**渗透测试全流程**

### 免责声明：网络安全法



本文档仅供内部学习参考，任何人利用其中知识违法行为均与文档编辑者无关。

## 渗透测试说明

渗透测试分为**白盒测试**和**黑盒测试**：

白盒测试：已知源码

黑盒测试：仅知资产名称

## ****准备期间须知****

1.    注入漏洞，只要证明可以读取数据就行，严禁读取表内数据。对于UPDATE、DELETE、INSERT 等注入类型，不允许使用自动化工具进行测试。

2.    越权漏洞，越权读取的时候，能读取到的真实数据不超过5组，严禁进行批量读取。

3.    帐号可注册的情况下，只允许用自己的2个帐号验证漏洞效果，不要涉及线上正常用户的帐号，越权增删改，请使用自己测试帐号进行。  
帐号不可注册的情况下，如果获取到该系统的账密并验证成功，如需进一步安全测试，请咨询SRC运营人员得到同意后进行测试。

4.    存储xss漏洞，正确的方法是插入不影响他人的测试payload，严禁弹窗，推荐使用console.log，再通过自己的另一个帐号进行验证，提供截图证明。对于盲打类xss，仅允许外带domain信息。所有xss测试，测试之后需删除插入数据，如不能删除，请在漏洞报告中备注插入点。

5.    如果可以shell或者命令执行的，推荐上传一个文本证明，如纯文本的1.php、1.jsp等证明问题存在即可，禁止下载和读取服务器上任何源代码文件和敏感文件，不要执行删除、写入命令，如果是上传的webshell，请写明shell文件地址和连接口令。

6.    在测试未限制发送短信或邮件次数等扫号类漏洞，测试成功的数量不超过50个。如果用户可以感知，例如会给用户发送登陆提醒短信，则不允许对他人真实手机号进行测试。

7.    如需要进行具有自动传播和扩散能力漏洞的测试（如社交蠕虫的测试），只允许使用和其他账号隔离的小号进行测试。不要使用有社交关系的账号，防止蠕虫扩散。

8.    禁止对网站后台和部分私密项目使用扫描器。

9.    除特别获准的情况下，严禁与漏洞无关的社工，严禁进行内网渗透。

10.  禁止进行可能引起业务异常运行的测试，例如：IIS的拒绝服务等可导致拒绝服务的漏洞测试以及DDOS攻击。

11.  请不要对未授权厂商、未分配给自己的项目、超出测试范围的列表进行漏洞挖掘，可与SRC运营人员联系确认是否属于资产范围后进行挖掘，否则未授权的法律风险将由漏洞挖掘者自己承担。

12.  禁止拖库、随意大量增删改他人信息，禁止可对服务稳定性造成影响的扫描、使用漏洞进行黑灰产行为等恶意行为。

13.  敏感信息的泄漏会对用户、厂商及上报者都产生较大风险，禁止保存和传播和业务相关的敏感数据，包括但不限于业务服务器以及Github 等平台泄露的源代码、运营数据、用户资料等，若存在不知情的下载行为，需及时说明和删除。

14、尊重《中华人民共和国网络安全法》的相关规定。禁止一切以漏洞测试为借口，利用安全漏洞进行破坏、损害用户利益的行为，包括但不限于威胁、恐吓SRC要公开漏洞或数据，请不要在任何情况下泄露漏洞测试过程中所获知的任何信息，漏洞信息对第三方披露请先联系SRC获得授权。企业将对违法违规者保留采取进一步法律行动的权利。

## ****第一阶段：信息收集****

### ****1. 互联网暴露面信息收集****

什么是互联网暴露面?

