1 实验目的和内容

1.1 实验目的

在合法范围内，实施一次 SQL 注入攻击（可自己搭建目标机器）

1.2 实验内容

SQL 注入是一种网络安全漏洞，攻击者可以通过该漏洞干扰应用程序对其数据库的查询。通常情况下，攻击者可以利用该漏洞获取到一些敏感数据，甚至修改或删除数据。常见的 SQL 注入有：检索隐藏数据、破怀应用逻辑、UNION 攻击、检索数据库、盲 SQL 注入。本实验将对上述的五种 SQL 注入进行分析，并在最后提出综合的解决方案，最后将借助所写的脚本进行攻击。

2 实验环境

2.1 目标机器的环境

虚拟机：VMware WorkStation 15 Pro
操作系统：Ubuntu 18.04 64 位
处理器内核总数：4
分配给虚拟机内存大小：8GB
编辑器：Vscode 1.42.1
mysql Ver 14.14 Distrib 5.7.30
PHP 7.2.24-0ubuntu0.18.04.4（cli）
Apache/2.4.29（Ubuntu）
Python 3.7.3

2.2 目标机器的构建 LAMP

本课程作业搭建的目标机器运行在虚拟机 VMware 上的 Ubuntu18.04，然后进行配置 LAMP 环境。首先,通过"sudo apt-get install apache2"命令安装 apache2；然后，输入命令"sudo apt-get install mysql-server mysql-client"，安装 MySQL，修改用户密码；最后，安装 PHP，输入命令"sudo apt-get install php7.0 libapache2-mod-php"，"sudo apt-get install php7.2-mysql"，并进行相应的配置操作。

在本地创建数据库 secure_info，创建表 users，之后的所有操作将对 users 表进行攻击。采用 PHP 进行网页编程，main.php 为目标网站主页面，选择 low，improved 两个选项，进入不同安全等级的网页，其中 low 安全等级最低，对输入的用户 ID 不进行任何过滤操作，improved 则对输入数据进行特殊字符过滤和是否为数字的检测。

将 projet 移至/var/www/SQL_injection/下，即可访问网站。访问网址是 http://192.168.254.147/SQL_injection/main.php#，其中 192.168.254.147 是 Ubuntu18.04 的 IPv4 地址，下图是数据库 secure_info 中的 users 表的全部信息(其中 cancel 列，值为 1，表示已被注销，查询时不显示，值为 0，表示未注销)：

3 SQL 注入

选择"low"选项，此时不对查询语句进行任

user_id	first_name	last_name	user	password	last_login	cancel	email	sex
1	admin	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99	2020-05-28 08:03:03	0	1120175664@bit.edu.cn	woman
2	Gordon	Brown	gordonb	e99a18c428cb38d5f260853678922e03	2020-05-28 08:02:58	0	1120134564@bit.edu.cn	woman
3	Hack	Me	1337	8d3533d75ae2c3966d7e0d4fcc69216b	2020-05-28 08:03:11	1	112564@bit.edu.cn	man
4	Pablo	Picasso	pablo	0d107d09f5bbe40cade3de5c71e9e9b7	2020-05-28 08:03:21	1	1120175664@bit.edu.cn	woman
5	Bob	Smith	smithy	5f4dcc3b5aa765d61d8327deb882cf99	2020-05-28 08:03:14	0	1120001342@bit.edu.cn	man

图 2.2-1 数据库 secure_info 中的 users 表的全部信息

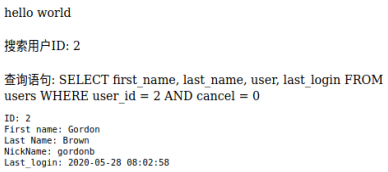

图 3.0-1 选择安全模式　图 3.0-2 low 模式下主页面　　图 3.0-3 查询用户 1 的返回结果

何过滤操作，查询操作有两种：通过输入账号和密码，登录用户账号，或者通过用户 ID 查询用户信息（见图 3.0-1，图 3.0-2，图 3.0-3）。常见的 SQL 注入有五种：检索隐藏数据库、破坏应用逻辑、检索数据库、UNION 攻击，和盲 SQL 注入。下述内容将对 low 级别的网页进行上述五种攻击，并提出综合的解决方案。

3.1 检索隐藏数据

检索隐藏数据(Retrieving hidden data),一般情况下，借助 ’ # ’ /’ --’ (注释)， ’ ’ ’ 等特殊字符,将查询语句的后半部分注释掉，达到去除部分限制条件的目的，并在’ # ’ 前，’ ’ ’ 后加入某个 sql 语句，返回隐藏数据。

