跨站脚本攻击(Cross Site Scripting)，为了不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为XSS。Web程序代码中把用户提交的参数未做过滤就直接输出到页面，参数中的特殊字符打破了HTML页面的原有逻辑，黑客可以利用该漏洞执行恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。

0x01 XSS

最简单的一个案例，输入即输出。

漏洞代码示例：

```
<?php
    echo $_REQUEST[ 'id' ];
?>
```

测试语句：`?id=<script>alert(/xss/)</script>`

0x02 编码解码

编码解码输出时，可能导致XSS编码绕过的情况

漏洞代码示例：

```
<?php
    $a=urldecode($_GET['id']); //接收参数并进行url解码
    $b=htmlspecialchars($a);   //HTML ENCODE处理,到这里都是没有问题的
    echo urldecode($b);        //最后，url解码输出
?>
```

测试语句：`id=%25253Cscript%25253Ealert(/xss/)%25253C/script%25253E`

这边代码逻辑中，问题根源在于最后一句的url解码输出，导致存在三重url编码绕过的情况。

根据实际情况，给出安全建议：HTML ENCODE处理后直接输出变量。

0x03 HTML不规范

HTML代码编写不规范，可能导致的问题，我们来看一个案例：

漏洞代码示例：

```
<?php
    $name = htmlspecialchars($_GET['name']);
?>

<input type='text' class='search' value='<?=$name?>'>
```

获取参数，在一个input元素的属性里输出这个变量，我们注意到这里使用的是单引号闭合，而函数默认只是转化双引号(")，不对单引号(')做转义。

因此，可以用单引号闭合，

测试语句：?name=222' onclick='alert(/xxs/)

安全建议：将HTML标签的属性值用双引号引起来。

0x04 黑名单过滤

通过在全局引入过滤函数，提供黑名单过滤，

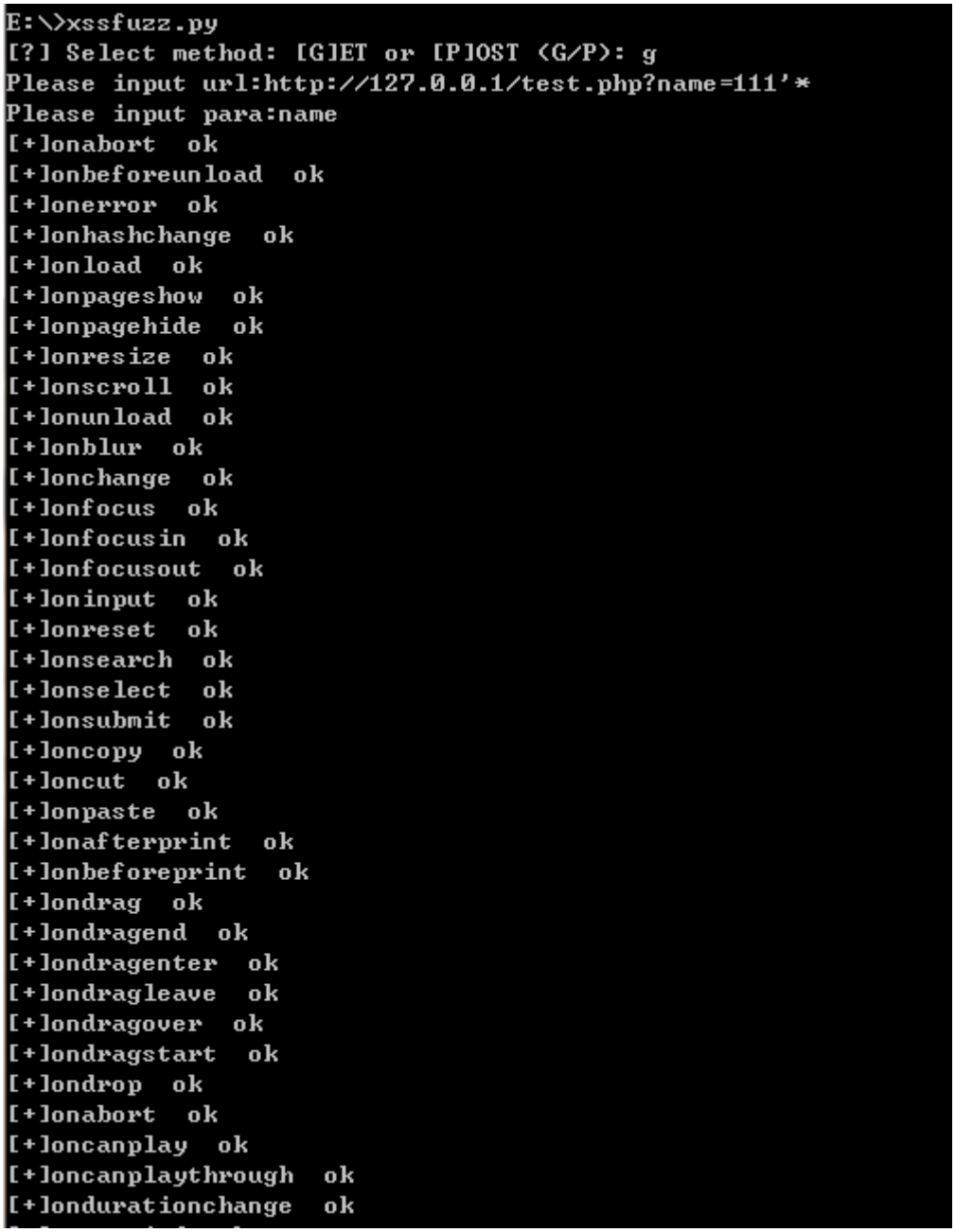
漏洞代码示例：

```
<?php
    $name = htmlspecialchars($_GET['name']);
    $pregs = "/<script>|
<\/script>|onclick|oncontextmenu|ondblclick|onmousedown|onmouseenter|onmouseleave|onmousemo
ve|onmouseover|onmouseout|onmouseup|onkeydown|onkeypress|onkeyup/i";

    $check = preg_match($pregs, $name);
    if ($check) {
           echo 'not found';
           exit;
    }
?>

<input type='text' class='search' value='<?=$name?>'>
```