#### ****管理入口地址获取****

site:xxx.com intext:管理、后台、登录、用户名、密码、系统、帐号

site:xxx.com inurl:login、admin、manage、manager、admin\_login、system

site:xxx.com intitle:管理、后台、登陆

#### ****上传漏洞搜索****

site:xxx.com inurl:file

site:xxx.com inurl:upload

#### ****SQL注入点寻找****

site:xxx.com inurl:php?id=

仅用此语法即可用于发现一个漏洞，敏感信息泄露漏洞，如学生、员工身份证信息，甚至企业OA系统的管理员账号密码。如果是超级管理员权限，还可以提升漏洞危害到中危

### ****2. 域名查询****

例如baidu.com就是百度的主域名，既然有主域名，当然也有子域名，例如www.baidu.com(百度浏览器)和zhidao.baidu.com(知道)

### ****3. IP地址查询****

经典漏洞有：

| **端口** | **漏洞** |
| --- | --- |
| 873 | rsync未授权访问 |
| 6379 | Redis未授权访问 |
| 3389 | 永恒之蓝漏洞 |
| 3306 | MySQL弱口令 |

命令提示符输入ping baidu.com

### ****4. 中间件、框架漏洞****

例如：

* **Apache Tomcat**
* **若依管理系统**
* **金蝶云星空**
* **宏景人力资源系统**
* **Thinkphp**

插件名称：Wappalyzer，这个插件在谷歌和火狐上应该都有，它可以查看当前网站所使用的语言，框架，调用库等信息，但并不能识别所有的框架，所以有时候还是要我们自己去做一个额外的信息收集的。

### ****5. C段扫描****

C段是IP的第三串数字，也就是192.168.1.0的1

192.168.2.1

192.168.3.1

192.168.4.1

192.168.5.1

### ****6. 网站漏洞扫描****

AWVS、yakit

## ****信息收集常用工具与网站****

### ****常用查询网站****

* **鹰图平台**：[hunter.qianxin.com](https://hunter.qianxin.com/)
* **fofa测绘平台**：[fofa.info](https://fofa.info/)
* **小蓝本企业资产查询**：[xiaolanben.com](https://www.xiaolanben.com/)
* **ICP备案查询**：[beian.miit.gov.cn](https://beian.miit.gov.cn/)

### ****常用工具****

* **nmap**（网络扫描工具）
* **Goby**（红队社区版）
* **Httpx** (信息收集)
* **灯塔系统**（樱花猫二次开发）

## ****第二阶段：十大漏洞的详细利用****

在信息收集后，我们进入**漏洞利用**阶段。本节详细介绍十大常见漏洞的基本原理、利用方式及修复建议。

### 1. SQL 注入 (SQL Injection)

**原理**：  
SQL 注入漏洞是 Web 应用程序中最常见、最危险的漏洞之一，主要源于应用在处理用户输入数据时未对其进行充分的过滤和验证。开发人员在构建 SQL 查询时，如果直接将用户输入拼接到查询语句中，攻击者就可以构造出恶意的 SQL 代码，从而改变查询逻辑。例如，在登录验证过程中，攻击者可以输入包含特殊字符的语句，使得 SQL 查询条件永远成立，从而绕过认证。攻击者不仅可以窃取数据库中的敏感信息（如用户账号、密码、信用卡信息等），还可能通过联合查询（UNION SELECT）、堆叠查询等方式获取更多数据，甚至修改或删除数据库中的数据。SQL 注入的危害性非常大，若利用得当，攻击者可能完全控制目标系统。  
此外，SQL 注入的变种形式还包括基于时间的盲注、错误信息注入等，这些攻击方式能在没有明显错误提示的情况下，逐步探测出数据库的结构和内容。由于很多应用程序开发时为了追求效率而忽略安全防范措施，导致 SQL 注入漏洞屡见不鲜。攻击者通过自动化工具可以迅速检测并利用此类漏洞。  
在实际应用中，SQL 注入常常发生在登录页面、搜索框、URL 参数等处。攻击者通过构造特定输入，如 admin' OR '1'='1，使得查询条件始终为真，从而实现绕过身份验证。甚至利用 UNION SELECT 语法，将原始查询与另一个查询合并，提取数据库中的敏感数据。  
**利用**：  
例如，攻击者在 URL 参数中注入恶意语句：

