. 测试标头是否正确实施并防止 MIME 嗅探。
3. 尝试通过操纵文件扩展名或内容来绕过内容类型限制。
4. 监视浏览器行为或服务器响应是否有 MIME 类型推断或内容类型绕过的迹象。
5. 如果内容类型被推断或绕过，则表明存在 X-Content-Type-Options 绕过漏洞。
6. 实施严格的服务器端验证，使用“nosniff”指令强制执行 X-Content-Type-Options 标头，并避免依赖 MIME 类型推断来防止内容类型绕过攻击并增强安全性。

**内容安全策略 (CSP) 绕过**

**参数：**

- 内容安全策略指令实施不充分或配置错误。 - 缺乏适当的实施或覆盖来防止 XSS 攻击。 - 通过策略绕过执行脚本或注入的漏洞。

**遵循的步骤：**

1. 检查 Web 应用程序配置和 HTTP 标头是否存在内容安全策略 (CSP) 指令。
2. 测试 CSP 指令是否正确配置和执行以防止 XSS 攻击。
3. 尝试利用错误配置或策略弱点来绕过 CSP 限制。
4. 监视浏览器行为或服务器响应，以发现 CSP 绕过或违反策略的迹象。

5. 如果不顾 CSP 指令执行或注入脚本，则表明存在 CSP 绕过漏洞。
6. 实施严格的 CSP 配置，强制执行安全标头，并定期审核和更新 CSP 策略，以降低 XSS 攻击的风险并防止 CSP 绕过。



# 2024漏洞分类指北手册

## 业务逻辑缺陷

### 验证不一致

**参数：**

- 应用程序的不同部分缺乏一致的验证检查。 - 验证规则或行为的变化导致数据处理不一致。 - 容易出现数据完整性问题或意外行为。

**遵循的步骤：**

1. 检查整个应用程序中实施的验证机制和规则。
2. 跨不同功能或模块测试各种输入场景。
3. 识别验证行为或结果中的不一致或差异。
4. 尝试利用验证规则的变化来操纵或绕过输入检查。
5. 监视因验证不一致而导致的意外行为或数据完整性问题。
6. 如果验证规则不一致或变化，则表明存在验证不一致的漏洞。
7. 实施标准化验证流程，确保验证规则的一致性，并进行彻底的测试，以保持数据完整性并防止因验证不一致而产生漏洞。

### 竞赛条件

**参数：**

- 并发访问共享资源或代码的关键部分。
- 多线程或分布式系统中缺乏适当的同步或锁定机制。
- 由于竞争条件而导致数据损坏或未经授权的访问的漏洞。 **遵循的步骤：**

1. 确定应用程序中多个进程或线程同时访问共享资源的区域。
2. 回顾多线程或分布式系统中实现的同步机制和锁定策略。
3. 测试多个进程或线程竞争访问关键资源的场景。
4. 尝试操纵时序或竞争条件以获得未经授权的访问或损坏数据。
5. 监视由于竞争条件导致的意外行为或数据不一致。
6. 如果并发访问下发生数据损坏或未经授权的访问，则表明存在竞争条件漏洞。
7. 实施适当的同步机制、使用事务处理并实施访问控制，以减轻竞争条件风险并维护数据完整性。

**订单处理漏洞**

**参数：**

- 订单处理工作流程中的漏洞，例如订单数据验证不足。



- 缺乏订单处理功能的授权或访问控制。 - 订单操纵或未经授权交易的漏洞。

**遵循的步骤：**

1. 分析应用程序中的订单处理工作流程和相关功能。
2. 测试订单数据是否经过正确验证，确保其真实性、完整性和授权。
3. 检查订单处理功能的访问控制和权限。
4. 试图操纵订单数据或绕过授权控制来下未经授权的订单。
5. 监控订单状态的意外变化或未经授权的交易。
6. 如果订单可被操纵或发生未经授权的交易，则表明订单处理存在漏洞。
7. 实施强大的验证检查，执行严格的授权控制，并实施交易保障措施，以防止订单处理漏洞和未经授权的交易。