在 low 模式下，输入″3 or 1 = 1#″，见图 3.1-1。返回结果见图 3.1-4，可以看到查询语句为 "SELECT first_name, last_name, user, last_login FROM users WHERE user_id = 3 or 1 = 1# AND cancel = 0", "#"将" AND cancel = 0"这部分的限定条件注释了，所以无返回结果必须为未注销的用户的限制，而 " or 1 =1"，条件永真，即无' user_id = 3 '限制条件，返回表中所有数据。

3.2 破怀应用逻辑

以下面例子说明破坏应用逻辑(Subverting application logic)。进入个人主页，在用户 ID 中输入" 3'# "，不输入用户密码，可以直接登入个人主页。可以看到查询语句变为" SELECT user_id, first_name, last_name, user, last_login, sex, email FROM users WHERE user_id = 3 # AND password=" AND cancel = 0 LIMIT 1"，" # "将后面的语句全部注释了，不需要输入密码也能进入个人主页，

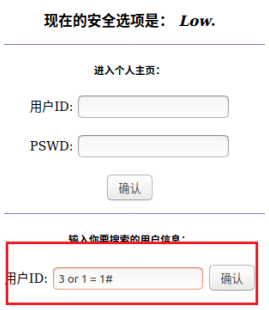

图 3.1-1 输入 3’ or 1 = 1#

图 3.1-2 返回结果

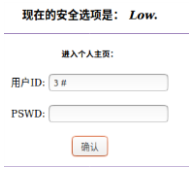

图 3.2-1 在用户 ID 中输入 3 #，不输入密码

图 3.2-2 返回结果

图 3.3-1 在第二栏的用户 ID 中输入有关信息

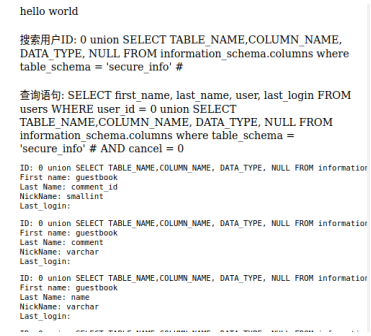

图 3.3-2 返回结果

破坏原有账号-密码的应用逻辑，成功以攻击者身份登入系统。

3.3 检索数据库

大多数的数据库软件(除了 Oracle)，都有一组称为 information_schema 的视图，这些视图提供有关数据库的信息，可以通过查询 information_schema.tables 来获取我们需要的表的信息，例如表名，列名等。如图 3.3-1 所示，输入 "0 union SELECT TABLE_NAME,COLUMN_NAME, DATA_TYPE, NULL FROM information_schema.columns where table_schema = 'secure_info' #，点击确认。图 3.3-2 展示了返回结果，查询语句为"SELECT first_name, last_name, user, last_login FROM users WHERE user_id = 0 union SELECT TABLE_NAME,COLUMN_NAME, DATA_TYPE, NULL FROM information_schema.columns where table_schema = 'secure_info' # AND cancel = 0，这里同样借助了注释符，其中 NULL 可以接收任何类型的数据，可以看到原本的 First Name，Last Name, NickName，现在分别对应于 TABLE_NAME,COLUMN_NAME, DATA_TYPE，成功获取得到数据库中表的信

图 3.4-1 在第一栏的用户 ID 中输入有关信息

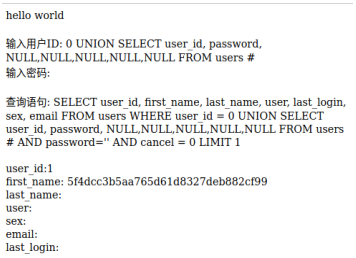

图 3.4-2 返回结果

息，为后续进一步查询具体信息做准备。

3.4 UNION 攻击

当应用程序可被 SQL 注入，并且查询结果在应用程序的响应中返回时，可通过 UNION 关键字执行一条或多条关键字，并将结果对应于原始查询中。UNION 攻击必须满足两个条件，第一，各个查询必须返回相同数目的列；第二，每列中的数据类型必须在各个查询之间兼容。第一个条件可以利用"1 order by 1 #", "1 order by 2", "1 order by 3"等语句获取得到返回列数，具体来说，ORDER BY 子句中的列可以由其索引指定，因此不需要任何列名，当列数超过结果集的实际列数时，数据库将返回一个错误或者显示无结果；第二个条件可以利用 NULL，原因在于 NULL 可转换为任一常见的数据类型。