?id=1' OR '1'='1

?id=-1 UNION SELECT username, password FROM users --

通过这种方式，攻击者能够让数据库返回全部用户数据，而不是只返回合法记录。

SQLMAP，SQL注入工具  
**修复建议**：  
(1）使用预编译语句，使用 PDO 需要注意不要将变量直接拼接到 PDO 语句中。所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到 SQL 语句中。当前几乎所有的数据库系统都提供了参数化 SQL 语句执行接口，使用此接口可以非常有效的防止 SQL 注入攻击。

（2）对进入数据库的特殊字符（’”<>&\*;等）进行转义处理，或编码转换。

（3）确认每种数据的类型，比如数字型的数据就必须是数字，数据库中的存储字段必须对应为 int 型。

（4）数据长度应该严格规定，能在一定程度上防止比较长的 SQL 注入语句无法正确执行。

（5）网站每个数据层的编码统一，建议全部使用 UTF-8 编码，上下层编码不一致有可能导致一些过滤模型被绕过。

（6）严格限制网站用户的数据库的操作权限，给此用户提供仅仅能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。

（7）避免网站显示 SQL 错误信息，比如类型错误、字段不匹配等，防止攻击者利用这些错误信息进行一些判断。

（8）过滤危险字符，例如：采用正则表达式匹配 union、sleep、and、select、load\_file 等关键字，如果匹配到则终止运行。

### 2. XSS (跨站脚本攻击)

**原理**：  
跨站脚本攻击（XSS）是一种通过在网页中注入恶意脚本代码，从而在用户浏览器中执行的攻击手法。攻击者利用目标网站未对用户输入进行严格过滤的缺陷，将 JavaScript 等脚本嵌入到网页内容中。当其他用户浏览这些网页时，嵌入的恶意脚本会自动执行，从而窃取用户的 Cookie、会话令牌甚至敏感信息，严重时还可能利用用户身份进行欺诈操作。XSS 攻击常见的类型包括存储型、反射型和 DOM 型。存储型 XSS 攻击通常发生在评论区、论坛等内容会持久化存储的位置；反射型 XSS 则依赖于 URL 参数或表单提交，攻击代码在服务器响应中直接返回；而 DOM 型 XSS 则利用浏览器端的脚本处理漏洞直接在客户端执行恶意代码。  
XSS 攻击之所以危险，不仅在于信息窃取，还在于它可以改变页面内容，使得用户无法分辨真假，从而造成更大范围的社会工程学攻击。攻击者常常借助 XSS 漏洞在页面中植入钓鱼链接、恶意广告甚至劫持用户会话。由于 XSS 攻击的执行环境在用户浏览器中，因此修复建议必须兼顾前端和后端。  
**利用**：  
常见的 XSS 利用方式是在输入框或 URL 参数中注入 <script>alert('XSS!')</script>，一旦该内容未经处理直接输出到网页中，浏览器就会执行其中的 JavaScript 代码。  
**修复建议**：  
（1）过滤输入的数据，对例如：“ ‘ ”，“ “ ”，” < “，” > “，” on\* “，script、iframe 等危险字符进行严格的检查。这里的输入不仅仅是用户可以直接交互的输入接口，也包括 HTTP 请求中的 Cookie 中的变量，HTTP 请求头部中的变量等。

（2）不仅验证数据的类型，还要验证其格式、长度、范围和内容。

（3）不仅在客户端做数据的验证与过滤，关键的过滤步骤在服务端进行。

（4）对输出到页面的数据进行相应的编码转换，如 HTML 实体编码、JS 编码等。对输出的数据也要检查，数据库里的值有可能会在一个大网站的多处都有输出，即使在输入做了编码等操作，在各处的输出点时也要进行检查。

### 3. CSRF (跨站请求伪造)

**原理**：  
跨站请求伪造（CSRF）是一种利用用户在已登录状态下的信任关系，诱导其在不知情的情况下执行恶意请求的攻击方式。CSRF 攻击的基本原理是，攻击者通过构造伪造的 HTTP 请求，使得受害者的浏览器在访问恶意页面时自动向目标网站发送请求，并携带受害者的身份认证信息（如 Cookie）。由于浏览器会自动附加 Cookie，目标网站便认为这是合法用户的正常操作，从而执行诸如转账、更改密码、修改账户信息等敏感操作。CSRF 攻击往往针对那些未对请求进行额外验证的接口，尤其是状态变更类操作。该攻击方式的成功往往依赖于用户的登录状态和缺乏有效的验证机制，因此威胁性极大。  
攻击者通常会在邮件、论坛、博客或钓鱼网站中嵌入恶意代码，如隐藏表单或图片请求。当受害者点击链接或浏览页面时，恶意请求会在后台自动提交，绕过常规身份验证。由于该攻击是利用浏览器的自动行为，用户往往毫无察觉，攻击后果严重。  
**利用**：  
例如，攻击者可以在恶意网页中嵌入如下表单：