从html编写不规范，我们可以使用单引号闭合，然后去进一步构造触发事件，可是常见的XSS事件大多都被过滤了，

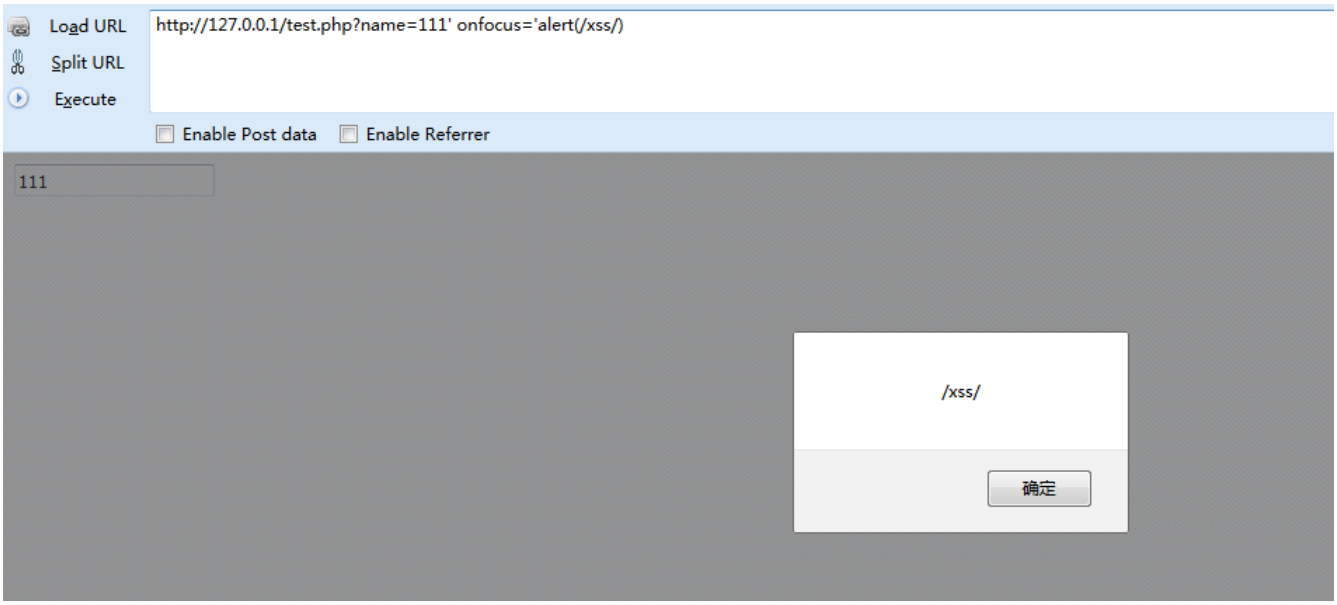
怎么快速地去找到可以拿来利用的XSS触发事件呢? 答：XSS FUZZ。

前提是要收集积累一些触发事件，利用自己编写python脚本进行fuzz



虽然fuzz出来很多事件，但要结合具体输出位置去分析，找到合适的事件，最终得出Payload。

测试语句：?name=111' onfocus='alert(/xss/)