**价格操纵**

**参数：**

- 定价机制中的漏洞，例如缺乏输入验证或授权检查不足。
- 缺乏审计跟踪或价格变化监控。
- 容易受到价格操纵或未经授权的折扣的影响。

**遵循的步骤：**

1. 检查应用程序中的定价机制和相关功能。
2. 测试定价输入数据是否经过正确验证和授权。
3. 评估审计跟踪或价格变化监控的实施情况。
4. 试图操纵定价数据或绕过授权控制来改变价格。
5. 监控价格的意外变化或未经授权的折扣。
6. 如果价格可以被操纵或出现未经授权的折扣，则表明定价机制存在漏洞。
7. 严格输入验证，执行授权控制，对价格变化实施全面监控，防止价格操纵漏洞和未经授权的折扣。

**账户枚举**

**参数：**

- 缺乏正确的错误处理或有效和无效帐户标识符的响应区分。
- 现有和不存在帐户的不同响应或时间变化。
- 帐户枚举或用户名收集的漏洞。

**遵循的步骤：**

1. 测试帐户相关功能，例如登录、注册或密码重置，以实现差异化响应。



2. 分析错误消息或响应以确定它们是否提供有关帐户有效性的线索。
3. 监控有效和无效帐户标识符之间的时间差异的响应时间或行为。
4. 尝试通过利用差异响应或时间变化来枚举用户帐户或获取用户名。
5. 监视帐户访问或枚举尝试的异常模式。
6. 如果可以进行帐户枚举或存在时间差异，则表明帐户枚举存在漏洞。
7. 实施标准化错误响应，避免时间变化，并强制执行一致的错误处理，以防止帐户枚举漏洞并保护用户隐私。

**基于用户的缺陷**

**参数：**

- 观察系统内的用户行为和交互。
- 分析访问控制和权限。
- 审查密码政策和数据处理程序。

**遵循的步骤：**

1. 开展用户培训和意识计划，以教育用户有关安全最佳实践的知识。
2. 实施强有力的访问控制并执行最小权限原则。
3. 定期审核和审查用户活动和权限。
4. 通过模拟场景测试用户对社会工程攻击的敏感度。
5. 监控异常用户行为或由用户操作引起的安全事件。

## 零日漏洞

**未知漏洞**

**参数：**

- 采取主动安全措施，例如漏洞扫描和渗透测试。
- 随时了解新出现的威胁和安全建议。
- 采用强大的监控和异常检测系统。

**遵循的步骤：**

1. 实施漏洞扫描和渗透测试，以在潜在漏洞被利用之前识别它们。
2. 及时了解安全建议和威胁情报来源，以了解潜在的风险领域。
3. 部署强大的监控和异常检测系统，以检测任何可疑行为或妥协迹象。
4. 定期评估系统配置和架构是否存在潜在漏洞。

# 2024漏洞分类指北手册

## 未修补的漏洞

**参数：**

- 监控供应商和安全公告以获取可用的补丁和更新。 - 评估系统配置和补丁管理流程。
- 进行漏洞评估并根据风险确定修补的优先级。

**遵循的步骤：**

1. 建立补丁管理流程，定期应用供应商提供的补丁和更新。
2. 监控供应商和安全公告，随时了解可用补丁。
3. 进行漏洞评估，识别未修补的漏洞，并根据风险确定修补的优先级。
4. 实施变更管理程序，确保及时有效的补丁部署。

## 零日漏洞

**参数：**

- 监控安全研究和威胁情报源以发现新出现的漏洞。

- 评估潜在攻击媒介的系统配置和架构。 - 部署入侵检测和预防系统。

**遵循的步骤：**

1. 及时了解安全研究、威胁情报来源和漏洞数据库，以识别新出现的漏洞。
2. 评估系统配置和架构以了解潜在的攻击媒介。
3. 部署入侵检测和防御系统来检测和阻止攻击尝试。
4. 实施安全控制和缓解措施，以减少零日攻击的影响，例如网络分段和应用程序防火墙。