html

<form action="http://victim.com/change-password" method="POST" style="display:none;">

<input type="hidden" name="password" value="hacked123">

<input type="submit" value="Submit"> </form> <script>document.forms[0].submit();</script>

当受害者访问该页面时，其浏览器会自动提交请求，导致密码被修改或其他敏感操作被执行。  
**修复建议**：

1、验证请求的 Referer 是否来自本网站，但可被绕过。

2、在请求中加入不可伪造的 token，并在服务端验证 token 是否一致或正确，不正确则丢弃拒绝服务。

### 4. 文件上传漏洞

**原理**：  
文件上传漏洞主要存在于允许用户上传文件的 Web 应用中。由于上传功能在实际业务中非常常见（如上传头像、附件等），开发人员往往只对文件扩展名或 MIME 类型进行简单判断，导致攻击者可以上传包含恶意代码的文件。攻击者常常利用这一点，将包含 WebShell 或其他后门程序的文件伪装成图片、文档等正常文件上传至服务器。上传成功后，如果服务器对文件执行权限控制不严格，攻击者便可以直接通过浏览器访问该文件，从而执行其中的恶意代码，实现远程命令执行或系统控制。文件上传漏洞不仅能导致数据泄露，还可能使整个服务器受到控制。该漏洞的出现往往与服务器配置、文件存储策略以及输入验证不严密密切相关。  
在利用过程中，攻击者可能会对上传文件的名称、后缀进行巧妙修改，如将文件命名为 shell.php.jpg，以混淆系统对文件类型的判断。攻击者还可能利用绕过文件大小、文件内容检测等漏洞，进一步提高上传恶意文件的成功率。  
**利用**：  
攻击者可构造如下恶意文件：

<?php system($\_GET['cmd']); ?>

上传后，通过访问上传路径和传递参数 cmd，即可执行服务器上的任意命令，从而获取敏感信息或完全控制服务器。  
**修复建议**：

1、对上传文件类型进行验证，除在前端验证外在后端依然要做验证，后端可以进行扩展名检测，重命名文件，MIME 类型检测以及限制上传文件的大小等限制来防御，或是将上传的文件其他文件存储服务器中。

2、严格限制和校验上传的文件，禁止上传恶意代码的文件。同时限制相关上传文件目录的执行权限，防止木马执行。

3、对上传文件格式进行严格校验，防止上传恶意脚本文件；

4、严格限制上传的文件路径。

5、文件扩展名服务端白名单校验。

6、文件内容服务端校验。

7、上传文件重命名。

8、隐藏上传文件路径。

### 5. 远程代码执行 (RCE)

**原理**：  
远程代码执行（RCE）漏洞是一种极其严重的漏洞，它允许攻击者在服务器上执行任意代码，从而获得完全控制权。RCE 漏洞通常出现在应用程序直接调用系统命令或执行不可信用户输入的场景中。开发人员在实现功能时，如果将用户输入未经严格验证直接传递给系统调用函数（如 PHP 中的 exec、system 等），便可能使攻击者注入恶意命令。利用 RCE，攻击者可以执行任意操作，包括读取敏感文件、修改数据库、安装恶意软件或进行横向移动。由于 RCE 漏洞直接影响到服务器底层操作系统，其危害性远超其他应用层漏洞。一旦被攻击，整个系统可能完全沦陷，给企业和用户带来巨大损失。  
在实际场景中，RCE 漏洞不仅存在于 Web 应用程序，还可能出现在网络服务、应用服务器和 API 接口中。攻击者通常通过构造恶意请求，借助漏洞执行远程命令，获取系统信息或植入后门，形成持续控制。RCE 漏洞的检测与防护要求开发人员对输入数据进行严格校验，并对系统调用进行白名单限制。  
**利用**：  
例如，在 PHP 应用中，代码若如下直接调用用户输入：