0x05 漏洞防护

1、PHP提供了两个函数htmlentities()和htmlspecialchars()，把一些预定义的字符转换为 HTML 实体。

防御代码示例:

```
<?php
    echo htmlspecialchars($_REQUEST[ 'id' ]);
?>
```

2、其它的通用的补充性防御手段

```
1.在输出html时，加上Content Security Policy的Http Header
（作用：可以防止页面被XSS攻击时，嵌入第三方的脚本文件等）
（缺陷：IE或低版本的浏览器可能不支持）
2.在设置Cookie时，加上HttpOnly参数
（作用：可以防止页面被XSS攻击时，Cookie信息被盗取，可兼容至IE6）
（缺陷：网站本身的JS代码也无法操作Cookie，而且作用有限，只能保证Cookie的安全）
3.在开发API时，检验请求的Referer参数
（作用：可以在一定程度上防止CSRF攻击）
（缺陷：IE或低版本的浏览器中，Referer参数可以被伪造）
```

附XSS FUZZ 脚本:

```
#! /usr/bin/env python
# _*_  coding:utf-8 _*_

import requests
import urlparse
import urllib

# 使用说明，修改第54行处的字典即可使用，支持GET、POST等简单XSS验证
# url 支持 * 号  如  http://127.0.0.1/test.php?id=1*3333    payload会替换*号内容
```

```
global result_dict
result_dict={}

def get(url,para,payload):
    params={}
    result=urlparse.urlparse(url)
    params=urlparse.parse_qs(result.query,True)
    if '*' in params[para][0]:
        params[para]=str(params[para][0]).replace("*", payload);
    else:
        params[para]=str(params[para][0])+payload
    m_url=result.scheme+"://"+result.netloc+result.path
    data = urllib.urlencode(params)
    geturl = m_url+'?'+data
    response = requests.get(geturl)
    result_dict[payload]=[response.content,len(response.content),response.status_code]
    return result_dict


def make_get_resule(url,para):
    with open('on.txt') as f:
        for payload in f.xreadlines():
            payload =payload.strip()
            if '#' in payload or len(payload)==0:
                pass
            else:
                get(url,para,payload)
                result_analysis(payload)

def post(url,data,para,payload):
    params={}
    params=urlparse.parse_qs(data,True)
    if '*' in params[para][0]:
        params[para]=str(params[para][0]).replace("*", payload);
    else:
        params[para]=str(params[para][0])+payload

    response = requests.post(url,data=params,timeout=5)
    result_dict[payload]=[response.content,len(response.content),response.status_code]
    return result_dict


def make_post_resule(url,data,para):
    with open('on.txt') as f:
        for payload in f.xreadlines():
            payload =payload.strip()
            if '#' in payload or len(payload)==0:
                pass
            else:
                post(url,data,para,payload)
                result_analysis(payload)
def result_analysis(payload):
```

```
    if  payload in result_dict[payload][0]:
        print "[+]" + payload +"  ok"

    if  result_dict[payload][0].count(payload)>1:
        print "[+]" + payload +"  repeat"

if __name__ == '__main__':
    result_dict={}
    methodselect = raw_input("[?] Select method: [G]ET or [P]OST (G/P): ").lower()
    if methodselect == 'g':
        url = raw_input("Please input url:")
        para = raw_input("Please input para:")
        if 'https://' in url:
            pass
        elif 'http://' in url:
            pass
        else:
            url = "http://"+url
        make_get_resule(url,para)

    elif methodselect == 'p':
        url = raw_input("Please input url:")
        data = raw_input("Please input data:")
        para = raw_input("Please input para:")
        if 'https://' in url:
            pass
        elif 'http://' in url:
            pass
        else:
            url = "http://"+url

        make_post_resule(url,data,para)
```

新文章将同步更新到我的个人公众号上，欢迎各位朋友扫描我的公众号二维码关注一下我，随时获取最新动态。