例如，在登录个人页面中，在用户 ID 中输入" SELECT user_id, first_name, last_name, user, last_login, sex, email FROM users WHERE user_id = 0 UNION SELECT user_id, password, NULL,NULL,NULL,NULL,NULL FROM users # AND password='' AND cancel = 0 LIMIT 1 "，不输入密码，直接点击确认，见图 3.4-1，返回结果见图 3.4-2。可以看到原本的 user_id, First Name，现在分别对应于 user_id, password 成功获取得到 users 表的用户 ID 和用户密码。

3.5 盲 SQL 注入

当应用程序可被 SQL 注入但其 HTTP 响应不包含相关 SQL 查询的结果或任何数据库错误的详细信息时，大多数注入技术将无效，这时可以采用盲 SQL 注入技术。常见有盲注入技术有基于布尔型盲注，基于时间盲注，基于报错盲注等。本实验的 SQL 注入脚本中采用了基于布尔型盲注和基于时间盲注。大体来说，盲 SQL 注入的流程可分为三个步骤：爆库，爆表，脱库。爆库指获取数据库的信息，例如数据库的名字；爆表指获取数据表的有关信息，例如要获取数据所在的数据表的名字，数据表中的列名，数据表中的行数；脱库指获取数据库中具体信息。每一步获取信息主要不断借助脚本来推测所需信息的某个字符是 ASCII 码中的哪一个，借此达到获取数据的目的。

3.5.1 脚本说明

本脚本使用 Python 语言编写，调用库 requests,一共四个脚本: db_name.py,

db_table_name.py，db_table_column.py, db_table_data. py, 其中 db_table_name. py 是基于时间的盲注，其余都是基于布尔型的盲注。

基于布尔型的盲注，以 db_table_data.py 为例。函数 get_table_data_length()确定密码长度，函数 get_table_data(length:int)确定密码，这里对函数 get_table_data(length:int)进行说明。基本网址为" http://192.168.254.147/SQL_injection/low.php? %s &Submit_ID=确认 "，其中%s 对应于 "id=1 and ASCII(SUBSTR("(SELECT password FROM users LIMIT 0,1),%d,1))=%d ",第一个%d 从 1 递增到需获取数据的长度，即从左到右依次扫描该数据的每一个字符，每次扫描一个，第二个%d 对应于当前猜测的 ASCII 码，若等于当前扫描到的字符的 ASCII 码，则为 true，否为 false，而这两种结果对应的返回结果的字符串长度不同，通过判断字符串长度间接判断是否为所猜测的 ASCII 码。

基于时间的盲注，主要借助 sleep()函数，当程序等待时间超过所设定的阈值时，可以判断此字符为结果字符串的组成。以 db_table_name. py 为例，函数 get_table_name_length()确定表名长度，函数 get_table_name(length:int)确定表名，这里以函数 get_table_name(length:int)为例。基本网址与基于布尔型的盲注的基本网址相同，其中%s 对应于 ″id=1 AND IF((ASCII(SUBSTR((SELECT table_name FROM information_schema.tables WHERE table_schema=DATABASE() LIMIT 1,1),%d,1)))=%d,SLEEP(3),0) ″，与基于布尔型的盲注类似，第一个%d 对应于每次扫描结果字符串中的一个字符，第二个%d 对应于当前猜测的 ASCII 码，若等于当前扫描到的字符的 ASCII 码，则为 true，延时 3 秒，否则为 false，不延时。调用函数 requests.get(url).elapsed.seconds 进行计时，通过延时间接判断是否为所猜测的 ASCII 码。

3.5.2 脚本运行

首先爆库。运行 db_name.py，获取数据库名字。

然后爆表。运行 db_table_name.py，获取所需信息所在数据表的表名。运 db_column_name.py，获取所需信息所在表的列名。

最后脱库。运行 db_table_data.py，确定自己要获取信息在数据表的行数和长度，获取所需信息的内容。

4 解决方案

避免 SQL 注入的主要方法是对输入语句进行处理，不直接将输入语句与 SQL 查询语句进行拼接。常用的两步处理有对输入语句的特殊字符加转义符，判断输入类型本应为数字的数据是否为数字。此处以安全等级为 improved 的目标网页为例，调用 mysqli_real_escape_string 函数，对可能出现的单引号进行转义，即"'"-> "\'"。除此之外，存在输入框接收数据类型为数字的情况，即可输入″2 AND 1=1 #″，进行攻击，这时 mysqli_real_escape_string 函数对其无效，所以此时调用 is_numeric 函数进一步判断所输入数据是否为数字。

利用脚本和人工输入对 improved. php 网页进行攻击，无输出结果，即上述两步处理能防范常见的 SQL 注入攻击。