?Id=exec($\_GET['cmd']);&cmd=ls

攻击者可以通过传递 cmd=whoami 等参数，在服务器上执行命令，从而获得系统信息。更复杂的攻击可能通过构造长链命令，实现数据窃取或系统破坏。  
**修复建议**：  
1、严格过滤用户输入的数据，禁止执行非预期系统命令。

2、减少或不使用代码或命令执行函数

3、客户端提交的变量在放入函数前进行检测

4、减少或不使用危险函数

### 6. SSRF (服务器端请求伪造)

**原理**：  
服务器端请求伪造（SSRF）是一种攻击手法，利用 Web 应用程序作为代理，迫使服务器向攻击者指定的目标地址发起请求。由于服务器通常拥有访问内网资源的权限，攻击者可以通过 SSRF 漏洞实现内网扫描、获取敏感信息或执行后续攻击。SSRF 攻击的核心在于，服务器在处理外部请求时未对目标 URL 进行严格校验，导致攻击者能够构造恶意请求，借助服务器访问受防火墙保护的内网服务。攻击者可以利用这一漏洞探测内部网络结构、访问管理接口，甚至结合 DNS 重绑定等技术进一步扩大攻击范围。由于请求发起源自合法服务器，传统防护手段往往难以检测到 SSRF 攻击的异常。  
攻击者在实际利用中，可能在允许用户输入 URL 的接口中注入类似 http://127.0.0.1/admin 的地址，诱使服务器访问内网地址，获取到本不应公开的数据。攻击者也可以通过构造多个连续请求，进一步探索内部网络架构和安全漏洞。  
**利用**：  
例如，攻击者构造请求：

http://localhost:8080/admin

服务器在接收到该请求后，会将请求转发至内网地址，从而可能返回敏感信息给攻击者。  
**修复建议**：  
1、禁用不需要的协议，只允许 HTTP 和 HTTPS 请求，可以防止类似于 file://, gopher://, 等引起的问题。

2、白名单的方式限制访问的目标地址，禁止对内网发起请求

3、过滤或屏蔽请求返回的详细信息，验证远程服务器对请求的响应是比较容易的方法。如果 web 应用是去获取某一种类型的文件。那么在把返回结果展示给用户之前先验证返回的信息是否符合标准。

4、验证请求的文件格式

5、禁止跳转

6、限制请求的端口为 http 常用的端口，比如 80、443、8080、8000 等

7、统一错误信息，避免用户可以根据错误信息来判断远端服务器的端口状态。

### 7. 逻辑漏洞

**原理**：  
逻辑漏洞源于业务流程设计不合理或实现上的漏洞，通常并非单纯的编程错误，而是由于设计上缺乏严密的逻辑校验，导致系统在处理特定业务流程时出现安全漏洞。逻辑漏洞常见于权限验证、数据处理、订单支付等关键业务环节，攻击者通过逆向分析系统的业务逻辑，可以构造恶意请求绕过安全检查。例如，在某些电商平台中，如果系统没有对价格参数进行严格校验，攻击者可能通过修改隐藏字段实现低价购买高价值商品；在权限管理中，如果不同用户的权限边界模糊，攻击者可能通过修改请求数据获取未授权访问权限。逻辑漏洞的危害不仅在于单个漏洞的利用，更在于它可能导致整个业务流程崩溃，破坏系统的信任机制。由于逻辑漏洞往往不依赖于特定的输入格式，其检测和防御难度较大，需要开发者对业务流程进行全方位的安全审查。  
攻击者利用逻辑漏洞时，通常会深入分析系统流程，利用系统未考虑到的异常操作路径，达到绕过安全机制的目的。攻击方式可能包括修改请求参数、篡改客户端数据或利用多次请求组合来突破验证。逻辑漏洞的修复往往需要重新设计业务流程，并对所有关键操作增加严格的服务器端验证。  
**利用**：  
例如，在订单支付系统中，攻击者可能通过修改隐藏参数将订单金额调至极低值，从而实现非法获利。或在权限验证过程中，通过构造特定请求，使得低权限用户获取管理员权限。  
**修复建议**：  
防御逻辑漏洞要求开发者对业务流程进行严格审查，确保每一步操作均在服务器端进行验证。常见措施包括：所有关键操作必须在服务器端进行二次校验，不信任客户端数据；使用多重验证机制确保操作合法性；建立详细的业务日志，对异常操作进行及时监控和告警。只有通过全面的流程安全设计，才能有效避免逻辑漏洞带来的安全隐患。

### 8. 弱口令

**原理**：  
弱口令问题是信息安全中最为普遍的问题之一，主要体现在用户或管理员设置的密码过于简单、易于猜测。由于很多用户为了记忆方便，经常采用诸如“123456”、“password”或“admin123”等弱口令，这使得攻击者可以利用暴力破解、字典攻击或凭借已知的泄露密码，快速获取账户访问权限。弱口令不仅使得单个账户容易被攻破，还可能引起横向扩散，导致整个系统受到威胁。特别是在一些系统中，管理员账户常使用默认密码或简单密码，一旦被攻击者破解，可能导致系统完全失控。弱口令问题反映出用户安全意识薄弱和系统密码策略不健全，常常是被攻击者首选利用的突破口。  
攻击者通常利用自动化工具进行大量尝试，结合已知的密码库进行撞库攻击，极短时间内即可突破弱密码账户。一旦获得账户权限，攻击者不仅可以窃取敏感数据，还可能进一步利用该账户权限实施更大范围的攻击，如植入恶意代码、修改系统配置等。  
**利用**：  
举例来说，攻击者可使用字典攻击工具，尝试成千上万的常见密码组合，快速获得管理员或普通用户的登录权限。一旦突破，即可对系统进行进一步的渗透攻击。  
**修复建议**：

1、强制用户首次登录时修改默认口令，或是使用用户自定义初始密码的策略；

2、完善密码策略，信息安全最佳实践的密码策略为 8 位（包括）以上字符，包含数字、大小写字母、特殊字符中的至少 3 种。

3、增加人机验证机制，限制 ip 访问次数。

# ****第三阶段：编辑报告的技巧****

一个好的src报告需要以下几点要素

### 标题：

某某某单位/学校存在XXX漏洞

### 漏洞URL：

漏洞所在网站的网址

### 归属证明：

证明此资产属于目标的资产之内，可以贴上ICP备案的截图，或者是网站内容的相关截图

### 漏洞原理：

写上这个是一是为了预防遇到傻子审核，二是为了开发人员精准定位漏洞成因，是个很好的加分项，为什么说要预防傻子审核呢，因为有很多SRC的审核是只懂皮毛或者完全不懂的，并且网安分支庞大，审核也不可能什么漏洞都能看懂，所以这里是要解释给审核人员，让审核感到专业性，可以在后续讲解漏洞危害时概率提升漏洞危害等级

### 漏洞复现的过程：

贴上从发现此资产到如何利用该漏洞所用到的所有代码和截图即可，这一可以验证代码是否存在的流程也叫POC

### 漏洞危害：

漏洞原理就是为此处铺垫的，漏洞危害一定要客观且用大白话说明，不要脑补夸大危害，否则会被审核评判为恶意夸大危害而降级漏洞危害

### 修复建议：

这里写上几点如何修复此漏洞即可。

## 总结

本场直播从渗透测试到提交报告的角度出发，详细介绍了信息收集和漏洞利用两个阶段。特别是在漏洞利用阶段，我们针对 SQL 注入、XSS、CSRF、文件上传漏洞、远程代码执行、XXE、SSRF、逻辑漏洞、弱口令等常见漏洞，分别从原理、利用方式及修复建议三个方面进行了简略阐述，主要目的是为了让初入网安的新生力量挖掘到属于自己的第一个漏洞。